Security and privacy in RFID systems

Kaoutar Elkhiyaoui

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2012-ENST-040

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Kaoutar ELKHIYAOUI

12 Septembre 2012

Sécurité et Protection de la Vie Privée dans les Systèmes RFID

Directeur de thèse : **Refik MOLVA**

Jury

M. Fabien LAGUILLAUMIE — **Rapporteurs**
Mme Marine MINIER
M. Gildas AVOINE — **Examinateurs**
M. Srdjan ČAPKUN
M. Claude CASTELLUCIA
M. Bruno MARTIN
M. Serge VAUDENAY

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

THÈSE

Doctorat ParisTech

Thesis Dissertation

for the degree of Doctor of Science from

TELECOM ParisTech

Computer Science and Networks

presented by

Kaoutar ELKHIYAOUI

Defense date : September 12th, 2012

Security and Privacy in RFID Systems

Jury

Fabien LAGUILLAUMIE	**Reviewers**
Marine MINIER	
Gildas AVOINE	**Examiners**
Srdjan ČAPKUN	
Claude CASTELLUCIA	
Bruno MARTIN	
Serge VAUDENAY	
Refik MOLVA	**Thesis adviser**

TELECOM ParisTech

École de l'Institut Mines-Télécom - membre de ParisTech

To my parents, who have been there for me from day one,
To my brothers and sisters, who are synonym with joy and fun,
Thank you for all of your love, patience and support,

Acknowledgements

The work presented in this thesis would have not been accomplished without the support, encouragement and assistance of a great number of individuals.

First of all, I am deeply grateful to Prof. Refik Molva who has been an insightful mentor whose guidance made the past few years a thoughtful experience and a rewarding journey.

I am also indebted to Dr. Erik-Oliver Blass whose help and comments have been crucial to the establishment of many of the results presented in this thesis.

I would like to thank my jury committee of Gildas Avoine, Srdjan Čapkun, Claude Castellucia, Fabien Laguillaumie, Bruno Martin, Marine Minier and Serge Vaudenay who kindly agreed to review and examine this thesis.

I have also been blessed to be surrounded by such great friends at EURECOM: Aymen, Sabir, Siouar, Fatma, Wael who made my time at EURECOM synonym with fun, fruitful discussions and heated debates.

Last but certainly not least, I would like to thank EURECOM's staff for their availability, help and dedication.

Abstract

While RFID systems are one of the key enablers helping the prototype of pervasive computer applications, the deployment of RFID technologies also comes with new privacy and security concerns ranging from people tracking and industrial espionage to product cloning and denial of service. Cryptographic solutions to tackle these issues were in general challenged by the limited resources of RFID tags, and by the formalizations of RFID privacy that are believed to be too strong for such constrained devices. It follows that most of the existing RFID-based cryptographic schemes failed at ensuring tag privacy without sacrificing RFID scalability or RFID cost effectiveness.

In this thesis, we therefore relax the existing definitions of tag privacy to bridge the gap between RFID privacy in theory and RFID privacy in practice, by assuming that an adversary cannot *continuously* monitor tags. Under this assumption, we are able to design secure and privacy preserving multi-party protocols for RFID-enabled supply chains. Namely, we propose a protocol for tag ownership transfer that features constant-time authentication while tags are only required to compute hash functions. Then, we tackle the problem of product genuineness verification by introducing two protocols for product tracking in the supply chain that rely on storage only tags. Finally, we present a solution for item matching that uses storage only tags and aims at the automation of safety inspections in the supply chain.

The protocols presented in this manuscript rely on operations performed in sub-groups of elliptic curves that allow for the construction of short encryptions and signatures, resulting in minimal storage requirements for RFID tags. Moreover, the privacy and the security of these protocols are proven under well defined formal models that take into account the computational limitations of RFID technology and the stringent privacy and security requirements of each targeted supply chain application.

Résumé

Vu que les tags RFID sont actuellement en phase de large déploiement dans le cadre de plusieurs applications (comme les paiements automatiques, le contrôle d'accès à distance, et la gestion des chaînes d'approvisionnement), il est important de concevoir des protocoles de sécurité garantissant la protection de la vie privée des détenteurs de tags RFID. Or, la conception de ces protocoles est régie par les limitations en termes de puissance et de calcul de la technologie RFID, et par les modèles de sécurité qui sont à notre avis trop forts pour des systèmes aussi contraints que les tags RFID.

De ce fait, on limite dans cette thèse le modèle de sécurité; en particulier, un adversaire ne peut pas observer toutes les interactions entre tags et lecteurs. Cette restriction est réaliste notamment dans le contexte de la gestion des chaînes d'approvisionnement qui est l'application cible de ce travail. Sous cette hypothèse, on présente quatre protocoles cryptographiques assurant une meilleure collaboration entre les différents partenaires de la chaîne d'approvisionnement. D'abord, on propose un protocole de transfert de propriété des tags RFID, qui garantit l'authentification des tags en temps constant alors que les tags implémentent uniquement des algorithmes symétriques, et qui permet de vérifier l'authenticité de l'origine des tags. Ensuite, on aborde le problème d'authenticité des produits en introduisant deux protocoles de sécurité qui permettent à un ensemble de vérificateurs de vérifier que des tags "sans capacité de calcul" ont emprunté des chemins valides dans la chaîne d'approvisionnement. Le dernier résultat présenté dans cette thèse est un protocole d'appariement d'objets utilisant des tags "sans capacité de calcul", qui vise l'automatisation des inspections de sécurité dans la chaîne d'approvisionnement lors du transport des produits dangereux.

Les protocoles introduits dans cette thèse utilisent les courbes elliptiques et les couplages bilinéaires qui permettent la construction d'algorithmes de signature et de chiffrement efficaces, et qui minimisent donc le stockage et le calcul dans les systèmes RFID. De plus, la sécurité de ces protocoles est démontrée sous des modèles formels bien définis qui prennent en compte les limitations et les contraintes des tags RFID, et les exigences strictes en termes de sécurité et de la protection de la vie privée des chaînes d'approvisionnement.

Contents

List of Figures

List of Tables

Papers Published during PhD

Kaoutar Elkhiyaoui, Erik-Oliver Blass, Refik Molva. T-MATCH: Privacy-Preserving Item Matching for Storage-Only RFID Tags. RFIDsec'12, 8th Workshop on RFID Security and Privacy 2012, July 1-3, 2012, Nimegen, Netherlands. To be published as Springer "Lecture Notes in Computer Science".

Erik-Oliver Blass, **Kaoutar Elkhiyaoui**, Refik Molva. PPS: Privacy-Preserving Statistics using RFID Tags. 3rd IEEE Workshop on Data Security and PrivAcy in wireless Networks (D-SPAN) June 25, 2012, San Francisco, CA, USA.

Kaoutar Elkhiyaoui, Erik-Oliver Blass, Refik Molva. CHECKER: On-site Checking in RFID-based Supply Chains. WiSec 2012, 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks, April 16-18, 2012, Tucson, Arizona, USA.

Erik-Oliver Blass, **Kaoutar Elkhiyaoui**, Refik Molva, Olivier Savry, Cedric Verilhac. Demo: The F_f Hardware Prototype for Privacy-Preserving Authentication. CCS 2011, ACM Conference on Computer and Communications Security, October 17-21, 2011, Chicago, USA.

Mehdi Khalfaoui, **Kaoutar Elkhiyaoui**, Refik Molva. Privacy Preserving Product Tracking in Clustered Supply Chain. SENSORCOMM 2011, 5th International Conference on Sensor Technologies and Applications, August 21-27, 2011, Nice/Saint Laurent du Var, France.

Kaoutar Elklhiyaoui, Erik-Oliver Blass, Refik Molva. ROTIV: RFID Ownership Transfer with Issuer Verification. RFIDsec'11, 7th Workshop on RFID Security and Privacy 2011, June 26-28, 2011, Amherst, Massachusetts, USA. Also published as Springer "Lecture Notes in Computer Science".

Erik-Oliver Blass, **Kaoutar Elkhiyaoui**, Refik Molva. TRACKER: Security and Privacy for RFID-based Supply Chains. NDSS'11, 18th Annual Network and Distributed System

Security Symposium, 6-9 February 2011, San Diego, California, USA, ISBN 1-891562-32-0, pp 455-472.

Daishi Kato, **Kaoutar Elkhiyaoui**, Kazuo Kunieda, Keiji Yamada, Pietro Michiardi. A Scalable Interest-oriented Peer-to-Peer Pub/Sub Network. "Peer-to-Peer Networking and Applications Journal", Springer, 2010, pp 1-13.

Olivier Billet, **Kaoutar Elkhiyaoui**. Two Attacks against the F_f RFID Protocol. Indocrypt 2009, 10th International Conference on Cryptology, December 13-16, 2009, New Delhi, India. Also published as Springer "Lecture Notes in Computer Science", Vol 5922/2009, ISBN:978-3-642-10627-9, pp 308-320.

Kaoutar Elkhiyaoui, Daishi Kato, Kazuo Kunieda, Keiji Yamada, Pietro Michiardi. A Scalable Interest-oriented Peer-to-Peer Pub/Sub Network. P2P'09, 9th International Conference on Peer-to-Peer Computing, September 8-11, 2009, Seattle, Washington, USA, pp 204-211.

1

Introduction

Radio Frequency IDentification (RFID for short) is a part of auto-identification technologies that comprise barcodes, biometrics, smart cards ... etc. An RFID tag is a wireless device that is equipped with a unique and unreusable 96 bit identifier, which contrary to optical barcodes, allows the identification of individual objects without line of sight or human intervention.

At first, RFID technology was envisioned to replace barcodes to automate data collection when handling products traveling in the supply chain. Current applications of RFID technology are not aimed exclusively at supply chains, but for a plethora of other areas that range from biometric passports and pet tracking to access control through car immobilizers.

What makes RFID technology attractive is its relatively low cost. An RFID tag can be sold for about US$0.15 without a volume discount (85). Although currently prohibitive for supply chain applications, the price of an RFID tag is expected to get lower after the standardization of RFID technology to reach commercially viable levels that may accommodate a wide adoption of RFID tags not only in supply chains, but in every other aspect of our life.

Nonetheless, the cost effectiveness of RFID tags comes at a price, which is the privacy of individuals holding RFID tags and the privacy of partners in the supply chain. It is important to note that RFID technology by its design is not privacy friendly, since the original goal of RFID was to enable fast and automated individual object identification and tracking. RFID tags are thus designed to send their identifiers *without the consent of their owners* whenever queried by a compatible RFID reader. This implies that privacy attacks such as tracking of individuals and industrial espionage can be mounted easily by merely querying RFID tags.

To address these privacy concerns two approaches have emerged. The first one relies on physical measures to limit the scope of these attacks. For instance: Faraday cages are now used to manufacture passport cases to prevent un-authorized scanning of RFID-enabled passports. The second approach which is of interest aims at protecting the privacy of RFID tags using *cryptographic* solutions.

Designing cryptographic protocols for RFID turned out to be a very difficult task for two reasons: **first**, it is of utmost importance to keep the cost of RFID technology low to favor

its wide deployment. Therefore, any cryptographic solution for RFID has to fit the limited resources of tags. **Second**, it is crucial to design time-efficient protocols that do not slack off the performances of RFID applications, especially in time-sensitive contexts such as supply chains.

These challenges raised by cryptographic approaches to solve the privacy issues in RFID systems have spurred an active research area, that dealt primarily with the design of *privacy preserving authentication* protocols that suit the computational capabilities of RFID tags. The aim of these protocols is to allow *authorized* RFID readers to authenticate and identify tags, while a non-authorized reader must not be able to learn the identity of a tag by eavesdropping on its communications or querying it. The cryptographic RFID authentication protocols proposed in the literature can be classified into three categories as follows:

- **Lightweight authentication:** Relying on bitwise operations (18, 66, 91), albeit efficient, these protocols were prone to key recovery attacks, see (14, 64, 128).

- **Symmetric authentication:** Protocols in this category use symmetric primitives, see (48, 50, 58, 122, 153). Although efficient on the tag side, Damgård and Pedersen (42) showed that there is a tradeoff between RFID privacy and the scalability of such protocols: to ensure privacy, a symmetric RFID authentication protocol *has to run in linear time* in the number of tags.

- **Public key authentication:** Contrary to symmetric authentication, solutions based on public key techniques (103, 113, 126) offer the possibility to perform *constant* time and *privacy preserving* authentication.

The diversity and the heterogeneity of RFID authentication protocols have stirred interest in formalizing definitions of RFID privacy (5, 92, 129, 159) that aspire to **first** capture the capabilities of a real world adversary against RFID tags, and **second** to measure information leakage through the wireless channel between RFID tags and RFID readers. These formal definitions paved the way for further analysis of existing protocols and for understanding the limitations of RFID privacy in terms of what can actually be achieved in reality.

Unfortunately, it has been shown that most of current RFID authentication protocols fell short of ensuring privacy against an adversary who *tampers with RFID tags and eavesdrops on all of their interactions*. In fact, Vaudenay (159) showed the intuitive result that states that privacy cannot be achieved against such an adversary. While a more positive result shows that in order to ensure privacy against a slightly weaker variant of this adversary, tags have to implement *key agreement protocols*, which mandates the use of public key cryptography in tags (159). Nonetheless, public key cryptography is impracticable for devices that are as constrained as RFID tags. As a result, we conclude that **1.)** cryptographic protocols using RFID tags can at best be built using symmetric primitives, and that **2.)** privacy models have

to be relaxed to bridge the gap between what is desirable and what is actually achievable in terms of tag privacy.

For these reasons, this thesis aims to:

- Formalize *suitable* privacy and security definitions that take into account the stringent constraints akin to RFID tags and the potential actions that an adversary can perform to jeopardize tag privacy. We emphasize that the computational limitations of RFID tags do not favor the implementation of public key primitives.

- Propose *secure and privacy preserving* solutions for supply chain applications that suit the computational limitations of RFID tags and improve collaboration between supply chain partners by reaching beyond the basic tag-reader authentication scenario. In particular, we focus on three applications which are: tag ownership transfer, genuineness verification and enforcing safety regulations in the supply chain. We stress that cryptographic solutions for supply chain applications have to be financially cheap and computationally efficient to assure wide deployment.

Along these lines, we consider in this thesis a relaxed privacy model in which an adversary is assumed to tamper with RFID tags and eavesdrop on their communications, *with the only restriction that he cannot monitor all of their interactions*.

We believe that in the *supply chain setting* the above assumption is *realistic* for two reasons: **1.)** RFID tags *are not tamper-resistant*. This means that any adversary who has access to tags at some point of their lifetime, can easily read and sometimes re-write their contents. **2.)** RFID tags in the supply chain often *change location*. As a matter of fact, RFID tags travel between different partners that usually reside in different countries or even different continents. This makes it difficult for an adversary to *continuously* eavesdrop on tags' communications.

Under this assumption, we are able to **first**, formalize privacy definitions that suit the requirements of RFID-enabled supply chains. **Second**, design cryptographic *multi-party* protocols that transcend the classical two party tag-reader authentication to offer privacy preserving solutions for supply chain applications, some of which can be implemented using *storage only* tags, as will be shown in Part II.

Structure and contributions

The sequel of this thesis is organized as follows:

- In Chapter 2, we provide a comprehensive background on cryptography that on the one hand, reviews the concepts related to provable security and the cryptographic primitives that we will refer to in the rest of this thesis either to help us in the discussion of previous work or in the construction of our cryptographic protocols, and on the other hand,

explains the assumptions underlying elliptic curve cryptography and bilinear pairing based cryptography which allow us to design efficient, provably secure and privacy preserving protocols for the supply chain.

The reader then can either move on to Part I of this thesis which surveys the most prominent work regarding RFID security and privacy, or to Part II which introduces our cryptographic protocols.

- In Chapter 3, we discuss some of the relevant work on RFID security and privacy. The chapter deals with three independent but complementary points. We describe first the privacy and security threats that may be caused by the proliferation of RFID tags. Then, we introduce the existing formalizations of RFID security and privacy while explaining their shortcomings. Finally, we analyze some of the relevant privacy preserving RFID authentication protocols. This summary of related work allows us to point out what we believe to be the limitation of RFID privacy which is: *"adversary models for computationally limited RFID tags assume a strong adversary against which privacy cannot be ensured"*.

- In Chapter 4, we address the problem of efficient and privacy preserving RFID tag ownership transfer in the supply chain. We identify and formalize the security and the privacy requirements of this type of application, and we propose a tag ownership protocol that features:

  - Constant-time authentication while tags are only required to evaluate hash functions.
  - Issuer verification that grants each partner in the supply chain the ability to verify the origin of tags present in his site, in order to prevent the injection of fake products that do not meet quality standards.
  - Provable security and privacy.

- In Chapter 5, we present two protocols that address the issue of product genuineness verification in the supply chain using RFID tags. The first one is *product traceability* by a trusted third party and the second one is *on-site checking* by different supply chain partners. Both protocols rely on the idea of checking product genuineness by verifying the paths that the products went through in the supply chain. The main contributions of this chapter are as follows:

  - Formal definitions that capture the security and the privacy requirements of RFID-based genuineness verification applications.
  - Efficient encoding of paths in the supply chain that does not depend on the number of steps composing the path.

  - Tags are not required to perform any computation. Both protocols target storage only tags and can be implemented using current off-the-shelf RFID tags.
  - Provable security and privacy.

- In Chapter 6, we propose a protocol that aims at enforcing safety regulations in RFID-enabled supply chains. The idea is to allow a reader in the supply chain to verify whether two items can be stored in close proximity or not while these items are labeled with storage only tags. The challenge in such an application scenario is to prevent the reader from getting access to the cleartext content of attributes stored in tags. Like in previous chapters, we first formalize the security and the privacy definitions that meet the requirements of item matching applications in the supply chain. Then, we show that our protocol is secure and privacy preserving while tags are not required to execute *any computation*.

The research work conducted by the author led to a number of scientific publications that overlap with the contributions presented in this thesis, see (14, 19, 53, 54, 55).

2

Cryptography Fundamentals

Our main goal in this thesis is to design provably secure and privacy preserving multi-party protocols for RFID environment. It is therefore natural to provide the reader with a quick overview of the concepts underlying provable security, and to survey the security definitions of the cryptographic primitives that we employ to devise our cryptographic schemes. Also, since most of our protocols take place in elliptic curves that support bilinear pairings, we review the different notions and the mathematical assumptions that laid the basis for elliptic curve cryptography and bilinear pairings.

This chapter is organized as follows: in Section 2.1, we briefly describe two paradigms of provable security which are: game-based security and simulation-based security. The aim of this section is to introduce the notational conventions that will be used in subsequent chapters. In Section 2.2, we present the cryptographic primitives that either will be used to help the exposition of previous work in Part I or to implement our protocols in Part II. Finally, in Section 2.3, we give a background on elliptic curve cryptography and bilinear pairings, namely, the hardness assumptions that ensure the security and the privacy of our RFID protocols.

2.1 Provable Security

For many years, a cryptographic protocol was considered secure as long as it withstood the attacks that the designer of the protocol had envisioned. However, this method of validation had fallen short as adversaries most of the time design their attacks by taking advantage of vulnerabilities in the protocol specification. This has resulted in the development of a more convincing method for security validation which is called *"provable security"*. This approach consists of proving the security of cryptographic schemes in the context of complexity theory. That is, when designing a cryptographic scheme, we do not make assumptions regarding the *strategy* that an adversary may use, but instead we make assumptions with regard to his *computational capabilities*.

Provable security consists of two major activities (69), and these are:

- Definitional activities: The formulation, identification and the definition of security models that capture the security requirements that cryptographic schemes have to fulfill.
- Constructive activities: Design of *efficient* cryptographic schemes that answer to the security definitions.

Note that the approach of provable security is concerned with the design of *efficient* cryptographic schemes for which it is *computationally infeasible* to violate the security. This means that legitimate users can execute the scheme in *polynomial-time* in the security parameter τ (typically, τ is the size in bits of the key used in the cryptographic scheme), while adversaries cannot break the security of the scheme in *polynomial-time*.

Definition 2.1. *A polynomial-time algorithm is an algorithm whose worst-case running time function is $O(p(\tau))$ for some polynomial function p, and where τ is the input length.*

To measure the success of an adversary in breaking a cryptographic scheme, we compute his *"advantage"*. The advantage of an adversary is defined as the difference between the probability that the adversary breaks the scheme and the probability of breaking the scheme by a random guess. A scheme is said to be secure if the advantage of any polynomial-time adversary is a *negligible function* in the security parameter τ.

Definition 2.2. *A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ is a negligible function if for every $c \geq 0$, there exists $N_c \in \mathbb{N}$ such that for all $n > N_c$, $\epsilon(n) \leq \frac{1}{n^c}$.*

When proving the security of a cryptographic scheme, one has to define first a security model against which the scheme is going to be shown secure. Roughly speaking, a security model specifies the security property that a scheme has to satisfy together with the set of actions that the adversary is allowed to take when mounting his attack.

In what follows, we present two paradigms of provable security that were extensively used in the literature to define security models, and these are: game-based security and simulation-based security.

2.1.1 Game-based Security

The security model is defined in terms of an adversarial goal that specifies the security requirements, and an attack model that defines the adversary's capabilities. The security model is then formalized using an interactive security game that is played between a polynomial-time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The challenger $\mathcal{C}$ controls a set of *oracles* that simulate all the computation required by the adversary $\mathcal{A}$ during the security game. In general, a security game consists of two main phases:

- **Learning phase:** Adversary $\mathcal{A}$ is allowed to make a polynomial number of queries to the oracles controlled by $\mathcal{C}$.
- **Challenge phase:** Adversary $\mathcal{A}$ is asked to perform a particular action determined by an adversarial goal that is specified beforehand. The adversary is said to win the game, if he achieves his adversarial goal.

Proving that some cryptographic scheme is secure is done by showing that if there is an adversary $\mathcal{A}$ who wins the security game, then this adversary $\mathcal{A}$ can be transformed in polynomial-time into an adversary $\mathcal{B}$ that solves some known hard problem. The transformation is performed by simulating the attack environment of adversary $\mathcal{A}$ using the input of the hard problem to be solved, with the restriction that it should be computationally infeasible for adversary $\mathcal{A}$ to distinguish between the simulated environment and the real world environment.

2.1.2 Simulation-based Security

Simulation-based security (69, 70) deals with formulating the intuitive requirement that an adversary $\mathcal{A}$ must *"gain nothing"* when he is maliciously executing some cryptographic scheme. This paradigm states that an adversary "gains nothing" if whatever he learns by deviating from the prescribed honest behavior can also be learned in an *"ideal model"* (69), in which the cryptographic scheme is replaced with an ideal scheme. The ideal model in this paradigm captures the security requirements that the cryptographic scheme has to fulfill. Now, to prove that a scheme is secure with respect to the simulation-based security paradigm, one shows that there exists a polynomial transformation of any adversary $\mathcal{A}$ against the scheme in the real model into an adversary $\mathcal{B}$ against the ideal scheme.

2.2 Cryptographic Primitives

we describe herein the cryptographic primitives – and their related security definitions – that we will refer to in this thesis either to review previous work or to build our cryptographic protocols.

2.2.1 Cryptographic Hash Functions

A cryptographic hash function is a deterministic algorithm that maps a variable-length input string called preimage into a fixed length output string called hash, such that any slight change to the input results in a different output. Thus, if two input strings have the same hash, then this implies that they are identical with an overwhelming probability. A property holds with an overwhelming probability, if it holds with a probability larger than $1 - \epsilon(\tau)$, where ϵ is a negligible function and τ is the security parameter.

Definition 2.3. *A cryptographic hash function* $H : \{0,1\}^* \rightarrow \{0,1\}^n$ *is an efficiently computable function that satisfies the following properties:*

- **Preimage resistance:** *For all* $y \in \{0,1\}^n$*, it is computationally infeasible to find an element* $x \in \{0,1\}^*$ *such that* $y = H(x)$*.*
- 2^{nd} **preimage resistance:** *For all* $x \in \{0,1\}^*$*, it is computationally infeasible to find* $x' \neq x$ *such that* $H(x) = H(x')$*.*
- **Collision resistance:** *It is computationally infeasible to find* $x \neq x' \in \{0,1\}^*$ *such that* $H(x) = H(x')$*.*

For a more comprehensive security definitions of cryptographic hash functions, we refer to the work of Rogaway and Shrimpton (137).

A cryptographic hash function can model a random function. This property have paved the way for the random oracle model that was shown to be very practical when validating cryptographic protocols.

2.2.1.1 Hash Functions and The Random Oracle Model

A popular approach to design secure protocols is the random oracle model. This model was proposed by Bellare and Rogaway (10) to bridge the gap between inefficient provable security and efficient practical security. The idea of the random oracle model is to first prove the security of protocols in an ideal setting in which all the parties including adversaries can make oracle queries to a truly random function (*ideal hash function*) $\mathcal{R} : \{0,1\}^\infty \rightarrow \{0,1\}^\infty$. Then, replace the random oracle with a cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$.

A proof of security in the random oracle model assures that the the overall design of a given protocol is sound. However, a secure implementation of that protocol relies on the security of the cryptographic hash function that will be used to replace the random oracle.

Although the random oracle model has been proven to be practical in the design of heuristically secure protocols, Canetti et al. (34) showed that it is possible to construct *unnatural* protocols that are secure in the random oracle model, but have no secure implementation in the real world. Yet, Canetti et al. (34) noted that the random oracle model is still a useful tool for designing and analyzing protocols, and can be regarded as a first step towards devising more efficient and secure ones.

2.2.2 Pseudo-random Generators

A pseudorandom generator (PRG) is a deterministic algorithm that maps a seed to a longer pseudorandom string such that no polynomial-time algorithm can distinguish the output of the pseudo-random generator and the output of the uniform distribution.

Definition 2.4. *A pseudo-random generator* $G : \{0,1\}^k \rightarrow \{0,1\}^n$, *where* $n \geq k$, *is a deterministic algorithm which on input of a random* k*-bit seed outputs a* n*-bit string which is* **computationally indistinguishable** *from uniformly chosen* n*-bit string.*

Here k *is called the seed length of generator* G *and* $n - k$ *is called the stretch of* G.

Now we give the formal definition of *computational indistinguishability* of two random variables.

Definition 2.5. *Let* $U = \{U_n\}_{n\in\mathbb{N}}$ *and* $V = \{V_n\}_{n\in\mathbb{N}}$ *be two sequences of random variables such that each* U_n *and* V_n *ranges over strings of length* n. U *and* V *are said to be* **computationally indistinguishable** *if for every (probabilistic) polynomial-time algorithm* $\mathcal{A}$ *the difference:*

$$\delta_{\mathcal{A}}(n) = |Pr(\mathcal{A}(U_n) = 1) - Pr(\mathcal{A}(V_n) = 1)|$$

is a negligible function in n.

2.2.3 Pseudo-random Function Family

A pseudo-random function family, abbreviated PRF, is a collection of efficiently-computable functions such that it is computationally infeasible to distinguish a function selected at random from the PRF family and a truly random function.

Goldreich et al. (71) proposed a security game to validate the security of pseudo-random function family. We denote this security game PRF-D.

Definition 2.6. *Let* $\mathcal{F} = \{f_K : \mathcal{D} \rightarrow \mathcal{R} \mid K \in \mathcal{K}\}$ *be a function family. Here* $\mathcal{D}$ *is the domain of* $\mathcal{F}$, $\mathcal{R}$ *is the range of* $\mathcal{F}$, *and* $\mathcal{K}$ *is the set of keys, and let* $\mathcal{A}(r_f, \epsilon)$ *be an adversary against the family function* $\mathcal{F}$.

The PRF-D game consists of three phases:

- **Learning:** *Adversary* $\mathcal{A}$ *calls the oracle* $\mathcal{O}_{f_K}$ *(controlled by challenger* $\mathcal{C}$*) for a polynomial number of queries* r_f *with messages* $\{m_1, m_2, ..., m_{r_f}\}$. *When queried with a message* $m_i \in \mathcal{D}$, $\mathcal{O}_{f_K}$ *returns* $y = f_K(m_i) \in \mathcal{R}$.

- **Challenge:** *Adversary* $\mathcal{A}$ *outputs a challenge message* $m_c \notin \{m_1, m_2, ..., m_{r_f}\}$. *Challenger* $\mathcal{C}$ *flips a fair coin* $b \in \{0,1\}$. *If* $b = 1$, *then challenger* $\mathcal{C}$ *returns* $y_c = f_K(m_c)$; *otherwise, he picks randomly* y_c *from the range* $\mathcal{R}$.

- **Guess:** *Adversary* $\mathcal{A}$ *outputs his guess* b' *for bit* b.

Adversary $\mathcal{A}$ *is said to win the game if* $b' = b$.
The advantage ϵ *of adversary* $\mathcal{A}$ *in winning the PRF-D game is defined as:*

$$\epsilon = Pr(\mathcal{A} \text{ wins}) - \frac{1}{2}$$

Definition 2.7. *Let $\mathcal{F} = \{f_K : \mathcal{D} \to \mathcal{R} \mid K \in \mathcal{K}\}$ be a function family. $\mathcal{F}$ is called a family of pseudo random functions (PRF for short) if:*

- *$\forall K \in \mathcal{K}, f_K$ is computable in polynomial-time.*
- *$\mathcal{F}$ is pseudorandom: no adversary $\mathcal{A}$ can distinguish a function f_K in $\mathcal{F}$ from a function f drawn at random from the set of all possible functions $\mathcal{F} : \mathcal{D} \to \mathcal{R}$. That is, for any adversary $\mathcal{A}(r_f, \epsilon)$, the advantage ϵ in winning the PRF-D game is negligible.*

For more details on how to construct pseudo-random function family from pseudorandom generators, we refer to the work of Goldreich et al. (71).

2.2.4 Message Authentication Codes

A message authentication code (MAC for short) is a cryptographic primitive that allows any party to compute a keyed hash σ of a message m using a secret key K, while any party possessing the secret key K can verify that σ is a valid MAC of m.

Definition 2.8. *A Message authentication code* MAC *consists of four algorithms:* Setup, KeyGen, MAC *and* Verify.

- Setup*: On input of a security parameter τ, this algorithm outputs a set $\mathcal{P}$ of public parameters that will be used by following algorithms, together with a key space $\mathcal{K}$, a message space $\mathcal{M}$ and a MAC space $\mathcal{S}$.*
- KeyGen*: On input of the public parameters $\mathcal{P}$ and the key space $\mathcal{K}$, this algorithm outputs a random key $K \in \mathcal{K}$. K is the MAC's secret key*
- MAC*: On input of a message $m \in \mathcal{M}$ and secret key K, this algorithm outputs $\sigma = \mathsf{MAC}_K(m) \in \mathcal{S}$.*
- Verify*: On input of a message m, a MAC σ and secret key K, this algorithm outputs a bit $b = \mathsf{Verify}_K(m, \sigma)$. $b = 1$, if $\sigma = \mathsf{MAC}_K(m)$; otherwise $b = 0$.*

A message authentication code scheme has to satisfy the following:

$$\sigma = \mathsf{MAC}_K(m) \Leftrightarrow \mathsf{Verify}_K(m, \sigma) = 1$$

A message authentication code has to ensure sender *authenticity* and message *integrity*. Particularly, it must be computationally infeasible for an adversary $\mathcal{A}$ who does not possess the secret key K to forge a valid MAC. The security of a message authentication code is usually measured by the inability of an adversary $\mathcal{A}$ to forge a new valid MAC of a message m of his choice under chosen plaintext attack. This is called resistance to existential forgery.

The resistance to existential forgery of message authentication codes under chosen plaintext attack is defined by an interactive game MAC-REF between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ that we are going to present next.

Definition 2.9. *Let* $\mathsf{MAC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *be a message authentication code, and let* $\mathcal{A}(r_s, \epsilon)$ *be an adversary against the resistance of existential forgery of* MAC*.*

The MAC-REF game consists of two phases:

- **Learning:** *Adversary* $\mathcal{A}$ *performs a polynomial number of queries* r_s *to a MAC oracle* $\mathcal{O}_{\mathsf{MAC}}$ *which is controlled by the challenger* $\mathcal{C}$*. When queried with a message* m*,* $\mathcal{O}_{\mathsf{MAC}}$ *returns* $\sigma = \mathsf{MAC}_K(m)$*.*
- **Challenge:** *Adversary* $\mathcal{A}$ *outputs a challenge message* m_c *and a MAC* σ_c*.*

Adversary $\mathcal{A}$ *is said to win the MAC-REF game if* $\mathsf{Verify}_K(m_c, \sigma_c) = 1$*, and if he did not query the oracle* $\mathcal{O}_{\mathsf{MAC}}$ *with message* m_c*.*

The advantage ϵ *of adversary* $\mathcal{A}$ *in winning the MAC-REF game is defined as:*

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 2.10. *A message authentication code* $\mathsf{MAC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{MAC}, \mathsf{Verify})$ *is said to be resistant to existential forgery,* **iff** *for any adversary* $\mathcal{A}(r_s, \epsilon)$*, the advantage* ϵ *in winning the MAC-REF game is negligible.*

2.2.5 Encryption

An encryption scheme consists of four efficient algorithms: *Setup* Setup, *key generation* KeyGen, *encryption* Enc and *decryption* Dec.

Definition 2.11. *An encryption* ENC *scheme is determined by four algorithms:*

- Setup*: On input of a security parameter* τ*, this algorithm outputs a set* $\mathcal{P}$ *of public parameters that will be used by following algorithms, together with a key space* $\mathcal{K}$*, a message space* $\mathcal{M}$ *and a ciphertext space* $\mathcal{C}$*.*
- KeyGen*: On input of the public parameters* $\mathcal{P}$ *and the key space* $\mathcal{K}$*, this algorithm outputs a pair of random keys* $(K_e, K_d) \in \mathcal{K}$*, where* K_e *is the encryption key and* K_d *is the corresponding decryption key.*
- Enc*: On input of a message* $m \in \mathcal{M}$ *and the encryption key* K_e*, this algorithm outputs a ciphertext* $c \in \mathcal{C}$*.*
- Dec*: On input of a ciphertext* $c \in \mathcal{C}$ *and the decryption key* K_d*, this algorithm outputs a message* $m \in \mathcal{M}$ *if the decryption succeeds; otherwise it outputs* $\perp$*.*

An encryption has to satisfy the following:

$$c = \mathsf{Enc}_{K_e}(m) \Leftrightarrow m = \mathsf{Dec}_{K_d}(c)$$

Definition 2.12. *A symmetric-key encryption scheme* $\mathsf{ENC}_{\mathsf{sym}}$ *is an encryption scheme where* $K_e = K_d$.

Definition 2.13. *A public-key encryption scheme* $\mathsf{ENC}_{\mathsf{pub}}$ *is an encryption scheme where* $K_e \neq K_d$. K_e *is called public key and usually denoted* pk *and* K_d *is called secret key and usually denoted* sk.

Note that public key encryption schemes enable any party A to send encrypted messages to another party B that only B can decrypt, without any prior agreement. Contrary to symmetric key encryption schemes where the parties A and B have to agree beforehand on an encryption key.

Next, we review the definitions of secure encryption that will be referenced in the remainder of this manuscript.

As proposed by Bellare et al. (11), we organize definitions by considering first the adversarial goal and then the attack model. As a result, security definitions are obtained as *"a pairing of a particular adversarial goal and a particular attack model"* (11).

Given an encryption scheme ENC and a challenge ciphertext c encrypted using the encryption key K_e, we consider two adversarial goals:

- *One-wayness OW*: The goal of an adversary $\mathcal{A}$ is to decrypt c without having access to the decryption key K_d.

- *Indistinguishability IND*: The goal of an adversary $\mathcal{A}$ is to tell whether a challenge ciphertext c encrypts a message m_0 or whether it encrypts a message m_1 with a probability significantly larger than one half, where m_0 and m_1 are two messages in $\mathcal{M}$ that were chosen by $\mathcal{A}$. Indistinguishability formalizes the inability of adversary $\mathcal{A}$ to learn any information about the plaintext m underlying the ciphertext c.

In addition to the adversarial goals, we consider two attack models depending on the information provided to the adversary $\mathcal{A}$. In order of increasing strength, these are: *chosen plaintext attack* and *chosen ciphertext attack*.

- *Chosen plaintext attack CPA*: An adversary $\mathcal{A}$ can encrypt any message of his choice. To this effect, $\mathcal{A}$ has access to an encryption oracle $\mathcal{O}_{\mathsf{Enc}}$, that when given a plaintext m and an encryption key K_e returns $c = \mathsf{Enc}_{K_e}(m)$.

- *Chosen ciphertext attack CCA*: Besides being able to query the encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ with messages of his choice, adversary $\mathcal{A}$ has access to a decryption oracle $\mathcal{O}_{\mathsf{Dec}}$, that when given a ciphertext c and a decryption key K_d returns $m = \mathsf{Dec}_{K_d}(c)$. Adversary $\mathcal{A}$ is allowed to query $\mathcal{O}_{\mathsf{Dec}}$ with ciphertexts of his choice except for the challenge ciphertext c.

If adversary $\mathcal{A}$ uses the decryption oracle only before obtaining the challenge ciphertext c, then the attack model is called non-adaptive chosen ciphertext attack (CCA1). Otherwise, the attack model is called adaptive chosen ciphertext attack (CCA2).

Consequently, we obtain six security models: OW-CPA, OW-CCA1, OW-CCA2, IND-CPA, IND-CCA1 and IND-CCA2. These security models are defined using interactive games in accordance with the work of Bellare et al. (11):

Definition 2.14. *Let* $\mathsf{ENC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme, and let* $\mathcal{A}(r_e, r_d, s_e, s_d, \epsilon)$ *be an adversary against the one wayness of* ENC.

The OW-ATK $\in \{$*OW-CPA, OW-CCA1, OW-CCA2*$\}$ *game consists of four phases:*

- **Learning-1:** *Adversary* $\mathcal{A}$ *makes a polynomial number of queries* r_e *to the encryption oracle* $\mathcal{O}_{\mathsf{Enc}}$ *and* r_d *queries to the decryption oracle* $\mathcal{O}_{\mathsf{Dec}}$.
- **Challenge:** *Challenger* $\mathcal{C}$ *picks at random a message* $m \in \mathcal{M}$ *and returns the challenge ciphertext* $c = \mathsf{Enc}_{K_e}(m)$ *to adversary* $\mathcal{A}$.
- **Learning-2:** *Adversary* $\mathcal{A}$ *makes a polynomial number of queries* s_e *to the encryption oracle* $\mathcal{O}_{\mathsf{Enc}}$ *and* s_d *queries to the decryption oracle* $\mathcal{O}_{\mathsf{Dec}}$*, with the restriction that he cannot query the decryption oracle* $\mathcal{O}_{\mathsf{Dec}}$ *with the challenge ciphertext* c.
- **Guess:** *Adversary* $\mathcal{A}$ *outputs a guess* m'.

Adversary $\mathcal{A}$ *is said to win the OW-ATK game if* $m = m'$*, where*

OW-ATK = OW-CPA, if $r_d = s_d = 0$.

OW-ATK = OW-CCA1, if $r_d \neq 0$ *and* $s_d = 0$.

OW-ATK = OW-CCA2, if $s_d \neq 0$.

The advantage ϵ *of adversary* $\mathcal{A}$ *in winning the OW-ATK game is defined as:*

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 2.15. *An encryption* $\mathsf{ENC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be OW-ATK secure,* **iff** *for any adversary* $\mathcal{A}(r_e, s_e, r_d, s_d, \epsilon)$*, the advantage* ϵ *in winning the OW-ATK game is negligible.*

Definition 2.16. *Let* $\mathsf{ENC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme, and let* $\mathcal{A}$ $(r_e, r_d, s_e, s_d, \epsilon)$ *be an adversary against the indistinguishability of* ENC.

The IND-ATK $\in \{$*IND-CPA, IND-CCA1, IND-CCA2*$\}$ *game consists of four phases:*

- **Learning-1:** *Adversary* $\mathcal{A}$ *makes a polynomial number of queries* r_e *to the encryption oracle* $\mathcal{O}_{\mathsf{Enc}}$ *and* r_d *queries to the decryption oracle* $\mathcal{O}_{\mathsf{Dec}}$.
- **Challenge:** *Adversary* $\mathcal{A}$ *provides challenger* $\mathcal{C}$ *with two messages* m_0 *and* m_1 *in* $\mathcal{M}$*. Challenger* $\mathcal{C}$ *flips a fair coin* $b \in \{0, 1\}$*, then returns the challenge ciphertext* $c_b = \mathsf{Enc}_{K_e}(m_b)$ *to adversary* $\mathcal{A}$.

- **Learning-2:** *Adversary $\mathcal{A}$ makes a polynomial number of queries s_e to the encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ and s_d queries to the decryption oracle $\mathcal{O}_{\mathsf{Dec}}$, with the restriction that he cannot query the decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ with the challenge ciphertext c_b.*

- **Guess:** *Adversary $\mathcal{A}$ outputs a guess b'.*

Adversary $\mathcal{A}$ is said to win the IND-ATK game if $b = b'$, where
IND-ATK = IND-CPA, if $r_d = s_d = 0$.
IND-ATK = IND-CCA1, if $r_d \neq 0$ and $s_d = 0$.
IND-ATK = IND-CCA2, if $s_d \neq 0$.
The advantage ϵ of adversary $\mathcal{A}$ in winning the IND-ATK game is defined as:

$$\epsilon = Pr(\mathcal{A}\ wins) - \frac{1}{2}$$

Definition 2.17. *An encryption* $\mathsf{ENC} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be IND-ATK secure,* **iff** *for any adversary $\mathcal{A}(r_e, s_e, r_d, s_d, \epsilon)$, the advantage ϵ in winning the IND-ATK game is negligible.*

$$\begin{array}{ccccc} \text{IND-CPA} & \Leftarrow & \text{IND-CCA1} & \Leftarrow & \text{IND-CCA2} \\ \Downarrow & & \Downarrow & & \Downarrow \\ \text{OW-CPA} & \Leftarrow & \text{OW-CCA1} & \Leftarrow & \text{OW-CCA2} \end{array}$$

Figure 2.1: Relations between security notions for encryption schemes (11)

2.2.6 Digital Signatures

A signature scheme is the alternative of MAC in the public key setting. A party can generate a signature $\mathcal{S}$ on a message m using its secret key sk, while anyone can verify the validity of the signature by using the public key pk corresponding to the secret key sk.

Definition 2.18. *A digital signature scheme denoted* DS*, is determined by four algorithms:*

- Setup*: On input of a security parameter τ, this algorithm outputs a set $\mathcal{P}$ of public parameters that will be used by following algorithms, together with a key space $\mathcal{K}$, a message space $\mathcal{M}$ and a signature space $\mathcal{S}$.*

- KeyGen*: On input of the public parameters $\mathcal{P}$ and the key space $\mathcal{K}$, this algorithm outputs a pair of random keys $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{K}$, where sk is the secret key and pk is the corresponding public key.*

- Sign*: On input of a message $m \in \mathcal{M}$ and secret key sk, this algorithm outputs $\mathcal{S} = \mathsf{Sign}_{\mathsf{sk}}(m) \in \mathcal{S}$.*

- Verify: *On input of a message* m*, a signature* $\mathcal{S}$ *and public key* pk*, this algorithm outputs a bit* $b = \mathsf{Verify}_{\mathsf{pk}}(m, \mathcal{S})$*.* $b = 1$*, if the signature is valid; otherwise* $b = 0$*.*

A digital signature scheme has to satisfy the following:

$$\mathcal{S} = \mathsf{Sign}_{\mathsf{sk}}(m) \Leftrightarrow \mathsf{Verify}_{\mathsf{pk}}(m, \mathcal{S}) = 1$$

Digital signatures have to ensure the *authenticity* and the *integrity* of the message signed. For this, it must be computationally infeasible for an adversary $\mathcal{A}$ who does not have access to the secret key sk to forge a valid pair $(m, \mathcal{S} = \mathsf{Sign}_{\mathsf{sk}}(m))$. Contrary to message authentication codes, digital signatures have to ensure as well the *non-repudiation* of signer. That is, it must be computationally infeasible for a signer to claim that a signature verifiable by his public key is forged.

Similar to message authentication codes, the security of digital signatures is measured by using an interactive game DS-REF that captures the capabilities of an adversary $\mathcal{A}$ against resistance to existential forgery under chosen plaintext attack.

Definition 2.19. *Let* $\mathsf{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a digital signature scheme, and let* $\mathcal{A}(r_s, \epsilon)$ *be an adversary against the resistance to existential forgery of* DS*.*

The DS-REF game consists of two phases:

- **Learning:** *Adversary* $\mathcal{A}$ *makes a polynomial number of queries* r_s *to a signing oracle* $\mathcal{O}_{\mathsf{sign}}$ *which is controlled by the challenger* $\mathcal{C}$*. When queried with a message* m*,* $\mathcal{O}_{\mathsf{sign}}$ *returns* $\mathcal{S} = \mathsf{Sign}_{\mathsf{sk}}(m)$*.*
- **Challenge:** *Adversary* $\mathcal{A}$ *outputs a challenge message* m_c *and a signature* $\mathcal{S}_c$*.*

Adversary $\mathcal{A}$ *is said to win the DS-REF game if* $\mathsf{Verify}_{\mathsf{pk}}(m_c, \mathcal{S}) = 1$ *and if he did not query the oracle* $\mathcal{O}_{\mathsf{sign}}$ *with message* m_c*.*

The advantage ϵ *of adversary* $\mathcal{A}$ *in winning the DS-REF game is defined as*

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 2.20. *A digital signature scheme* $\mathsf{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is said to be resistant to existential forgery,* **iff** *for any adversary* $\mathcal{A}(r_s, \epsilon)$*, the advantage* ϵ *in winning the DS-REF game is negligible.*

We refer the reader to the work of Goldwasser et al. (72) for a more detailed discussion on the security notions of digital signatures.

2.3 Elliptic Curve Cryptography

In 1985, Neal Koblitz and Victor Miller suggested independently the use of elliptic curves to devise public key schemes, and since then, elliptic curve cryptography (abbreviated ECC)

has emerged as a viable alternative to cryptography in finite fields. The main advantage of elliptic curve based schemes over the other public key schemes is their short key size, which results in more efficient and faster schemes. For example, the typical key size for EC schemes that provide the same level of security as 1024-bits public key schemes in finite fields is 160 bits, cf. (78, 124). In fact, ECC has short keys because the *index calculus* algorithm cannot be executed in elliptic curves to solve the discrete logarithm problem, while it can be used successfully in finite fields.

For more details on elliptic curve cryptography, we refer to (15, 16, 78, 162).

2.3.1 Elliptic curves

Definition 2.21. *An elliptic curve $\mathcal{E}(\mathbb{K})$ over a field $\mathbb{K}$ consists of a special point $\mathbb{1}_{\mathcal{E}}$ called* **point at infinity** *and a set of points $g = (x, y) \in \mathbb{K}^2$ that satisfy the* **Weierstrass equation***:*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.1}$$

where $a_i \in \mathbb{K}$ for $i = 1, 2, 3, 4, 6$.

An elliptic curve has to be **nonsingular***, i.e., the polynomial $P(x) = x^3 + a_2x^2 + a_4x + a_6$ must have single roots.*

Remark 2.1. *Equation 2.1 is useful when the characteristic of $\mathbb{K}$ $\mathsf{char}(\mathbb{K}) \in \{2, 3\}$. However, when $\mathsf{char}(\mathbb{K}) \notin \{2, 3\}$, equation 2.1 can be simplified by applying the following transformation:*

$$\begin{aligned} x_1 &\leftarrow x + \frac{4a_2 + a_1^2}{12} \\ y_1 &\leftarrow y + \frac{a_1x + a_3}{2} \end{aligned}$$

Hereby, we obtain:

$$y_1^2 = x_1^3 + Ax_1 + B$$

where $A, B \in \mathbb{K}$.

For the sake of simplicity, in the rest of this section we assume that $\mathsf{char}(\mathbb{K}) \neq 2, 3$.

Remark 2.2. *Let r_1, r_2 and r_3 denote the roots of polynomial $P(x)$.*

The discriminant of $P(x)$ is defined as:

$$\begin{aligned} \Delta &= (r_1 - r_2)^2(r_1 - r_3)^2(r_2 - r_3)^2 \\ &= -(4A^3 + 27B^2) \end{aligned}$$

Consequently, to check whether an elliptic curve $\mathcal{E}(\mathbb{K})$ over field $\mathbb{K}$ is nonsingular, it suffices to compute Δ and to check whether $\Delta \neq \mathbb{0}_{\mathbb{K}}$.

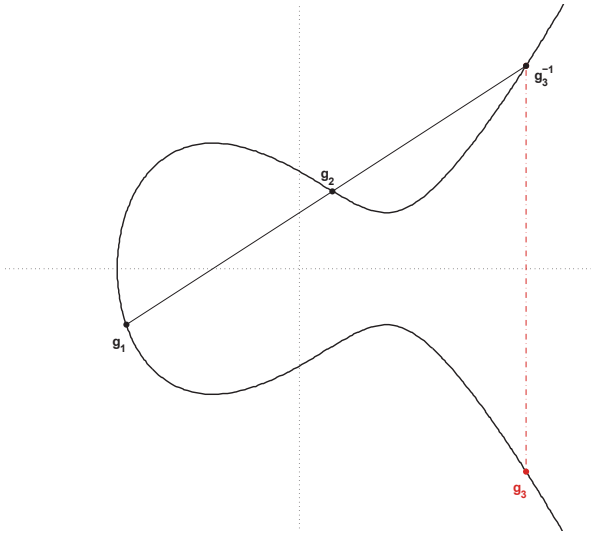

Figure 2.2: Adding points on an elliptic curve

The Group Law

Let $\mathcal{E}(\mathbb{K})$ be an elliptic curve over the field $\mathbb{K}$ defined by $y^2 = x^3 + Ax + B$, $A, B \in \mathbb{K}$.

Let $g_1 = (x_1, y_1)$ and $g_2 = (x_2, y_2)$ be points on $\mathcal{E}$ with $g_1, g_2 \neq \mathbb{1}_\mathcal{E}$. We define $g_3 = g_1 \times g_2 = g_1 g_2 = (x_3, y_3)$ as follows:

1. If $x_1 \neq x_2$, then

$$x_3 = s^2 - x_1 - x_2,\ y_3 = s(x_1 - x_3) - y_1, \text{ where } s = \frac{y_2 - y_1}{x_2 - x_1}$$

2. If $x_1 = x_2$ and $y_1 = -y_2$, then $g_3 = \mathbb{1}_\mathcal{E}$

3. If $x_1 = x_2$ and $y_1 = y_2 \neq \mathbb{0}_\mathbb{K}$, then

$$x_3 = s^2 - 2x_1,\ y_3 = s(x_1 - x_3) - y_1, \text{ where } s = \frac{3x_1^2 + A}{2y_1}$$

Moreover,

$$\forall\ g \in \mathcal{E}(\mathbb{K}),\ g \times \mathbb{1}_\mathcal{E} = \mathbb{1}_\mathcal{E} \times g = g$$

Theorem 2.1. *The points on an elliptic curve $\mathcal{E}(\mathbb{K})$ form an* **abelian group** *with respect to the $\times$ operation defined above, where the identity element is the* **point at infinity** *$\mathbb{1}_\mathcal{E}$, and the inverse of a point $g = (x, y)$ on $\mathcal{E}(\mathbb{K})$ is defined as $g^{-1} = (x, -y)$.*

Definition 2.22. *For all $g \in \mathcal{E}(\mathbb{K})$ and $k \in \mathbb{Z}$, point multiplication of g by k (denoted g^k) is defined as:*

1. *If $k \geq 1$, then $g^k = \underbrace{g \times g \times ... \times g}_{k \text{ times}}$;*

2. *if* $k = 0$, *then* $g^0 = \mathbb{1}_{\mathcal{E}}$;

3. *if* $k \leq -1$, *then* $g^k = (g^{-1})^{-k}$.

Definition 2.23. *A point* $g \in \mathcal{E}(\mathbb{K})$ *is called a torsion point,* **iff** g *is a point of finite order. More precisely,* g *is said to be a* q*-torsion point* ($q \in \mathbb{N}$), **iff** $g^q = \mathbb{1}_{\mathcal{E}}$.

2.3.2 Elliptic Curves over Finite Fields

Let $\mathbb{F}_p$ be a finite field of order p and let $\mathcal{E}(\mathbb{F}_p)$ be an elliptic curve over $\mathbb{F}_p$. Given that there is finitely many pairs $(x, y) \in \mathbb{F}_p^2$, it follows that the abelian group $\mathcal{E}(\mathbb{F}_p)$ is also finite.

Theorem 2.2 (Hasse (162))**.** *Let* $\mathcal{E}(\mathbb{F}_p)$ *be an elliptic curve over the finite field* $\mathbb{F}_p$*. The order* $\#\mathcal{E}(\mathbb{F}_p)$ *satisfies the following inequality:*

$$|p + 1 - \#\mathcal{E}(\mathbb{F}_p)| \leq 2\sqrt{p}$$

Remark 2.3. *Let* $\mathcal{E}(\mathbb{F}_p)$ *be an elliptic curve defined over a finite field* $\mathbb{F}_p$*, then any point* $g \in \mathcal{E}(\mathbb{F}_p)$ *is a torsion point of some order* q *that divides* $\#\mathcal{E}(\mathbb{F}_p)$.

The finite order of elliptic curves over *finite fields* was the starting point for elliptic curve cryptography, which relies on a set of mathematical problems that are believed to be hard in elliptic curves over finite fields. Namely, the *discrete logarithm problem* and the *Diffie-Hellman problems*.

2.3.2.1 Elliptic Curve Discrete Logarithm Problem

Definition 2.24 (Elliptic Curve Discrete Logarithm Problem (DLP))**.** *Let* $\mathcal{E}(\mathbb{F}_p)$ *be an elliptic curve over a finite field* $\mathbb{F}_p$*, the elliptic curve discrete logarithm problem is:*

Given a q*-torsion point* $g \in \mathcal{E}(\mathbb{F}_p)$ *and* $\tilde{g} \in \langle g \rangle$*, find the integer* $x \in \mathbb{Z}_q$ *such that* $\tilde{g} = g^x$*. The integer* x *is called the discrete logarithm of* $\tilde{g}$ *to the base* g*, denoted* $x = \mathsf{log}_g(\tilde{g})$.

The advantage ϵ *of an algorithm* $\mathcal{A}$ *in solving the DL problem is defined as:*

$$\epsilon = Pr(\mathcal{A}(\mathcal{E}, g, \tilde{g}) \text{ computes } x)$$

Definition 2.25 (Elliptic Curve Discrete Logarithm Assumption (DL))**.** *We say that the discrete logarithm assumption holds in* $\mathcal{E}(\mathbb{F}_p)$*, if for every probabilistic polynomial-time algorithm* $\mathcal{A}$*, the advantage* ϵ *in solving DLP in* $\mathcal{E}(\mathbb{F}_p)$ *is negligible.*

Note that the elliptic curve parameters for cryptographic schemes should be carefully chosen so as to resist known attacks on the DL problem. The best known algorithm to solve DLP is a combination of the Pohlig-Hellman algorithm and the Pollard's rho algorithm, which runs in $O(\sqrt{q_i})$ where q_i is the largest divisor of q. In order to withstand this attack, the elliptic curve $\mathcal{E}(\mathbb{F}_p)$ and the point g should be chosen so that the order q of point g is divisible by a sufficiently large prime number q_i. Typically $|q_i| = 160$ bits.

2.3.2.2 Elliptic Curve Diffie-Hellman Problems

The Diffie-Hellman problems (DHP) are mathematical problems that were first introduced in the seminal work of Diffie and Hellman (49) to solve the issue of secure key exchange over public (insecure) channels. It is noteworthy that the Diffie-Hellman problems, like the discrete logarithm problem, were proposed initially in the context of finite fields, however in this manuscript, we only focus on their elliptic curve variants.

Definition 2.26 (Elliptic Curve Computational Diffie-Hellman Problem (CDHP))**.** *Let $\mathbb{G}$ be a cyclic subgroup of order q in $\mathcal{E}(\mathbb{F}_p)$, and g be a generator of $\mathbb{G}$, the computational Diffie-Hellman problem is:*

Given g, g^x, g^y in $\mathbb{G}$ for randomly chosen $x, y \in \mathbb{Z}_q$, compute g^{xy}.

The advantage ϵ of an algorithm $\mathcal{A}$ in solving the CDH problem is defined as:

$$\epsilon = Pr(\mathcal{A}(\mathbb{G}, g, g^x, g^y) \text{ computes } g^{xy})$$

Definition 2.27 (Elliptic Curve Computational Diffie-Hellman Assumption (CDH))**.** *We say that the computational Diffie-Hellman assumption holds in $\mathbb{G}$, if for every probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage ϵ in solving CDHP in $\mathbb{G}$ is negligible.*

Remark 2.4. *If there is a polynomial-time algorithm that can solve DLP in $\mathcal{E}(\mathbb{F}_p)$, then this algorithm can use g and g^x to compute x. Then, it can compute $g^{xy} = (g^y)^x$ to solve CDHP in $\mathcal{E}(\mathbb{F}_p)$.*

Definition 2.28 (Elliptic Curve Decisional Diffie-Hellman Problem (DDHP))**.** *Let $\mathbb{G}$ be a cyclic subgroup of order q in $\mathcal{E}(\mathbb{F}_p)$, and g be a generator of $\mathbb{G}$, the decisional Diffie-Hellman problem is:*

Given g, g^x, g^y, g^z in $\mathbb{G}$, decide whether $z = xy$.

Let U be the distribution $(\mathbb{G}, g, g^x, g^y, g^{xy})$, and V be the distribution $(\mathbb{G}, g, g^x, g^y, g^z)$, where x, y, z are randomly selected in $\mathbb{Z}_q$.

The advantage ϵ of an algorithm $\mathcal{A}$ in solving the DDH problem is defined as:

$$\epsilon = |Pr(\mathcal{A}(U) = 1) - Pr(\mathcal{A}(V) = 1)|$$

Definition 2.29 (Elliptic Curve Decisional Diffie-Hellman Assumption (DDH))**.** *We say that the decisional Diffie-Hellman assumption holds in $\mathbb{G}$, if for every probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage ϵ in solving DDHP in $\mathbb{G}$ is negligible. This means that it is computationally infeasible for any polynomial-time algorithm $\mathcal{A}$ to distinguish between the distribution $U = (\mathbb{G}, g, g^x, g^y, g^{xy})$ and the distribution $V = (\mathbb{G}, g, g^x, g^y, g^z)$ for randomly selected $x, y, z \in \mathbb{Z}_q$.*

Note that if there is a polynomial-time algorithm $\mathcal{A}$ that solves CDHP in $\mathcal{E}(\mathbb{F}_p)$, then this algorithm can be used to solve DDHP in $\mathcal{E}(\mathbb{F}_p)$. Using g, g^x and g^y, $\mathcal{A}$ computes g^{xy}

and checks whether $g^{xy} = g^z$. Nonetheless, the reverse is not true. Joux and Nguyen (87) showed that there exist cyclic subgroups of elliptic curves over finite fields where DDHP is easy and CDHP is hard. These subgroups are known as the *gap Diffie-Hellman* (GDH for short) groups.

The algorithm proposed in (87) solves DDHP by using symmetric *bilinear pairings*. A symmetric bilinear pairing e is a *bilinear* function that maps a pair of points $(g, h) \in \mathcal{E}(\mathbb{F}_p)$ to an element of an extension $\mathbb{F}_{p^r}$ of the finite field $\mathbb{F}_p$. Since the function e is bilinear, i.e., $e(g^x, g^y) = e(g, g)^{xy}$, the DDH problem can be solved by checking whether $e(g^x, g^y) = e(g, g^z)$ or not.

Now, we present some of the definitions related to bilinear pairings on elliptic curves over finite fields that will be used in the sequel of this thesis.

2.3.3 Bilinear Pairings

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of the same finite order q.

Definition 2.30. *A bilinear pairing is a map e: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, with the following properties:*

1. *e is* **bilinear***: $\forall\, x, y \in \mathbb{Z}_q$, $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, $e(g^x, h^y) = e(g, h)^{xy}$;*
2. *e is* **computable***: there is an efficient algorithm to compute $e(g, h)$ for any $(g, h) \in \mathbb{G}_1 \times \mathbb{G}_2$;*
3. *e is* **non-degenerate***: if g is a generator of $\mathbb{G}_1$ and h is a generator of $\mathbb{G}_2$, then $e(g, h)$ is a generator of $\mathbb{G}_T$.*

Typically, the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of some elliptic curve $\mathcal{E}$ over a finite field $\mathbb{F}_p$, while $\mathbb{G}_T$ is a multiplicative subgroup of an extension $\mathbb{F}_{p^r}$ of the finite field $\mathbb{F}_p$. In this context, r is called the embedding degree of the curve $\mathcal{E}$. Verheul (161) proposed computing bilinear pairings by modifying the Weil and the Tate pairings [1]. By definition, the Tate and the Weil pairings map a pair of points $(g, h) \in \mathbb{G}_1 \times \mathbb{G}_2$ to a q^{th} root of unity in $\mathbb{G}_T$.

Remark 2.5. *Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing.*

- *If $\mathbb{G}_1 = \mathbb{G}_2$, then the pairing e is said to be* symmetric *(or of Type 1). Otherwise, it is said to be* asymmetric*.*
- *If the pairing e is asymmetric and if there is an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ and no efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$, then e is said to be of Type 2. If there are efficiently computable homomorphisms in both directions, then e can be reinterpreted as a Type 1 pairing.*

[1] Bilinear pairings can be defined for all elliptic curves, however, they are efficiently computable only when the embedding degree r is small (87).

- *If the pairing e is asymmetric and if there is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$, then e is said to be of Type 3* [2].
- *If e is a pairing of Type 1, then the DDH problem is easy in $\mathbb{G}_1$. If e is a pairing of Type 2, then the DDH problem is easy in $\mathbb{G}_2$.*

Remark 2.6. *Pairing-based cryptographic schemes usually employ Type 1 and Type 2 pairings, however, Chatterjee and Menezes (38), Galbraith et al. (63) showed that Type 3 pairings offer better performances and better security.*

Type 1 pairings are generally computed in supersingular curves, while Type 2 and Type 3 pairings are computed in ordinary (non-supersingular) curves such as MNT curves proposed by Miyaji et al. (116). We refer to the work of Freeman et al. (61) for more comprehensive overview on the construction of pairing friendly curves.

Although, bilinear pairings were first introduced in cryptography to construct fast algorithms to solve the DL problem (114) and the DDH problem (87) in elliptic curves, they paved the way for practical cryptographic solutions to long standing problems such as: one-round key agreement (86), identity-based encryption (22), short signatures (23, 25), group signatures (26), secret handshake (9), ... etc. The fast development of pairing-based cryptography has led to the establishment of new hardness assumptions that we present next.

2.3.4 Bilinear Diffie-Hellman Problems

Definition 2.31 (Bilinear Computational Diffie-Hellman Problem (BCDHP))**.** *Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing. Let g be a generator of $\mathbb{G}_1$ and h be a generator of $\mathbb{G}_2$.*

The bilinear computational Diffie-Hellman problem is:

Given $g, g^x, g^y, g^z \in \mathbb{G}_1$ and $h, h^x, h^y \in \mathbb{G}_2$ for random $x, y, z \in \mathbb{Z}_q$, compute $e(g,h)^{xyz}$.

We denote $U = (\mathbb{G}_1, g, g^x, g^y, g^z)$ and $V = (\mathbb{G}_2, h, h^x, h^y)$.

The advantage ϵ of an algorithm $\mathcal{A}$ in solving the BCDH problem is defined as:

$$\epsilon = Pr(\mathcal{A}(U, V) \text{ computes } e(g,h)^{xyz})$$

Definition 2.32 (Bilinear Computational Diffie-Hellman Assumption (BCDH))**.** *We say that the BCDH assumption holds, if for every probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage ϵ in solving BCDHP is negligible.*

Definition 2.33 (Bilinear Decisional Diffie-Hellman Problem (BDDHP))**.** *Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear pairing. Let g be a generator of $\mathbb{G}_1$ and h be a generator of $\mathbb{G}_2$.*

The bilinear decisional Diffie-Hellman problem is:

Given $g, g^x, g^y, g^z \in \mathbb{G}_1$, $h, h^x, h^y \in \mathbb{G}_2$ for random $x, y, z \in \mathbb{Z}_q$ and $e(g,h)^{z'} \in \mathbb{G}_T$, decide whether $z' = xyz$ or not.

[2] A homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ can always be defined, however, the computation of such a homomorphism is supposed to be as hard as the discrete logarithms in $\mathbb{G}_1$ and in $\mathbb{G}_2$ (63).

We denote U the distribution $(\mathbb{G}_1, \mathbb{G}_2, g, g^x, g^y, g^z, h, h^x, h^y, e(g,g)^{xyz})$, and V the distribution $(\mathbb{G}_1, \mathbb{G}_2, g, g^x, g^y, g^z, h, h^x, h^y, e(g,g)^{z'})$, where x, y, z and z' are selected randomly in $\mathbb{Z}_q$.

The advantage ϵ of an algorithm $\mathcal{A}$ in solving the BDDH problem is defined as:

$$\epsilon = |Pr(\mathcal{A}(U) = 1) - Pr(\mathcal{A}(V) = 1)|$$

Definition 2.34 (Bilinear Decisional Diffie-Hellman Assumption (BDDH))**.** *We say that the BDDH assumption holds, if for every probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage ϵ in solving BDDHP is negligible. This means that it is computationally infeasible for any polynomial-time algorithm $\mathcal{A}$ to distinguish between the two distributions U and V.*

Although a symmetric bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ enables solving the DDH problem in $\mathbb{G}_1$ and in $\mathbb{G}_2$, it is believed that if e is a Type 2 (Type 3 resp.) bilinear pairing, then the DDH assumption still holds in $\mathbb{G}_1$ (in both $\mathbb{G}_1$ and $\mathbb{G}_2$ resp.). To that effect, Scott (144) introduced two related hardness assumptions which are: the *external Diffie-Hellman assumption* and the *symmetric external Diffie-Hellman assumption.*

Definition 2.35 (External Diffie-Hellman Assumption (XDH))**.** *We say that the external Diffie-Hellman assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$, if $\mathbb{G}_1$ and $\mathbb{G}_2$ are two groups with the following properties:*

1. *There exists a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$;*
2. *the decisional Diffie-Hellman assumption holds in $\mathbb{G}_1$.*

Definition 2.36 (Symmetric External Diffie-Hellman Assumption (SXDH))**.** *We say that the symmetric external Diffie-Hellman assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$, if $\mathbb{G}_1$ and $\mathbb{G}_2$ are two groups with the following properties:*

1. *There exists a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$;*
2. *the decisional Diffie-Hellman assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

2.4 Summary

In this chapter, we surveyed some of the concepts of provable security, together with the cryptographic primitives that will be referenced in the remainder of this thesis. We also provided an overview of elliptic curve cryptography and bilinear pairings which are used to design our security protocols. Now, the reader can either move on to Part I where we present previous work on RFID security and privacy, or to Part II, where we introduce our secure multi-party protocols for the RFID setting.

Part I

From RFID Authentication to Privacy Preserving Supply Chain Management

3

RFID Security and Privacy

The proliferation of RFID tags comes with new threats to the security and the privacy of companies/individuals owning tags. These potential threats have given rise to an active research area that deals primarily with the formalization of security and privacy models, and with the design of secure and privacy preserving RFID authentication protocols. The main challenges in this area are the definition of formal models that comprehensively capture the capabilities of a *real world adversary*, and the design of authentication protocols **1.)** that are provably secure and privacy preserving with respect to the formal models, and **2.)** that fit the stringent computational resources of RFID tags.

The purpose of this chapter therefore is to introduce the existing privacy and security models and to survey some of the proposed RFID authentication protocols. To this end, we start with a quick overview, in Section 3.1, of RFID technology and the main privacy and security threats related to the potential deployment of this technology. We then present the formal security and privacy definitions while explaining how they capture the adversarial behavior in the RFID environment. In Section 4.2, we analyze some of the prominent authentication protocols in the literature which we classify depending on their computational requirements on RFID tags. Finally in Section 3.4, we wrap up the chapter by highlighting some of the limitations of RFID privacy and the need for a *more realistic* adversary model. The latter allows us to design secure and privacy preserving RFID protocols that go beyond simple tag-reader authentication to propose secure and efficient solutions for supply chain management. These protocols will be presented in Part II.

3.1 RFID Fundamentals

RFID is a technology that primarily identifies and tracks objects with neither direct line of sight nor human intervention. An RFID tag is a low cost wireless device which labels the object to which it is attached by having a unique and unreusable identifier. The tag's unique identifier acts as a pointer to a database entry that contains the history of the tagged object.

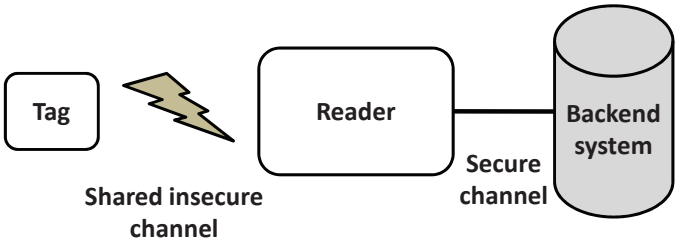

Figure 3.1: RFID environment

Consequently, RFID technology was envisioned to replace barcodes in the supply chain as it favors fast and automated product identification and tracking, together with the possibility of recording and tracing the history of tagged products from production, to distribution, to finally end users.

An RFID system involves more components than the already mentioned RFID tags. A typical RFID system consists of:

- RFID tags;
- RFID readers;
- backend systems.

Tags and readers communicate over a shared insecure wireless channel, whereas the channel between RFID readers and backend systems is generally assumed to be secure.

In the sequel of this chapter, we discuss in more details the components and the applications of RFID technology. Then, we list some of the security and the privacy challenges that hinder the deployment of this technology.

For a more thorough description of RFID systems, we refer to (59).

3.1.1 RFID Tags

An RFID tag consists of a small microchip that features limited data storage, limited logical functionalities, and an antenna. Tags can be classified based on their operating frequency. High frequency HF tags operate at 13.56 MHz frequency and their maximum read range is 1 m. Ultra-high frequency UHF tags operate in the 858 to 930 MHz frequency band and their average read range is 3 m. UHF tags are the dominant technology for supply chain applications, whereas HF tags are more suitable for RFID-based ticketing or near field applications. Tags can also be distinguished based on the underlying powering method (135). A *passive* tag is a tag which does not have any power supply (i.e., battery) of its own, and therefore, relies on the the signal sent by readers nearby to harvest the necessary energy it needs to reply to readers' queries. An *active* tag on the other hand, has its own power supply

and can initiate communication with readers. A *semi-active* tag is a hybrid tag which has its own power supply but never initiates communication with readers.

Passive tags are much cheaper than active ones and therefore, more suitable to replace optical barcodes in supply chains. The advantages of passive tags are their low cost, their small size and their lifetime which is not restricted by battery life. However, passive tags come with little resources and few computational capabilities which turn the design of RFID-based applications into a real challenge.

We thus primarily focus on passive tags.

3.1.2 RFID Readers and Backend Systems

RFID readers are transceivers which are able to communicate with tags using a radio frequency channel. A reader may be able to read or write data into tags. A reader consists of an antenna, a microprocessor, a power supply, and possibly an interface that enables the reader to forward the data received from tags to a backend system.

The backend system is typically a database that collects information forwarded by readers for various purposes that depend on the application for which RFID technology is used.

There are two categories of readers (59):

- Stationary readers: Readers are placed at a fixed location and permanently connected to a network so as to communicate with the backend system, e.g., RFID-based access control systems where readers are located at the entry point of some secured area.
- Portable readers: Readers can be handheld and not be required to communicate permanently with a backend system. They are mostly used for querying prices of products at a supermarket or for inventorying.

3.1.3 RFID Applications

RFID technology can be embedded into various applications depending on the purpose of tag identification. The most prominent applications for RFID are automated payment, access control, tracking and *supply chain management*.

The largest area of RFID applications is supply chain management whereby tags store application specific data in addition to tag identifiers. This additional data is used to automate and to regulate the processes of production and distribution in the supply chain, while minimizing errors and human intervention. By attaching RFID tags to products, managers of supply chains can automatically identify counterfeits, production bottlenecks, stock shortage and the origin of defective products. This type of applications is of a great business value as it reduces both time and errors when managing products, while decreasing the number of people involved in the supply chain.

Sometimes a tag is not only required to identify itself but it is also required to prove that the proclaimed identity is legitimate through authentication. Such a functionality is needed

in applications such as automated payment, anti-counterfeiting, car immobilizers and access control to secured areas.

Another field of application is tracking the location of tagged objects. Since readers are placed in fixed and known locations, the location of a tagged object can be easily traced with a certain accuracy. Such an application is useful for example to track pets, to detect the presence of assets or products in a factory or a warehouse, or to locate people inside a building.

Moreover, the advocates of RFID tags believe that the potential ubiquity of RFID will lead to applications that assist people with daily tasks. One of these applications is "intelligent homes" with smart appliances such as washing machines that *"automatically select the appropriate wash cycles to prevent damaging delicate fabrics, or refrigerators that detect food expiration or shortage"*(89). Along the same lines, RFID technology could be used to facilitate home navigation and medication compliance for the elderly, for e.g. an RFID enabled medicine cabinet could verify whether a patient complies with his medication intakes or not (89).

3.1.4 Security and Privacy Threats

In this section, we describe some of the security and privacy threats related to the deployment of RFID technology.

3.1.4.1 Security Threats

RFID technology faces various security threats such as denial of service, relay attacks, and cloning.

- *Denial of service:* Such an attack can be performed by creating a signal in the same frequency band as legitimate readers, and causing therefore electromagnetic jamming that prevents legitimate tags from communicating with legitimate readers.
- *Relay attacks:* These attacks are implemented by placing an adversarial device between a legitimate RFID tag and a legitimate reader. This device relays information exchanged between the two legitimate parties which are fooled into thinking that they are physically close to each other.
- *Cloning:* This attack can be executed by eavesdropping on tags' communication with readers to retrieve the tags' unique identifiers, then writing these identifiers into new rewritable and reprogrammable tags. Cloning attacks could be for instance used to replace the content of tags attached to expensive objects with the content of tags attached to cheaper ones at a retail store.

To safeguard RFID systems against the above attacks, Karygiannis et al. (95) suggested some security countermeasures that can be taken. For example, cloning can be mitigated by

using challenge-response authentication protocols. However, the scarcity of computational resources in RFID tags makes the design of secure protocols withstanding attacks by powerful adversaries very challenging. Moreover, RFID distance bounding protocols (8, 27, 77, 99) have been proposed to protect against relay attacks. The idea behind distance bounding protocols is to estimate the physical distance separating readers and tags during tag-reader communication, detecting thus relay attacks.

Finally, jamming attacks can be tackled by increasing physical security near RFID readers through guards, fences, cameras, and shielding walls to block external electromagnetic signals to limit both accidental and malicious radio interferences (95).

3.1.4.2 Privacy Threats

As RFID tags respond to any reader without the consent of their owners or holders, the proliferation of RFID also brings up new exposures that can lead to potential privacy violations such as industrial espionage, consumer profiling and tracking of individuals.

- *Industrial espionage:* By eavesdropping on tagged objects traveling along the supply chain, a company can gather confidential and sensitive information about the internal business processes of an industrial competitor. Such information could be used to infer production and distribution schedules, daily rate of production, availability or shortage of stock, and the identity of suppliers and partners.

- *Consumer profiling:* A person carrying objects tagged with RFID is prone to surreptitious inventorying. By reading tags attached to products that a person carries when entering a shop, the shop owner can learn what type of products interest that person, and he may then adjust his offers based on the information he just has gathered.

- *Tracking:* As most RFID tags transmit static unique identifiers, they can be used to track the position and trace the activity of individuals holding RFID tagged objects.

In the following, we list some of the proposed approaches to mitigate the privacy threats related to RFID technology.

- Tag deactivation: RFID tags can be deactivated by using a "KILL" command sent by readers. When a tag receives the KILL command from a reader, it becomes permanently out of service. Now, to prevent denial of service attacks through tags' deactivation, the KILL command is protected with a secret PIN that only authorized readers know. Even though killing tags is a very effective measure to protect the privacy of individuals, this technique precludes the potential post-purchase applications of RFID technology.

- Proxying: This approach aims at protecting tag privacy by using privacy enforcing devices that act as RFID firewalls (94, 136). These devices relay reader requests while implementing sophisticated privacy policies. A reader's request is forwarded to a tag only when it meets the privacy policies specified by the tag holder.

- Tag blocking: This approach protects tag privacy by relying on physical measures. For instance, a Faraday cage can be used to protect tags from unauthorized reading by blocking external radio signals. It is also possible to prevent an unauthorized tag reading by using a blocker tag (93). A blocker tag exploits the properties of the anti-collision protocols that readers use to communicate with tags to disrupt tag singulation. When a reader starts a tag singulation protocol, the blocker tag simulates all tags in the universe in order to cause continuous collisions, and to eventually stall the interrogating reader.

- Pseudonyms: Instead of having a unique permanent identifier, Inoue and Yasuura (82) proposed using tag pseudonyms that change over time to prevent tracking. A reader is required to periodically rewrite the pseudonym (identifier) of tags that it is reading while keeping a record of tags' old pseudonyms.

- Re-encryption: While encrypting tags' identifiers may protect identifier confidentiality, it cannot prevent the tracking of tags. When the identifier of a tag is encrypted, the tag sends the encryption of its identifier when queried, instead of sending its identifier in cleartext. However, this encryption can serve as a "new identifier" to trace and track the tag. To tackle this limitation, Ateniese et al. (3), Golle et al. (73), Juels and Pappu (90) suggest using re-encryption techniques. A tag in this approach stores an IND-CPA encryption (cf. Definition 2.17) of its identifier. When a reader reads the encryption c stored into a given tag T, it re-encrypts the ciphertext c to obtain a new ciphertext c' and then it writes c' into T. Consequently, an adversary cannot track tags over a long period of time.

- Privacy preserving authentication: This approach allows tags to authenticate themselves to legitimate readers in a privacy preserving manner. That is, after tag authentication, adversaries only learn whether the tag authentication was successful, while only legitimate readers can identify tags.

Most of previous work on RFID security and privacy has focused on

- Privacy preserving authentication protocols that suit the resource constraints of RFID tags. These protocols range from lightweight authentication protocols that rely on bitwise operations (18, 66, 91), to symmetric authentication protocols (48, 50, 58, 122, 153), to finally public key authentication protocols (103, 113, 126).

- Formal security and privacy models that provide a comprehensive description of the adversary's capabilities and goals (5, 92, 129, 159).

We present the prominent formal RFID security and privacy definitions in Section 3.2, then in Section 3.3, we discuss in more details some of the state of the art of RFID authentication.

3.2 RFID Security and Privacy

As highlighted in the previous section, the widespread deployment of RFID technology poses threats to the security and the privacy of individuals and companies. To mitigate these issues, a myriad of RFID authentication protocols have been proposed in the literature (6, 48, 50, 66, 91, 103, 122, 126, 153). The emphasis on these protocols has spurred attempts by Avoine (5), Juels and Weis (92), Vaudenay (159) to formalize both RFID security and privacy.

Before presenting the security and the privacy definitions regarding RFID authentication, we introduce the conventions and the notations that will be used throughout this section.

3.2.1 Definitions

In line with previous work on RFID security and privacy, we assume that the RFID system involves one reader R and that reader R and the backend system form one single entity.

Definition 3.1. *An RFID system is composed of*

- $\mathsf{InitReader}(\tau)$ *is a probabilistic algorithm which on input of a security parameter* τ *generates a pair of secret key and public key* $(\mathsf{sk}, \mathsf{pk})$ *for reader* R*. It also creates a database* DB_R *which will contain the identifiers and the keys of legitimate tags in the system.*

- $\mathsf{InitTag}(\tau, \mathsf{ID}, \mathsf{pk})$ *is a probabilistic algorithm which on input of security parameter* τ*, tag identifier* ID *and reader* R*'s public key* pk *returns* **first***, a pair* (K_T, S_T)*, where* K_T *is the secret key of tag* T *corresponding to identifier* ID*, and* S_T *is the initial state of tag* T*.* **Then***, it stores the pair* (ID, K_T) *into database* DB_R*. Let* $\mathcal{T}$ *denote the set of tags that were initialized by* $\mathsf{InitTag}$*.*

- $\pi(R, T)$ *is a polynomial-time interactive protocol between reader* R *and tag* T*. At the end of the protocol execution, reader* R *either identifies tag* T *and outputs* $b = 1$*, or rejects tag* T *and outputs* $b = 0$*.*

We have now to define the capabilities of an adversary $\mathcal{A}$ against such a system. It is assumed that reader R cannot be corrupted by adversary $\mathcal{A}$. However, adversary $\mathcal{A}$ may **1.)** play the role of dishonest readers and interact with tags. He may as well **2.)** intercept messages exchanged between tags and reader R, and also **3.)** access the internal states of tags. Finally, he may **4.)** access the output of reader R at the end of a protocol execution.

To capture formally the above capabilities, adversary $\mathcal{A}$ is given access to the following oracles that are controlled by some challenger $\mathcal{C}$.

- $\mathcal{O}_{\mathsf{Tag}}(\mathsf{param})$: When queried, the oracle $\mathcal{O}_{\mathsf{Tag}}$ returns a tag T from the set $\mathcal{T}$ to adversary $\mathcal{A}$ that satisfies the parameters param specified by adversary $\mathcal{A}$. A parameter could be for instance the probability distribution according to which the oracle $\mathcal{O}_{\mathsf{Tag}}$ samples tags.

- $\mathcal{O}_{\mathsf{Read}}(T)$: When queried with tag T, the oracle $\mathcal{O}_{\mathsf{Read}}(T)$ returns the current state S_T of tag T. When $\mathcal{A}$ calls the oracle $\mathcal{O}_{\mathsf{Read}}$ with tag T, we say that $\mathcal{A}$ corrupts tag T in accordance with previous work of Vaudenay (159) and Paise and Vaudenay (129).
- $\mathcal{O}_{\mathsf{Write}}(T, S'_T)$: When called with tag T and state S'_T, the oracle $\mathcal{O}_{\mathsf{Write}}$ rewrites the current state of tag T with the state S'_T.
- $\mathcal{O}_{\mathsf{Launch}}(T, m)$: When called, the oracle $\mathcal{O}_{\mathsf{Launch}}$ invokes reader R to start a new session of the RFID protocol π. Reader R then generates a session identifier sid and sends m and sid to tag T.
- $\mathcal{O}_{\mathsf{Result}}(\mathsf{sid})$: When the session sid of the RFID protocol π is complete, the oracle $\mathcal{O}_{\mathsf{Result}}$ returns a bit b, such that $b = 1$, if reader R outputs 1, and $b = 0$ otherwise.
- $\mathcal{O}_{\mathsf{SendR}}(m, \mathsf{sid})$: When queried with message m and session identifier sid, the oracle $\mathcal{O}_{\mathsf{SendR}}$ sends message m to reader R for the protocol session sid, and outputs the response r of reader R.
- $\mathcal{O}_{\mathsf{SendT}}(m, T)$: When queried with message m and tag T, the oracle $\mathcal{O}_{\mathsf{SendT}}$ sends message m to tag T, and outputs the response r of tag T.
- $\mathcal{O}_{\mathsf{Execute}}(T)$: When called with tag T, the oracle $\mathcal{O}_{\mathsf{Execute}}$ executes a complete session of protocol π between reader R and tag T, by querying first the oracle $\mathcal{O}_{\mathsf{Launch}}$, and then by making successive calls to the oracles $\mathcal{O}_{\mathsf{SendR}}$ and $\mathcal{O}_{\mathsf{SendT}}$. At the end of the protocol execution, the oracle $\mathcal{O}_{\mathsf{Execute}}$ returns the transcript $\mathsf{tran} = (\mathsf{sid}, m_1, r_1, m_2, r_2, ...)$ of the protocol execution together with the session identifier sid.

3.2.2 Security

We note first that a legitimate tag T is defined to be a tag T whose current state S_T corresponds to some pair (ID, K_T) in DB_R.

Now, an RFID scheme is said to be secure if it ensures both *completeness* and *soundness*.

3.2.2.1 Completeness

Roughly speaking, completeness ensures that when a legitimate tag $T \in \mathcal{T}$ engages in an execution of the RFID protocol with reader R, then the protocol outputs $b = 1$, meaning that reader R accepts tag T.

Definition 3.2. *An RFID scheme is complete* $\Leftrightarrow$ *If* T *is a legitimate tag, then* $\pi(R, T) = 1$.

Deng et al. (46) defined *adaptive completeness* which says that after any attack strategy followed by adversary $\mathcal{A}$, the protocol execution between reader R and any legitimate tag T should still be complete, i.e., $\pi(R, T) = 1$. Adaptive completeness captures particularly the ability of an RFID scheme to recover from desynchronizing attacks.

Adaptive completeness is defined using a game as depicted in Algorithm 3.2.1 and Algorithm 3.2.2. In the learning phase, adversary $\mathcal{A}$ is allowed to execute the RFID protocol for any tag in the RFID system by calling the oracle $\mathcal{O}_{\mathsf{Execute}}$, and to access the output of the protocol execution by querying the oracle $\mathcal{O}_{\mathsf{Result}}$. He is also allowed to corrupt tags by querying the oracles $\mathcal{O}_{\mathsf{Read}}$ and $\mathcal{O}_{\mathsf{Write}}$.

Algorithm 3.2.1: Learning phase of adaptive completeness (46)

// $\mathcal{A}$ may call the following oracles in any interleaved order for a polynomial number of
// times
$T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{T_i})$;
$(\mathsf{tran}_i, \mathsf{sid}_i) \leftarrow \mathcal{O}_{\mathsf{Execute}}(T_i)$;
$b_i \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid}_i)$;
$S_{T_i} \leftarrow \mathcal{O}_{\mathsf{Read}}(T_i)$;
$\mathcal{O}_{\mathsf{Write}}(T_i, S'_{T_i})$;

Algorithm 3.2.2: Challenge phase of adaptive completeness (46)

// $\mathcal{A}$ selects a challenge tag T_c which he did not corrupt in the learning phase
$\mathsf{T}_c \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{\mathsf{T}_c})$;
// Challenger $\mathcal{C}$ executes the RFID protocol for tag T_c
$(\mathsf{tran}_c, \mathsf{sid}_c) \leftarrow \mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_c)$;
$b \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid}_c)$;

In the challenge phase, adversary $\mathcal{A}$ selects a challenge tag T_c which he did not corrupt in the learning phase. The challenger then invokes the RFID protocol between the challenge tag T_c and reader R, and returns a bit b which is the output of the protocol execution.

$\mathcal{A}$ is said to win the adaptive completeness game, if $b = 0$. The advantage ϵ of adversary $\mathcal{A}$ in breaking adaptive completeness is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 3.3. *An RFID scheme is said to ensure adaptive completeness,* **iff** *for any adversary $\mathcal{A}$, the advantage ϵ in winning the adaptive completeness game is negligible.*

3.2.2.2 Soundness

Soundness ensures that when a tag T and reader R engages in an execution of the RFID protocol that ends with reader R outputting $b = 1$, then this implies that T is a legitimate tag with an overwhelming probability.

Soundness is formalized in (159) using a security game as described in Algorithm 3.2.3 and Algorithm 3.2.4. In the learning phase, adversary $\mathcal{A}$ can execute the RFID protocol for any tag in the RFID system, access the output of the protocol execution, and corrupt tags.

Algorithm 3.2.3: Learning phase of soundness (159)

// $\mathcal{A}$ may call the following oracles in any interleaved order for a polynomial number of
// times
$T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{T_i})$;
$(\mathsf{tran}_i, \mathsf{sid}_i) \leftarrow \mathcal{O}_{\mathsf{Execute}}(T_i)$;
$b_i \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid}_i)$;
$S_{T_i} \leftarrow \mathcal{O}_{\mathsf{Read}}(T_i)$;
$\mathcal{O}_{\mathsf{Write}}(T_i, S'_{T_i})$;

Algorithm 3.2.4: Challenge phase of soundness (159)

$(\mathsf{tran}_c, \mathsf{sid}_c) \leftarrow \mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_c)$;
$b \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid}_c)$;

In the challenge phase, adversary $\mathcal{A}$ engages in an execution of the RFID protocol with reader R by sending messages through the oracle $\mathcal{O}_{\mathsf{SendR}}$. That is, adversary $\mathcal{A}$ impersonates some legitimate tag T_c to reader R. At the end of the challenge phase, the challenger $\mathcal{C}$ returns a bit b which is the output of the protocol execution between reader R and adversary $\mathcal{A}$.

Adversary $\mathcal{A}$ wins the soundness game,

- if $b = 1$, meaning that the RFID protocol identified some legitimate tag T_c; and if
- tag T_c is not corrupted by adversary $\mathcal{A}$; and if
- tag T_c and reader R did not engage in a protocol execution that has the same transcript $\mathsf{tran}_c = (\mathsf{sid_c}, m_1, r_1, m_2, r_2, ...)$ as the protocol execution between reader R and adversary $\mathcal{A}$. That is, adversary $\mathcal{A}$ did not perform a relay attack between reader R and legitimate tag T_c.

The advantage ϵ of adversary $\mathcal{A}$ in winning the soundness game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 3.4. *An RFID scheme is said to be sound,* **iff** *for any adversary $\mathcal{A}$, the advantage ϵ in winning the soundness game is negligible.*

We note that the soundness game above does not grasp the soundness of mutual authentication in the RFID setting (i.e., also reader R is authenticated by tags). We point out however, that the difference between soundness as defined above and soundness of mutual authentication lies in that adversary $\mathcal{A}$ wins the soundness game not only if he successfully impersonates a legitimate tag T_c when interacting with reader R, but also if he successfully impersonates reader R during a mutual authentication with some legitimate tag T_c.

Next, we introduce the prominent privacy definitions in the RFID literature.

Algorithm 3.2.5: Learning phase of strong privacy (92)

// $\mathcal{A}$ may call the following oracles in any interleaved order for a polynomial number of
// times
$T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{T_i})$;
$(\mathsf{tran}_i, \mathsf{sid}_i) \leftarrow \mathcal{O}_{\mathsf{Execute}}(T_i)$;
$b_i \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid}_i)$;
$S_{T_i} \leftarrow \mathcal{O}_{\mathsf{Read}}(T_i)$;
$\mathcal{O}_{\mathsf{Write}}(T_i, S'_{T_i})$;

3.2.3 Privacy

Formalizing RFID privacy has been a challenging task that resulted in several privacy definitions (5, 92, 129, 159). These definitions can be classified into *three* categories: *indistinguishability-based privacy*, *unpredictability-based privacy* and *simulator-based privacy*, that differ mainly in the approach used to measure information leakage during the execution of an RFID protocol.

3.2.3.1 Indistinguishability-based Privacy

One of the first attempts to formalize RFID privacy was presented in (5). Avoine (5) introduced the notion of *tag untraceability* (also known as tag *unlinkability*) which is formalized by the ability of an adversary $\mathcal{A}$ to distinguish between two challenge tags T_0 and T_1 based on their protocol executions. Avoine (5) discriminated between *universal* untraceability and *existential* untraceability. Universal untraceability captures the ability of adversary $\mathcal{A}$ to distinguish between the challenge tags T_0 and T_1 at any point of time, whereas existential untraceability grasps the ability of adversary $\mathcal{A}$ to distinguish between the challenge tags T_0 and T_1 at some specific time window chosen by adversary $\mathcal{A}$. Extending the work by Ohkubo et al. (122), Avoine (5) formally defined tag *forward privacy* or *forward untraceability*. Forward privacy ensures that even if adversary $\mathcal{A}$ corrupts some tag T (i.e., reveals its internal state), he still cannot link T to its past protocol executions that took place before T's corruption.

However, this privacy definition does not allow adversary $\mathcal{A}$ to select the challenge tags, neither does it take into account the availability of reader R's output (i.e., the protocol output) to the adversary $\mathcal{A}$. Actually, adversary $\mathcal{A}$ can always learn whether an RFID protocol execution succeeded on the reader by only observing the reader's behavior, for e.g. a gate that opens or not at a subway station.

Juels and Weis (92) extended the definition of tag unlinkability and introduced the notion of *strong privacy*. As in (5), strong privacy is formalized using a game that captures the ability of an adversary $\mathcal{A}$ to tell two challenge tags T_0 and T_1 apart as depicted in Algorithm 3.2.5 and Algorithm 3.2.6. An adversary $\mathcal{A}$ against strong privacy has access to tags in two phases. In the learning phase, $\mathcal{A}$ is allowed to execute the RFID protocol while accessing its output.

Algorithm 3.2.6: Challenge phase of strong privacy (92)

// $\mathcal{A}$ selects two tags T_0 and T_1 which he did not corrupt in the learning phase
$\mathsf{T}_0 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{\mathsf{T}_0})$;
$\mathsf{T}_1 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{\mathsf{T}_1})$;
$b \leftarrow \{0,1\}$; // challenger $\mathcal{C}$ flips a fair coin b in $\{0,1\}$
$\mathcal{A} \leftarrow \mathsf{T}_b$; // challenger $\mathcal{C}$ provides $\mathcal{A}$ with access to tag T_b
// $\mathcal{A}$ may call the following oracles in any interleaved order for a polynomial number of
// times
$T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{T_i})$;
$(\mathsf{tran}_i, \mathsf{sid}_i) \leftarrow \mathcal{O}_{\mathsf{Execute}}(T_i)$;
$b_i \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid}_i)$;
$S_{T_i} \leftarrow \mathcal{O}_{\mathsf{Read}}(T_i)$; // $T_i \neq \mathsf{T}_b$
$\mathcal{O}_{\mathsf{Write}}(T_i, S'_{T_i})$; // $T_i \neq \mathsf{T}_b$
// $\mathcal{A}$ outputs his guess for bit b
Output b';

He is also allowed to corrupt tags. Contrary to (5), adversary $\mathcal{A}$ is allowed to select two challenge tags T_0 and T_1 in the challenge phase, under the restriction that these two tags should not be corrupted by $\mathcal{A}$ in the learning phase. Then, challenger $\mathcal{C}$ gives adversary $\mathcal{A}$ access to tag T_b selected randomly from $\{\mathsf{T}_0, \mathsf{T}_1\}$. Adversary $\mathcal{A}$ then can execute the RFID protocol while accessing its output, and corrupt any tag in the RFID system except for tag T_b. The challenge phase of strong privacy ends with adversary $\mathcal{A}$ outputting a guess bit b' for the bit b.

Adversary $\mathcal{A}$ wins the strong privacy game if $b' = b$. The advantage ϵ of adversary $\mathcal{A}$ in winning the strong privacy game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins}) - \frac{1}{2}$$

Definition 3.5. *An RFID scheme is said to ensure strong privacy,* **iff** *for any adversary $\mathcal{A}$, the advantage ϵ in winning the strong privacy game is negligible.*

3.2.3.2 Unpredictability-based Privacy

Ha et al. (75) introduced the notion of *unp-privacy* (short for unpredictability-based privacy) which defines privacy by the ability of an adversary $\mathcal{A}$ to predict the output of a tag or a reader when engaging in an RFID protocol. The unp-privacy is defined with respect to 3-round canonical RFID protocol. The RFID protocol starts when reader R sends a challenge message $m_1 \in \{0,1\}^{k_1}$ to some tag T, then tag T replies with a response message $r \in \{0,1\}^k$, the protocol ends with reader R sending a third message $m_2 \in \{0,1\}^{k_2}$ (in the case of mutual authentication).

The unp-privacy is formalized using a privacy game where adversary $\mathcal{A}$ accesses the RFID system in two phases. In the learning phase, cf. in Algorithm 3.2.7, adversary $\mathcal{A}$ is allowed

to execute the RFID protocol, to access the result of the protocol execution and to corrupt tags in the RFID system.

Algorithm 3.2.7: Learning phase of unp-privacy (75)

// $\mathcal{A}$ may call the following oracles in any interleaved order for a polynomial number of
// times
$T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{T_i})$;
$(\mathsf{tran}_i, \mathsf{sid}_i) \leftarrow \mathcal{O}_{\mathsf{Execute}}(T_i)$;
$b_i \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid}_i)$;
$S_{T_i} \leftarrow \mathcal{O}_{\mathsf{Read}}(T_i)$;
$\mathcal{O}_{\mathsf{Write}}(T_i, S'_{T_i})$;

In the challenge phase, cf. Algorithm 3.2.8, adversary $\mathcal{A}$ selects a challenge tag T_c that he did not corrupt and a challenge message m_1. The challenger $\mathcal{C}$ flips a fair coin $b \in \{0, 1\}$. If $b = 1$, then challenger $\mathcal{C}$ executes the RFID protocol with tag T_c and sends message m_1 in the first round of the protocol. Finally, challenger $\mathcal{C}$ returns to adversary $\mathcal{A}$ the transcript $\mathsf{tran}_{\mathsf{T}_c} = (m_1, r^*, m_2^*)$ of the protocol execution. If $b = 0$, then challenger $\mathcal{C}$ returns to $\mathcal{A}$ a transcript $\mathsf{tran}_{\mathsf{T}_c} = (m_1, r^*, m_2^*)$ where r^* and m_2^* are random strings.

As in the learning phase, adversary $\mathcal{A}$ is allowed to corrupt and execute the RFID protocol while accessing its output for any tag except for the challenge tag T_c. At the end of the challenge phase, adversary $\mathcal{A}$ outputs his guess bit b' for the bit b.

Adversary $\mathcal{A}$ is said to win the unp-privacy game, if $b' = b$. The advantage ϵ of adversary $\mathcal{A}$ in winning the unp-privacy game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins}) - \frac{1}{2}$$

Definition 3.6. *An RFID scheme is said to ensure unp-privacy,* **iff** *for any adversary $\mathcal{A}$, the advantage ϵ in winning the unp-privacy game is negligible.*

Ma et al. (112) showed that the minimal condition for RFID tags to achieve unp-privacy is to implement a *pseudo-random function family* (PRF). However, Deng et al. (46) identified two limitations to the unp-privacy formalization:

- Unp-privacy requires messages (r, m_2) to be pseudorandom. Nonetheless, any privacy preserving RFID protocol $\pi = (m_1, r, m_2)$ can be transformed into an RFID protocol $\pi' = (m_1, r||1, m_2)$, where $||$ denotes string concatenation operation, that is not privacy preserving according to the unp-privacy definition, since the message $r||1$ is not pseudorandom. Yet intuitively, the protocol π' is privacy preserving as all tags in the RFID system appends the same bit 1 to their reply r.

- Adversary $\mathcal{A}$ is not allowed to access the output of the protocol for tag T_c in the challenge phase. As pointed by Deng et al. (46), adversary $\mathcal{A}$ can break unp-privacy by

Algorithm 3.2.8: Challenge phase of unp-privacy (75)

// $\mathcal{A}$ selects a tag T_c that is not corrupted and returns a challenge message c
$\mathsf{T}_c \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{\mathsf{T}_c})$;
$m_1 \leftarrow \mathcal{A}$;
// Challenger $\mathcal{C}$ executes the protocol with tag T_c
$(m_1, r, m_2) \leftarrow \mathcal{C}$;
$b \leftarrow \{0,1\}$; // challenger $\mathcal{C}$ flips a fair coin $b \in \{0,1\}$
if $b = 1$ **then**
  $(m_1, r^*, m_2^*) = (m_1, r, m_2)$;
end
else
  $r^* \leftarrow \{0,1\}^k$; // Challenger $\mathcal{C}$ picks r^* randomly
  $m_2^* \leftarrow \{0,1\}^{k_2}$; // Challenger $\mathcal{C}$ picks m_2^* randomly
end
$\mathsf{tran}_{T_c} = (m_1, r^*, m_2^*)$;
$\mathcal{A} \leftarrow \mathsf{tran}_{\mathsf{T}_c}$;
// $\mathcal{A}$ may call the following oracles in any interleaved order for a polynomial number of
// times for any tag $T_i \neq \mathsf{T}_c$
$T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{T_i})$;
$(\mathsf{tran}_i, \mathsf{sid}_i) \leftarrow \mathcal{O}_{\mathsf{Execute}}(T_i)$;
$b_i \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid}_i)$;
$S_{T_i} \leftarrow \mathcal{O}_{\mathsf{Read}}(T_i)$;
$\mathcal{O}_{\mathsf{Write}}(T_i, S'_{T_i})$;
// $\mathcal{A}$ outputs his guess for bit b
Output b';

forwarding the message r received from challenger $\mathcal{C}$ to reader R. If reader R accepts the message r, then this implies that r was generated by tag T_c and adversary $\mathcal{A}$ outputs $b = 1$. Otherwise, r is a random string and adversary $\mathcal{A}$ outputs $b = 0$.

3.2.3.3 Simulator-based Privacy

Vaudenay (159) introduced a comprehensive privacy model where privacy is defined as the ability of an adversary $\mathcal{A}$ to infer *non-trivial* information about tags' ID from protocol messages exchanged between tags in the RFID system and reader R. According to (159), an RFID scheme is said to be privacy preserving, if the messages exchanged between a tag T and reader R leak no information about tag T to adversary $\mathcal{A}$. That is, the interaction between tag T and reader R can be successfully simulated to adversary $\mathcal{A}$ without the secret information of tag T or the reader.

This privacy definition is formalized by considering two adversaries $\mathcal{A}$ and $\mathcal{A}^{\mathsf{S}}$, as illustrated in Algorithm 3.2.9 and Algorithm 3.2.10. In the learning phase, both adversaries have access to the RFID system through the set of oracles presented in Section 3.2.1. The difference between the two adversaries lies in the fact that adversary $\mathcal{A}$ is provided access to the

oracle $\mathcal{O}_{\mathsf{Execute}}$ and the oracle $\mathcal{O}_{\mathsf{Result}}$, while adversary $\mathcal{A}^{\mathcal{S}}$ has access instead to the simulated oracles $\mathcal{O}^{\mathcal{S}}_{\mathsf{Execute}}$ and $\mathcal{O}^{\mathcal{S}}_{\mathsf{Result}}$ controlled by some simulator $\mathcal{S}$, which does not know the secrets of tags or the secrets of reader R. Hence, adversary $\mathcal{A}^{\mathcal{S}}$ is said to be a *blinded* adversary.

Algorithm 3.2.9: Learning phase of the privacy game as defined in (159)

Adversary $\mathcal{A}^{\mathcal{S}}$
// $\mathcal{A}^{\mathcal{S}}$ may call the following oracles in an interleaving order for a polynomial number of
// times
$T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{T_i})$;
$(\mathsf{tran}_i, \mathsf{sid}_i) \leftarrow \mathcal{O}^{\mathcal{S}}_{\mathsf{Execute}}(T_i)$; // simulator $\mathcal{S}$ simulates $\mathcal{O}_{\mathsf{Execute}}$
$b_i \leftarrow \mathcal{O}^{\mathcal{S}}_{\mathsf{Result}}(\mathsf{sid})$; // simulator $\mathcal{S}$ simulates $\mathcal{O}_{\mathsf{Result}}$
$S_{T_i} \leftarrow \mathcal{O}_{\mathsf{Read}}(T_i)$;
$\mathcal{O}_{\mathsf{Write}}(T'_i, S'_{T_i})$;
Adversary $\mathcal{A}$
// $\mathcal{A}$ may call the following oracles in an interleaving order for a polynomial number of
// times
$T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{T_i})$;
$(\mathsf{tran}_i, \mathsf{sid}_i) \leftarrow \mathcal{O}_{\mathsf{Execute}}(T_i)$;
$b_i \leftarrow \mathcal{O}_{\mathsf{Result}}(\mathsf{sid})$;
$S_{T_i} \leftarrow \mathcal{O}_{\mathsf{Read}}(T_i)$;
$\mathcal{O}_{\mathsf{Write}}(T'_i, S'_{T_i})$;

In the challenge phase, adversary $\mathcal{A}$ and adversary $\mathcal{A}^{\mathcal{S}}$ are provided with tables $\mathcal{T}$ and $\mathcal{T}^{\mathcal{S}}$ respectively that contain the identifiers of tags that $\mathcal{A}$ and $\mathcal{A}^{\mathcal{S}}$ accessed in the learning phase.

At the end of the challenge phase, adversary $\mathcal{A}$ and blinded adversary $\mathcal{A}^{\mathcal{S}}$ are required to output a bit $b \in \{0,1\}$ and a bit $b^{\mathcal{S}} \in \{0,1\}$ respectively.

Now, the advantage ϵ of adversary $\mathcal{A}$ in winning the privacy game is defined as:

$$\epsilon = |Pr(\mathcal{A} \text{ outputs } b = 1) - Pr(\mathcal{A}^{\mathcal{S}} \text{ outputs } b^{\mathcal{S}} = 1)|$$

Definition 3.7. *An RFID scheme is said to ensure privacy according to the definition of Vaudenay (159),* **iff** *for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that the advantage ϵ defined above is negligible.*

As indicated in (159), the privacy definition above captures information leakage through the wireless channel between tags and reader R in the RFID system but not through tag corruption, since queries to the oracles $\mathcal{O}_{\mathsf{Read}}$ and $\mathcal{O}_{\mathsf{Write}}$ are not simulated. In other words, tag corruption is assumed to always compromise tag privacy.

Within the Vaudenay's model, adversaries against RFID schemes are categorized into the following classes:

- *Weak adversary*: Adversary $\mathcal{A}$ is not allowed to corrupt tags, i.e., adversary $\mathcal{A}$ cannot call the oracle $\mathcal{O}_{\mathsf{Read}}$ nor can he call the oracle $\mathcal{O}_{\mathsf{Write}}$.

Algorithm 3.2.10: Challenge phase of the privacy game as defined in (159)

Adversary $\mathcal{A}^{S}$
// Let $\mathcal{T}^{S}$ be the table of the identifiers of tags that were accessed by $\mathcal{A}^{S}$ in the learning
// phase
$\mathcal{T}^{S} \leftarrow \mathcal{C}$; // challenger $\mathcal{C}$ returns table $\mathcal{T}^{S}$ to $\mathcal{A}^{S}$
Output b^{S};
Adversary $\mathcal{A}$
// Let $\mathcal{T}$ be the table of the identifiers of tags that were accessed by $\mathcal{A}$ in the learning
// phase
$\mathcal{T} \leftarrow \mathcal{C}$; // Challenger $\mathcal{C}$ returns table $\mathcal{T}$ to $\mathcal{A}$
Output b;

- *Forward adversary*: Adversary $\mathcal{A}$ is allowed to corrupt tags. However, once adversary $\mathcal{A}$ corrupts a tag, he cannot do anything except for corrupting more tags. A protocol that ensures privacy against forward adversaries is said to be *forward* privacy preserving.

- *Destructive adversary*: Adversary $\mathcal{A}$ is allowed to do anything after corrupting a tag, but under the restriction that adversary $\mathcal{A}$ cannot reuse a tag after corrupting it. Adversary $\mathcal{A}$ can neither interact with a corrupted tag nor impersonate a corrupted tag to reader R.

- *Strong adversary*: Adversary $\mathcal{A}$ can corrupt tags without any restrictions.

Furthermore, for each class of adversary $\mathcal{A}$, Vaudenay (159) defined two variants. **1.)** *Narrow*, where adversary $\mathcal{A}$ is not allowed to access the output of the protocol by reader R, i.e., adversary $\mathcal{A}$ cannot call the oracle $\mathcal{O}_{\mathsf{Result}}$. **2.)** *Wide* or *non-narrow*, where adversary $\mathcal{A}$ can call the oracle $\mathcal{O}_{\mathsf{Result}}$. We note that a *non-narrow strong* adversary corresponds to adversary $\mathcal{A}$ described in Algorithm 3.2.9 and Algorithm 3.2.10.

In (159), Vaudenay established that privacy against a non-narrow strong adversary is impossible, and that narrow strong privacy can be achieved if the tags and the reader in the RFID system implement a key agreement protocol. Moreover, Paise and Vaudenay (129) extended the above privacy definition to take into account mutual authentication protocols, and showed that an RFID scheme that ensures secure mutual authentication, can ensure narrow forward privacy only if tags feature erasable temporary memory.

As it is impossible to have an RFID scheme that is privacy preserving against *strong* adversaries, several adaptations of the model of Vaudenay (159) have been proposed to formalize a weaker, yet a realistic privacy definition. For example, Ng et al. (120) introduced the notion of *wise* adversary, who is an adversary that cannot query the same oracle twice with the same input nor can he call oracles with queries to which he already knows the answer. Under these restrictions, Ng et al. (120) showed that privacy under tag corruption can be achieved, however, their privacy model prohibits adversaries from accessing the oracle $\mathcal{O}_{\mathsf{Result}}$.

Also, Deng et al. (46) introduced a new definition for RFID privacy called *zero-knowledge* privacy (zk-privacy for short). Similar to the definition of Vaudenay (159), information leakage is measured by comparing the view of an adversary $\mathcal{A}$ who has access to the RFID system through oracles and the view of a *blinded* adversary $\mathcal{A}^{S}$ who has access to a simulated RFID system. However, the definition of Deng et al. (46) focuses on deriving information about a specific challenge tag, contrary to the definition of Vaudenay (159) where privacy is defined as the inability of an adversary $\mathcal{A}$ to learn information about any tag in the RFID system.

3.3 RFID Authentication Protocols

Designing RFID authentication protocols proved to be a very challenging research topic, since these protocols must not only be secure and privacy preserving, but must also fit the stringent characteristics of RFID tags in terms of gate equivalents (G.E. for short) and power consumption; a tag is assumed to provide 10000 G.E. in average and operates at 1 mW. These strict requirements have led to several proposals in the literature that can be categorized into four main classes: lightweight authentication, symmetric authentication, public key authentication and physical authentication. These classes of authentication protocols differ in the computational requirements on the tags and on the reader. Lightweight authentication relies on lightweight binary operations, while symmetric authentication requires that tags compute symmetric cryptographic operations. We note that both lightweight authentication and symmetric authentication require the backend server to perform a linear amount of computation in the number of tags in the RFID system. In order to allow constant time authentication while ensuring both tag privacy and security, some protocols use public key primitives. Namely, elliptic curve cryptography that uses relatively short keys and which can be efficiently implemented in hardware (102, 104). Finally, RFID protocols based on physical approaches exploit the physical properties of the RFID environment to enforce tag privacy and security.

3.3.1 Lightweight Authentication

Lightweight primitives require RFID tags to only compute bit-wise operations such as "$\oplus$", "$\vee$" and "$\wedge$", and to store relatively short keying material, which suit perfectly the constrained computational resources of RFID tags. As a result, the design of secure lightweight primitives was the focus of a lot of work on secure and privacy preserving RFID authentication, cf. (18, 29, 66, 91, 130, 131, 157). However, most of these protocols were shown to be vulnerable to key recovery attacks see, (14, 64, 108, 109, 128).

In this section, we first survey the HB^{+} protocol (91) and some of its variants (29, 66). Then, we describe the F_f protocol (18) which we were able to break using an attack of 2^{39} steps.

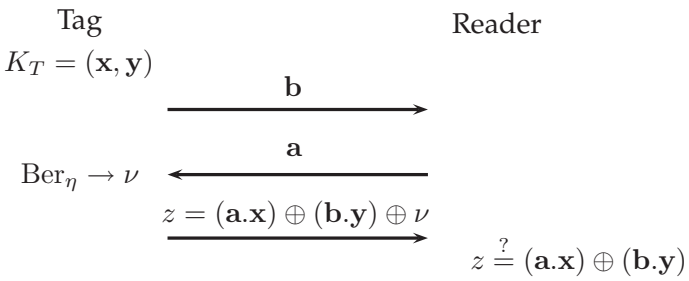

Figure 3.2: The HB$^+$ protocol

3.3.1.1 The HB Protocols

Among the well-investigated lightweight RFID authentication protocols there are the HB protocols (HB$^+$ (91), HB^{++} (29) , HB$^\#$ (66)), whose security and privacy rely on the *learning parity with noise* (LPN for short) problem.

Definition 3.8. *Let* $\mathbf{U}$ *be a random* $q \times k$ *binary matrix, let* $\mathbf{x}$ *be a random* k*-bit vector, let* $\eta \in]0, \frac{1}{2}[$ *be a constant noise parameter, and let* $\boldsymbol{\nu}$ *be a random* q*-bit vector whose hamming weight* $\mathsf{hamm}(\boldsymbol{\nu}) \leq \eta q$.

The learning parity with noise (LPN) problem is defined as:

Given $\mathbf{U}$, η, *and* $\mathbf{z} = \mathbf{U}\mathbf{x} \oplus \boldsymbol{\nu}$, *find a* k*-bit vector* $\mathbf{x}'$ *such that:* $\mathsf{hamm}(\mathbf{U}\mathbf{x}' \oplus \mathbf{z}) \leq \eta q$.

The LPN problem is known to be NP-complete (13). The best known algorithms to solve the LPN problem have a complexity of $2^{O(\frac{k}{log(k)})}$. The first algorithm to reach this complexity was proposed by Blum et al. (20), further optimizations were introduced later by Levieil and Fouque (107), but they only led to slight improvements to the above complexity of solving the LPN problem.

The first protocol in the HB family is HB$^+$ (91), see Fig. 3.2. This protocol is a modification of the HB protocol (80) which is a protocol that addresses the problem of secure identification by humans without the assistance of trusted hardware or software. A tag T in the HB$^+$ protocol shares a secret key $K_T = (\mathbf{x}, \mathbf{y}) \in \{0,1\}^k \times \{0,1\}^k$ with reader R. In each round of the protocol execution, tag T generates a random k-bit vector $\mathbf{b} \in \{0,1\}^k$ and sends $\mathbf{b}$ to reader R. Reader R then sends a challenge vector $\mathbf{a} \in \{0,1\}^k$ to tag T. Tag T generates a bit ν according to the Bernoulli distribution Ber_η, where $\eta \in]0, \frac{1}{2}[$, and computes a reply $z = (\mathbf{a} \cdot \mathbf{x}) \oplus (\mathbf{b} \cdot \mathbf{y}) \oplus \nu$, where "$\cdot$" denotes the inner product. Reader R accepts T's reply, only if $z = (\mathbf{a} \cdot \mathbf{x}) \oplus (\mathbf{b} \cdot \mathbf{y})$, i.e., $\nu = 0$. Finally, reader R authenticates tag T after q rounds only if T's reply was rejected in less than ηq rounds (i.e., $\nu = 1$ in less than ηq rounds).

Assuming the hardness of the LPN problem, the HB$^+$ protocol is provably secure against passive adversaries (i.e., "eavesdroppers"). Additionally, Katz et al. (98) proved that HB$^+$ remains secure under concurrent executions, meaning that the HB$^+$ can be parallelized to run in fewer rounds. However, Gilbert et al. (64) showed a man in the middle attack that allows an

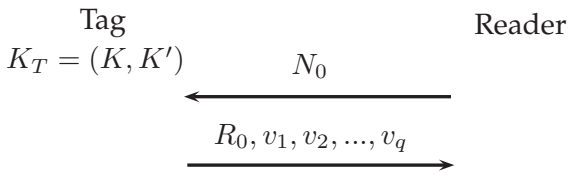

Figure 3.3: The F_f protocol

active adversary $\mathcal{A}$ to recover the secret $K_T = (\mathbf{x}, \mathbf{y})$. To thwart this attack, several protocols based on HB$^+$ have been proposed such as: HB^{++} (29) and HB-MP (119). Nonetheless, Gilbert et al. (65) showed again that these variants are not secure against man in the middle attacks. Furthermore, HB protocols are not complete. For 80-bit security, the probability of the reader rejecting a legitimate tag attains 44% as shown in (66). Consequently, Gilbert et al. (66) proposed a new variant called HB$^{\#}$ that aims to have a lower rate of false negatives and to withstand active attacks. The main idea of HB$^{\#}$ is to use $k_x \times p$ and $k_y \times p$-binary matrices $\mathbf{X}$ and $\mathbf{Y}$ as the tag 's secrets instead of k-bit binary vectors. The HB$^{\#}$ protocol proceeds similarly to HB$^+$, except that the tag is required to send a p-bit message at the end of each round instead of one bit. Now, to optimize storage requirements on tags, Gilbert et al. (66) use Toeplitz matrices that can be entirely defined by the first row and the first column, and it follows that a $k \times p$ matrix can be stored in $k + p - 1$ bits rather than in kp bits. However, Ouafi et al. (128) designed a man in the middle attack against HB$^{\#}$ that enables an adversary $\mathcal{A}$ to recover the secret matrices $(\mathbf{X}, \mathbf{Y})$.

3.3.1.2 The F_f Protocol

Inspired by the work of Di Pietro and Molva (48), Blass et al. (18) proposed F_f, a lightweight protocol for RFID tag authentication whose implementation fits in less than 2000 G.E. The main idea behind F_f is instead of relying on a secure hash function to authenticate tags, F_f uses a lightweight function called F_f whose output size is very small. The small output size of the F_f function results in a large number of collisions (i.e., for different keys, F_f outputs the same value) which is mitigated by executing the F_f protocol in q rounds (typically $q = 60$). In each round, the F_f function is computed, and its output is used by reader R to filter the secret keys that do not match. In the q^{th} round, only one secret key is left, and it corresponds to the authenticated tag.

Before detailing the F_f protocol, we describe first the F_f function. The F_f function is built upon a small fan-in function $f : \{0,1\}^l \times \{0,1\}^l \rightarrow \{0,1\}^l$, and it is defined as:

$$F_f : \{0,1\}^{lt} \times \{0,1\}^{lt} \rightarrow \{0,1\}^l, \; F_f(x,y) = \bigoplus_{i=1}^{t} f(x^{[i]}, y^{[i]})$$

Where $x^{[i]}$ respectively $y^{[i]}$ denote the i^{th} l-bit block of x, respectively y.

Now, we turn to the detailed description of the F_f protocol. Each tag T in the system stores a secret key $K_T = (K, K')$ that it shares with reader R. An execution of the protocol is as follows:

- Reader R sends a nonce $N_0 \in \{0,1\}^{lt}$ to tag T;

- tag T replies with a random number R_0 and the following q values v_i:

$$\begin{aligned} v_1 &= F_f(K, R_1^{a_1}) \oplus F_f(K', N_1) \\ v_2 &= F_f(K, R_2^{a_2}) \oplus F_f(K', N_2) \end{aligned}$$

$$v_q = F_f(K, R_q^{a_q}) \oplus F_f(K', N_q).$$

We recall that in the F_f protocol, each tag is equipped with two LFSRs. One LFSR computes q random number N_i from the nonce N_0 sent by reader R, and the other generates a random number R_0 and q sets of d random numbers $\{R_i^1, R_i^2, ..., R_i^d\}$, as shown in the following equations.

$$\begin{aligned} N_i &= \text{LFSR}(N_{i-1}) \quad 1 \le i \le q \\ R_1^1 &= \text{LFSR}(R_0) \\ R_i^1 &= \text{LFSR}(R_{i-1}^d) \quad 2 \le i \le q \\ R_i^j &= \text{LFSR}(R_i^{j-1}) \quad 1 \le i \le q,\ 2 \le j \le d \end{aligned}$$

To compute the i^{th} value v_i sent to the reader, tag T first secretly selects a number $a_i \in \{1, 2, ..., d\}$, then outputs:

$$v_i = F_f(K, R_i^{a_i}) \oplus F_f(K', N_i)$$

After receiving the response of tag T, reader R first derives the q random numbers N_i, then the q sets of the d random numbers $\{R_i^1, R_i^2, ..., R_i^d\}$. Next, for each v_i, the reader discards from its database every pair of keys (K_j, K'_j) that verifies the following:

$$\forall\ a \in \{1, 2, ..., d\},\ F_f(K_j, R_i^a) \oplus F_f(K'_j, N_i) \neq v_i$$

Contrary to the HB protocols, the F_f protocol is complete, i.e., a valid tag is never rejected. Also, if the function f is balanced, the parameters d, l and q can be chosen in such a way that minimizes the probability of breaking the soundness of F_f, see (18) for more details.

To implement F_f, Blass et al. (18) proposed a practical set of parameters as depicted in

Table 3.1: Values of F_f's parameters

lt	l	t	d	q
256	4	64	8	60

Table 3.1, and defined the function $f : \{0,1\}^4 \times \{0,1\}^4 \rightarrow \{0,1\}^4$, $f(x,y) = z$ such that:

$$\begin{aligned}
z_1 &= x_4y_1 \oplus x_1y_2 \oplus x_2y_3 \oplus x_3y_4 \oplus x_1x_2y_1y_2 \oplus x_2x_3y_2y_3 \oplus x_3x_4y_3y_4 \\
z_2 &= x_1y_1 \oplus x_2y_2 \oplus x_3y_3 \oplus x_4y_4 \oplus x_1x_3y_1y_3 \oplus x_2x_4y_2y_4 \oplus x_1x_4y_1y_4 \\
z_3 &= x_3y_1 \oplus x_4y_2 \oplus x_1y_3 \oplus x_2y_4 \oplus x_1x_2y_1y_4 \oplus x_2x_3y_2y_4 \oplus x_3x_4y_1y_3 \\
z_4 &= x_2y_1 \oplus x_3y_2 \oplus x_4y_3 \oplus x_1y_4 \oplus x_1x_3y_3y_4 \oplus x_2x_4y_2y_3 \oplus x_1x_4y_1y_2
\end{aligned}$$

Where (x_1, x_2, x_3, x_4), (y_1, y_2, y_3, y_4) and (z_1, z_2, z_3, z_4) stand for the binary representation of x, y and z respectively.

However, we showed in (14) two attacks that allow an adversary $\mathcal{A}$ to recover the secret key $K_T = (K, K')$ in 2^{52} and 2^{39} steps respectively. The first attack relies on the properties of the f function and transforms the problem of extracting K_T into the LPN problem. The second attack exploits the relatively short length (64 bits) of the LFSR's internal state. In the following, we only describe the second attack as it has better performances, and it does not depend on the properties of the f function.

The starting point of this attack is to find two protocol executions that involve the same tag T and which use the same random seed R_0. As the LFSR used to generate R_0 has an internal state of 64 bits, a tag uses the same seed R_0 after 2^{32} protocol executions.

First, adversary $\mathcal{A}$ picks two nonces $N_{(0,1)}$ and $N_{(0,2)}$, and challenges tag T with each of these nonces for 2^{32} times. Eventually, adversary $\mathcal{A}$ will find at least two protocol executions involving nonce $N_{(0,1)}$ and nonce $N_{(0,2)}$ respectively that use the same seed R_0. Therefore, adversary $\mathcal{A}$ is able to collect values $v_{(i,1)} = F_f(K, R_i^{a_{(i,1)}}) \oplus F_f(K', N_{(i,1)})$ and $v_{(i,2)} = F_f(K, R_i^{a_{(i,2)}}) \oplus F_f(K', N_{(i,2)})$ for $1 \leq i \leq q$. Now, if $F_f(K, R_i^{a_{(i,1)}}) = F_f(K, R_i^{a_{(i,2)}})$, then adversary $\mathcal{A}$ obtains the following equation:

$$v_{(i,1)} \oplus v_{(i,2)} = F_f(K', N_{(i,1)}) \oplus F_f(K', N_{(i,2)})$$

- Let π_j denote the projection from $\{0,1\}^l$ to $\{0,1\}$ that sends any element of $\{0,1\}^l$ to its j^{th} bits, i.e., for all $x = (x_1, x_2, ..., x_l) \in \{0,1\}^l$ $\pi_j(x) = x_j$.

- Let $E_{(i,j)}$ denote the event that $\pi_j(v_{(i,1)} \oplus v_{(i,2)}) = \pi_j(F_f(K', N_{(i,1)}) \oplus F_f(K', N_{(i,2)}))$.

Event $E_{(i,j)}$ occurs either when $a_{(i,1)} = a_{(i,2)}$ or when $a_{(i,1)} \neq a_{(i,2)}$ but $\pi_j(F_f(K, R_i^{a_{(i,1)}})) = \pi_j(F_f(K, R_i^{a_{(i,2)}}))$ over $\{0,1\}$. Since F_f is well balanced and R_i^j is randomly chosen from the set $\{R_i^1, R_i^2, ..., R_i^d\}$, the first case occurs with probability $\frac{1}{d}$, whereas the second case occurs

with probability $\frac{1}{2}(1-\frac{1}{d})$. Therefore, event $E_{(i,j)}$ happens with probability $\frac{1}{8}+\frac{1}{2}(1-\frac{1}{8}) = \frac{1}{2}+\frac{1}{16}$ for $d=8$.

Since $Pr(E_{(i,j)}) > \frac{1}{2}$, adversary $\mathcal{A}$ can repeat his attack several times to obtain N samples of the same equation $\pi_j(v_{(i,1)} \oplus v_{(i,2)})$, and if N is large enough, adversary $\mathcal{A}$ can decide the correct value of $\pi_j(F_f(K', N_{(i,1)}) \oplus F_f(K', N_{(i,2)}))$ by using a majority vote.

Using Chernoff bounds, we deduce that the probability of adversary $\mathcal{A}$ obtaining the correct value of $\pi_j(F_f(K', N_{(i,1)}) \oplus F_f(K', N_{(i,2)}))$ in more than $\frac{N}{2}$ samples is larger than

$$1 - \mathsf{exp}(\frac{-2N\epsilon^2}{1+2\epsilon})$$

Where $\epsilon = \frac{1}{16}$.

Finally, the linearized set of equations $\pi_j(F_f(K', N_{(i,1)}) \oplus F_f(K', N_{(i,2)}))$ contain exactly 4×64 linear monomials and 6×64 monomials of degree 2. As a consequence, adversary $\mathcal{A}$ must get 640 correct equations $\pi_j(v_{(i,1)} \oplus v_{(i,2)}) = \pi_j(F_f(K', N_{(i,1)}) \oplus F_f(K', N_{(i,2)}))$ to recover the key K'. This happens with probability greater than

$$(1 - \mathsf{exp}(\frac{-2N\epsilon^2}{1+2\epsilon}))^{640}$$

Since there are $q = 60$ rounds in one execution of the protocol, it follows that adversary $\mathcal{A}$ needs $\tilde{N} = 2^{33}\frac{N}{q}(1 - \mathsf{exp}(\frac{-2N\epsilon^2}{1+2\epsilon}))^{640}$ interactions with tag T to get a correct linearized system in the 640 monomials. Setting $N = 4096$, we obtain $\tilde{N} = 2^{39.09}$.

3.3.2 Authentication based on Symmetric Primitives

Contrary to lightweight primitives, symmetric cryptography provides the means for provable RFID security and privacy. Additionally, it can be put into practice without requiring tags to store a large amount of keying material or to perform very expensive computations, see (58, 147, 163).

Along these lines, Feldhofer et al. (58) proposed a mutual authentication protocol that relies on AES, cf. Fig. 3.4. A tag T in this protocol stores an internal state S_T that consists of a secret key K_T that it shares with the reader. To start the authentication, the reader sends a random number N_R. The tag then returns the encryption $c_T = \mathsf{Enc}_{K_T}(N_R||N_T)$ where N_T is a random number generated by the tag. The reader decrypts c_T using the secret key K_T, gets the plaintext $a||b$, then checks whether $a = N_R$. If so, the reader accepts the tag and completes the mutual authentication by sending a second ciphertext $c_R = \mathsf{Enc}_{K_T}(b||N_R)$ to the tag. The authors showed that 128-bit AES can be implemented in 3628 G.E., while requiring 992 clock cycles at 100 KHz frequency. This result was among the first to confirm that real RFID tags can perform symmetric challenge response protocols. However, this protocol assumes that

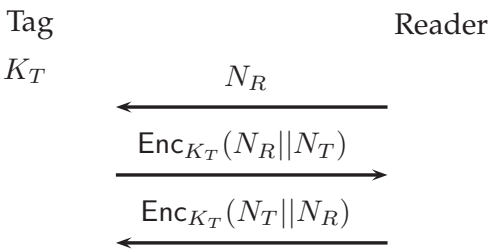

Figure 3.4: AES-based protocol

tags in the system share the same secret key K_T. As a result, the protocol only enables tag authentication but not tag identification, and if one tag is compromised, so is the other tags in the system. Also, the protocol is not *forward privacy preserving*, i.e., if a tag is corrupted by some adversary $\mathcal{A}$, $\mathcal{A}$ can easily link this tag to its previous interactions. Despite the fact that the first two problems can be solved by allowing tags to have different secrets, the problem of forward privacy is more difficult to mitigate.

One of the first protocols to address the problem of forward privacy was proposed by Ohkubo et al. (122). The authors designed a scheme called OSK that ensures forward privacy by equipping tags in the RFID system with two *one-way* hash functions H and G, where H is used to authenticate tags and G is used to update their secret keys. At initialization, each tag stores some secret key K_0 which is updated after each reading. Upon the i^{th} reader query, the tag computes first a reply $r = H(K_{i-1})$, then updates its secret key by evaluating $K_i = G(K_{i-1})$, and finally sends the reply r to the reader, as depicted in Fig. 3.5. When receiving the tag reply, the reader parses its database until it finds a match. If so, the reader updates the corresponding secret key using the one-way hash function G. It was shown in (122) that the OSK scheme is forward privacy preserving in the random oracle model. However, this protocol is not scalable and it is vulnerable to denial of service (DoS) attacks. An adversary $\mathcal{A}$ can query a tag T for l consecutive times, forcing the reader to perform a database search of complexity $O(ln)$ to identify T, where n is the number of tags in the system. Now, if l is too large, the reader may stall, hindering thus, the overall performance and availability of the RFID system. To tackle theses concerns, Avoine and Oechslin (6) presented a time-memory trade-off to reduce the computation load on the reader side. Still, OSK is not only prone to DoS but also to replay attacks. An adversary $\mathcal{A}$ can query a tag, then replay the tag's response to authenticate himself to the reader.

In the vein of OSK scheme, Berbain et al. (12) proposed a challenge response protocol that provides provable security and forward privacy. This protocol uses a hash function H and a random number generator G. A Tag in this scheme uses the hash function H to authenticate, and the random number generator G to update its internal state (i.e., secret key K_i). To improve the protocol performances, the hash function H is implemented as a family of *universal* hash functions which can be efficiently implemented in hardware, see

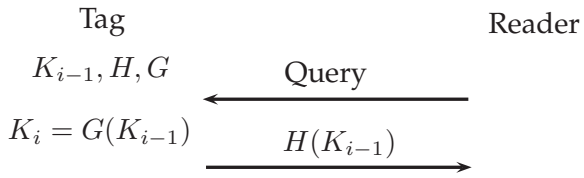

Figure 3.5: The OSK protocol

(35, 100). At initialization, each tag stores an initial key K_0 which is updated after each protocol invocation. The reader starts the protocol with some tag by sending a challenge message m. When receiving the reader query, the tag first generates a $k_1 + k_2$-bit random number $G(K_{i-1}) = K_{(i,1)}||K_{(i,2)}$, where $|K_{(i,1)}| = k_1$ and $|K_{(i,2)}| = k_2$, and sets its new key to $K_i = K_{(i,1)}$. Then, it picks a hash function $H_{K_{(i,2)}}$ from its family of universal hash functions using $K_{(i,2)}$ as index, and computes its reply $r = H_{K_{(i,2)}}(m)$, which it sends to the reader. This protocol can be efficiently implemented in 4000 $\mathsf{G.E.}$ as demonstrated by Berbain et al. (12), nonetheless, it is not scalable and it is susceptible to denial of service attacks just like OSK.

In an attempt to prevent DoS attacks, some schemes (33, 51) proposed that the reader authenticates itself at the end of the protocol execution, and that the tag updates its internal state only when the reader's authentication is successful. Despite the efficiency of such a counter-measure against DoS attacks, it fails at assuring forward privacy between two successful mutual authentications. As noted in (12), *it is impossible to assure simultaneously forward privacy and resistance to denial of service using only symmetric key cryptography*; either tags do not *always* refresh their states after each query and the scheme is then not forward privacy preserving, or they refresh their states after each query and the scheme is then vulnerable to DoS attacks. Yet, being prone to DoS attacks compromises privacy; an adversary can always recognize a tag which it queries too many times by observing whether the reader stall or not.

The above issues have led to work on efficient *linear/sublinear* protocols that still assure some level of privacy and security, cf. (42, 48, 50, 117). For instance, Dimitriou (50) presented a constant-time protocol for RFID mutual authentication, see Figure 3.6. Each tag stores a secret key K_{i-1} and computes a hash function H. The reader invokes the protocol by sending a nonce N_R to the tag. The tag computes first $H(K_{i-1})$, generates a random nonce N_T and evaluates $\mathsf{MAC}_{K_{i-1}}(N_T, N_R)$ using its secret key K_{i-1}. Finally, the tag replies with message $r = (N_T, H(K_{i-1}), \mathsf{MAC}_{K_{i-1}}(N_T, N_R))$. The reader identifies the tag using $H(K_{i-1})$ in constant time, then retrieves K_{i-1} and verifies the MAC. If the authentication succeeds, the reader computes the tag's new key K_i which it uses to evaluate $\mathsf{MAC}_{K_i}(N_T, N_R)$. The tag authenticates the reader, and if the authentication is successful, the tag updates its key. Since the tag sends the hash of its key $H(K_{i-1})$ in every protocol invocation, the tag is traceable until the next successful protocol execution.

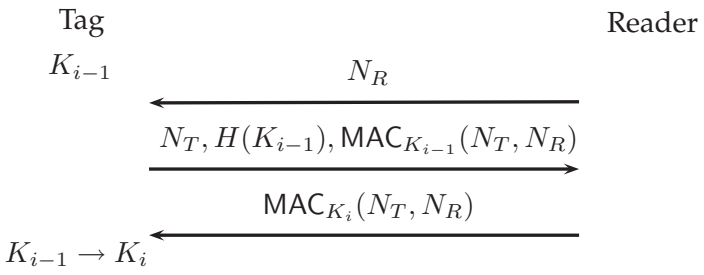

Figure 3.6: Dimitriou's protocol

Also, Molnar et al. (117) presented a scheme that achieves authentication in logarithmic time by organizing tags' secrets in a tree where each node is mapped to some secret key. Each tag in this scheme is associated with a leaf in the tree, and it is assumed to store all the keys $K_1, K_2, ..., K_d$ along the path from the root of the tree to its corresponding leaf. When a tag is queried with a nonce N_R, it replies with $N_T, F_{K_1}(N_T, N_R), F_{K_2}(N_T, N_R), ..., F_{K_d}(N_T, N_R)$, where N_T is a random number generated by the tag and F is a pseudorandom function. Using the values transmitted by the tag, the reader identifies the path leading to the tag in logarithmic time[3]. In spite of the apparent efficiency of (117), the reliance on correlated keys to speed up the authentication procedure affects the privacy of tags; if an adversary $\mathcal{A}$ learns the secrets of one tag, he also learns the secrets of other tags.

Furthermore, Damgård and Pedersen (42) proved the intuitive result that was already indicated in (92) which states that *"any complete, sound and strongly privacy preserving (according to (92)) symmetric RFID system requires the reader to perform a linear search in its database, in order to identify and authenticate tags"*. Thus, the authors suggested limiting the number of tags that an adversary can corrupt in order to assure soundness, privacy and efficiency.

Finally, Di Pietro and Molva (48) proposed an RFID protocol called DPM that combines lightweight identification with symmetric authentication. The idea is to use a lightweight primitive (bitwise operations) to identify the tag first, then to use a keyed hash function for authentication. Consequently, the overall computational performances of the reader are improved. In each protocol execution, the reader is only required to perform binary operations and to compute one keyed hash function. However, Soos (150) found a key-recovery attack against the lightweight primitive of DPM.

The efficiency limitations of symmetric cryptography spurred interest in the use of public key cryptography in RFID environment, particularly *elliptic curve cryptography* so as to achieve constant time RFID authentication while protecting tag security and privacy. The challenges in using public key cryptography are keeping the computation load and the storage requirements on tags reasonable. Hence, most of the work on RFID public key authentication

[3]It is noteworthy that this protocol is very similar in principle to the tree walking algorithm used for tag singulation.

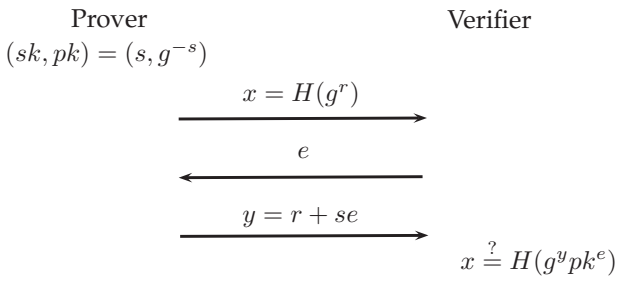

Figure 3.7: The GPS protocol

focused not only on proving privacy and security but also on implementation feasibility.

3.3.3 Authentication based on Asymmetric Primitives

One of the first public key solutions for RFID authentication was introduced by McLoone and Robshaw (113). This scheme relies on an elliptic curve variant of the GPS identification protocol proposed in (67, 68), cf. Fig. 3.7. Elliptic curve GPS is a three round protocol between a prover P and a verifier V. First, prover P and verifier V agree on an elliptic curve $\mathcal{E}$, and on a base point g in $\mathcal{E}$ of order q. Then, the identity of prover P is mapped to a pair of secret and public keys $(\mathsf{sk}, \mathsf{pk}) = (s, g^{-s})$, where s is picked randomly in $\mathbb{Z}_q$. In the first round of the protocol, prover P chooses a random number $r \in \mathbb{Z}_q$, computes g^r, then outputs its reply $x = H(g^r)$, where H is a cryptographic hash function. Verifier V picks randomly a challenge message e that it sends back to P, who responds with $y = r + se$. Finally, verifier V checks P's identity by verifying whether $H(g^y \mathsf{pk}^e) = x$. Now, RFID tag authentication ("identification") proceeds in the same manner, where a tag plays the role of the prover P and the reader plays the role of the verifier V. The tag however does not compute x, but rather it stores a set of pre-computed *coupons* $(r_i, x_i = H(g^{r_i}))$ which are used *only once*. When queried, the tag sends x_i, and upon receiving the challenge message e_i, it computes $y_i = r_i + se_i$. As a result, the tag is only required to execute arithmetic operations in $\mathbb{Z}$. Moreover, the authors showed that for $|q| = 160$ bits and $|e_i| = 32$ bits, their protocol can fit an area of 1642 G.E. while requiring 401 clock cycles, at 100 Khz. The GPS protocol however suffers from the following limitations. **1.)** It requires the reader to perform a search of a linear complexity in the number of tags, **2.)** it is not *forward privacy* preserving, and **3.)** it is prone to DoS attacks: if tags store l coupons, then tags can only respond to l queries and an adversary $\mathcal{A}$ can easily render a tag inoperative by querying it l times.

Notice that the last limitation of GPS can be tackled if tags are assumed to be able to execute elliptic curve operations. To validate this assumption, Kumar and Paar (102) and Lee et al. (104) investigated the feasibility of elliptic curve cryptography in low cost tags, and showed encouraging implementation results of elliptic curve processors. In fact, Kumar and Paar (102) implemented elliptic curve operations over a finite field of size 193 bits in an area

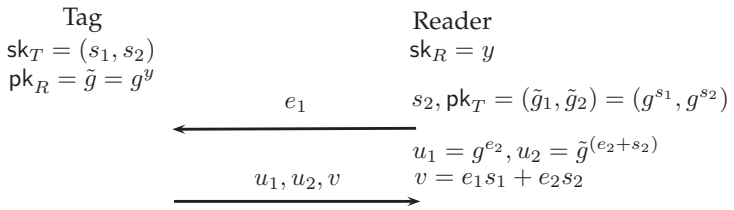

Figure 3.8: The EC-RAC protocol

of $18K$ G.E., whereas Lee et al. (104) implemented elliptic curve operations over a finite field of size 163 bits within an area of $15K$ G.E.

In line with these results, Lee et al. (103) proposed EC-RAC, a public key authentication protocol that is inspired from Schnorr's (143) and Okamoto's (123) identification schemes. Contrary to (143) and (123), EC-RAC is claimed to be secure and privacy preserving against active adversaries under the hardness of the discrete logarithm problem. Each tag T in this scheme is associated with a secret key $\mathsf{sk}_T = (s_1, s_2)$ and a "public key"[4] $\mathsf{pk}_T = (\tilde{g}_1, \tilde{g}_2) = (g^{s_1}, g^{s_2})$, as illustrated in Fig. 3.8. Whereas, the reader is associated with a pair of keys $(\mathsf{sk}_R, \mathsf{pk}_R) = (y, \tilde{g} = g^y)$. The reader starts the protocol by sending a challenge message e_1 to tag T, the tag picks a random number e_2 and computes $u_1 = g^{e_2}, u_2 = \tilde{g}^{e_2+s_2}$ and $v = e_1 s_1 + e_2 s_2$. When receiving (u_1, u_2, v), the reader identifies the tag by computing $\tilde{g}_2 = g^{s_2} = \frac{(u_2)^{\frac{1}{y}}}{u_1}$, then accepts the tag if $\tilde{g}_1 = (\frac{g^v}{u_1^{s_2}})^{\frac{1}{e_1}}$. The EC-RAC protocol has been implemented in $17K$ G.E., and executed within 500 ms at 500 KHz. Nevertheless, Bringer et al. (30) presented two attacks against EC-RAC. The first attack enables an adversary $\mathcal{A}$ to compute the value of $g^{\frac{1}{s_2}}$ from two protocol transcripts of the same tag, i.e., if adversary $\mathcal{A}$ eavesdrops on other protocol executions of this tag, he can easily track it by using the value $g^{\frac{1}{s_2}}$. The second attack allows an adversary $\mathcal{A}$ who eavesdrops on the same tag three times, to impersonate this tag as many times as he wants. To circumvent the above attacks, Lee et al. (105) proposed a revision of EC-RAC, yet van Deursen and Radomirovic (158) presented a man in the middle attack that allows a non-narrow adversary in the sense of (159) to track tags. In another attempt to resist man in the middle attacks, Lee et al. (106) proposed a third protocol which is EC-RACIII. Still, Fan et al. (57) demonstrated that EC-RACIII is as well vulnerable to tracking attacks that are conducted by a non-narrow adversary.

While most of the work on RFID public key authentication relies on elliptic curve cryptography as the underpinning technique, other approaches were proposed which are based on finite field cryptography. We mention namely the work by Oren and Feldhofer (126) that builds upon a variant of the Rabin cryptosystem (133) which was introduced by Shamir (146). As depicted in Fig. 3.9, the reader sends a random nonce N_R to the tag. The tag then generates two random numbers $N_{(T,1)}$ and $N_{(T,2)}$, together with a plaintext $m = f(N_R, N_{(T,1)}, \mathsf{ID})$,

[4]The public keys of tags in both GPS and EC-RAC are only known to authorized readers.

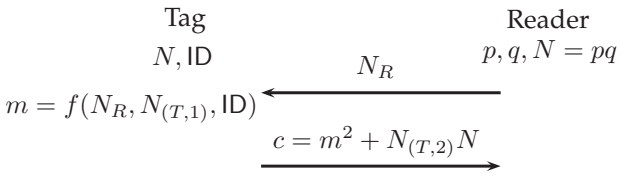

Figure 3.9: RFID authentication protocol based on Rabin cryptosystem

where f is a simple byte interleaving function, and ID is the tag's identifier. Finally, the tag computes the ciphertext $c = m^2 + N_{(T,2)}N$, where N is the reader's public key. When receiving the tag's reply c, the reader uses its Rabin's secret key (i.e., N's factorization) to decrypt c. There are 4 possible decryptions m_i for ciphertext c. As a consequence, the reader verifies first whether one of the resulted m_i contains the string N_R, if so the reader retrieves the identifier ID, and authenticates the tag by checking whether there is an entry in its database that corresponds to the identifier ID. The authors showed that this protocol can be implemented in $5K$ G.E.; this efficiency is due to the fact that the tag is only required to perform arithmetic operations in $\mathbb{Z}$. Note that on the one hand, the privacy of this protocol relies on the non-disclosure of public key N. On the other hand, if an adversary $\mathcal{A}$ corrupts a tag, he can easily retrieve the public key N. After the disclosure of public key N, adversary $\mathcal{A}$ still cannot compute the tag ID which is encrypted using the Rabin scheme. Nevertheless, the ciphertext c sent in the last round of the protocol can leak information about the tag, since the Rabin encryption is not IND-CPA.

Also, Paise and Vaudenay (129) presented a mutual authentication protocol, see Fig. 3.10 based on public key encryption, and showed that if the underlying encryption is IND-CPA, then the authentication protocol is narrow strong privacy preserving according to (129, 159). They also proved that if the encryption is IND-CCA (cf. Definition 2.17) then, the authentication protocol is also forward privacy preserving.

Among the known IND-CPA encryption schemes that could serve as the encryption in the Paise and Vaudenay's protocol, there is elliptic curve Elgamal (52). As EC-RAC, Elgamal only requires two exponentiations which was proven to be feasible in RFID environment, see (103). However, when using elliptic curve Elgamal, the tag has to first map the plaintext m to be encrypted into a point $\tilde{g}$ in the elliptic curve, and then encrypts the point $\tilde{g}$ to get a ciphertext c, whereas the reader has to decrypt c and invert the point mapping to get the plaintext m. As for now, there are few efficient invertible point mapping schemes, see (3), and it is still unknown if they are feasible in RFID tags. Moreover, the other IND-CPA encryptions that operate in $\mathbb{Z}_N$ are unsuitable for RFID tags. The same problem of point mapping arises when using Elliptic curve variants of IND-CCA encryptions such as Cramer-Shoup (41) to ensure forward privacy.

Although public key cryptography may allow for scalable RFID authentication protocols, the question of constructing efficient and provably secure and privacy preserving public key

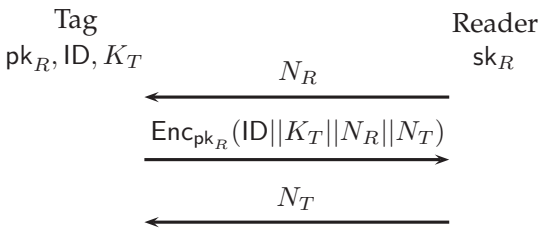

Figure 3.10: RFID authentication protocol based on public key cryptography

protocols remains open. As shown in this section, the provably secure and privacy preserving protocols require RFID tags to perform expensive computations that slacken the overall system, while the practical schemes have been proven to be vulnerable to tracking attacks.

3.3.4 Physical Layer Techniques

RFID authentication protocols that build upon the physical characteristics of the RFID environment can be classified into two categories. The first category exploits the properties of the wireless channel called *channel impairments* to secure the RFID communication against eavesdroppers. While the second category exploits the physical characteristics of RFID tags themselves to implement an alternative to tamper-resistance. The idea is to use the inherent variability of the wire and gate delays – which are unique to every single integrated circuit (IC) – to evaluate a pseudo-random function called *physically unclonable function* (abbreviated PUF), which is then used to securely identify tags.

3.3.4.1 Channel Impairment-based Protocols

Channel impairments are the physical factors such as interference, fading, shadowing ... etc, that result in the degradation of the quality of transmission. Schemes such as (36, 37, 93) take advantage of interference for instance, to make the reader's channel far better than the eavesdroppers' channel. In fact, Juels et al. (93) introduced the concept of the *blocker tag* which as discussed in Section 3.1.4 prevents unauthorized scanning by jamming the wireless channel. Whereas, Castelluccia and Avoine (36) introduced the concept of *noisy tag* which contrary to the blocker tag does not aim to block unauthorized tag scanning, but rather, aims to allow the reader to securely share a secret key with any tag in its vicinity in the presence of eavesdroppers. The protocol by Castelluccia and Avoine (36) relies on two assumptions: **1.)** the noisy tag and the tag T in question reply simultaneously, **2.)** the channel is additive, i.e., when several tags reply simultaneously, the amplitude of the different bits get added. Now, to enable secure key exchange between tag T and the reader, the authors divide time into slots t_i, and in each time slot t_i, tag T and the noisy tag – which is controlled by the reader– have to send a single bit simultaneously. If the noisy tag and tag T send the same bit $b = 1$ ($b = 0$ resp.), then the reader and eavesdroppers get a symbol S_{11} (S_{00} resp.), and they

both know that tag T sent bit $b = 1$ ($b = 0$ resp.). Consequently, the reader discards such symbols. If the noisy tag and tag T send different bits, then the reader and the eavesdroppers observe a symbol S_{01}, however only the reader can retrieve the bit sent by tag T, as it knows the bit that was sent by the noisy tag. The protocol ends with a *reconciliation phase*, where the reader provides tag T with the relevant time slots, i.e., the slots where the reader got the symbol S_{01}. Hence, at the end of the protocol, tag T and the reader are able to share some secret key K that could be used either to establish a secure channel to authenticate the tag or to refresh the tag's identifier. Chabanne and Fumaroli (37) improved (36) by taking into account possible transmission errors, and they augmented the protocol with a feature for integrity verification. They also suggested enhancing the randomness of the shared secret key by applying a universal hash function to the string that the reader and the tag agreed on in the reconciliation phase. Despite the fact that such schemes offer a good solution against eavesdroppers, their latency increases with the rate of transmission errors. That is to say, in the presence of a high transmission error rate, the reconciliation phase may require many interactions between the tag and the reader before both parties agree on the same secret string.

3.3.4.2 Protocols based on PUF

A PUF is a challenge response circuit which on an input a returns an output σ which depends heavily on the physical parameters of the circuit. The main advantage of PUF is that the PUFs of two circuits that execute the same logical functionality produce different outputs when queried with the same challenge, which implies that a PUF's output can uniquely identify and authenticate a tag. Another advantage of PUFs is that any physical attack on the tag's circuitry cannot be carried out without changing the physical properties of the tag, and therewith, the output of the PUF. As a result, a PUF is suited for tamper detection applications. Moreover, it is believed that the output of the PUF is unpredictable which may enable tags to generate good randomness that is not expensive in hardware. It follows that the application scenarios of physically unclonable functions can be classified into three categories:

- Source of randomness (79): In this case, the challenge a is used as a seed to produce a "true" random number.
- Tamper resistance enforcement (79): In this scenario, the PUF is treated as a physical fingerprint of the tag. This is achieved by querying the PUF of a tag T with some challenge a, then storing the output σ of T's PUF in the reader's database. When the reader is presented with tag T, it sends the challenge a and records T's answer σ'. The reader accepts tag T only if $\sigma = \sigma'$.
- Privacy preserving tag authentication (21, 47, 76, 154): The basic idea of such protocols is that instead of storing one pair of PUF challenge and response per tag as in the

previous application scenario, the reader stores several pairs to avoid querying tags with the same challenge twice. This results in a trade-off between tag privacy and the size of the reader's database. For instance, the reader in (21, 154) is required to store a large database, which may not be always practical in the presence of a large number of RFID tags.

We note that in practice, the output of the PUF matches only probabilistically its expected value, and it varies considerably depending on the physical parameters of the environment surrounding the PUF's circuit. This in reality allows for PUF-based authentication only in a controlled environment whose physical conditions do not vary drastically from the conditions in which tag initialization occurred. In addition, Rührmair et al. (138) presented several modeling attacks on the current implemented PUFs that enable an adversary $\mathcal{A}$ to spoof a PUF, breaking thus the widely admitted assumption that PUFs cannot be "cloned".

3.4 On the Limitations of Tag Privacy

Most of the protocols that we presented so far aim at ensuring tag privacy at the *application layer*, however, Avoine and Oechslin (7) pointed out that the unlinkability of tags (i.e., resistance to tracking attacks) cannot be assured only by relying on cryptographic protocols at the application layer. Namely, a privacy preserving RFID authentication protocol does not prevent an adversary $\mathcal{A}$ from tracking tags by inferring information from the communication or the physical layer. For instance, Danev et al. (43) and Zanetti et al. (164) exploited the spectral features of the responses emitted by tags when subjected to reader signals to extract RFID physical-layer fingerprints that enable a reader to accurately identify individual tags of the same manufacturer and model. The authors suggested thereby to use these physical fingerprints to detect cloned products in the supply chain and to check the genuineness of RFID-enabled identity documents. It is evident that accurate physical fingerprints jeopardize tag privacy even if tag-reader communication is protected using cryptographic protocols, and thereby voids all the counter-measures that were suggested to ensure tag privacy at the application layer. Fortunately, an accurate physical layer identification requires a controlled environment where tags are in close proximity and at a fixed position (43) with respect to the reader, which is not always feasible by an adversary $\mathcal{A}$ aiming to track a tag. Consequently, it is still useful to ensure tag privacy at the application layer through cryptographic protocols.

Still, assuring privacy at the *application layer* in the constrained RFID setting turned out to be a very difficult task. The problem lies in the fact that the existing formalizations of tag privacy generally assume *a strong adversary against which privacy cannot be achieved using the limited resources on RFID tags*. As a result, we believe that designing privacy preserving RFID protocols calls for a weaker, but realistic adversarial model that captures the capabilities of a real world adversary and fits the computational limitations of RFID technology.

In the remainder of this manuscript, we consider an adversary $\mathcal{A}$ who can interact and tamper with tags' internal states, yet cannot monitor all of their interactions. This assumption can also be stated as follows: *that there is at least one protocol execution between tags and legitimate readers that is unobserved by adversary* $\mathcal{A}$. This is in fact compliant with the work of Ateniese et al. (3), Dimitriou (51), Lim and Kwon (111) and Sadeghi et al. (139). We argue that such an assumption is valid, given that in the real world, an adversary $\mathcal{A}$ cannot always monitor devices that are as ubiquitous and mobile as RFID tags.

Furthermore, we turn to multiparty protocols that involve more than one reader, extending thus the focus of our research beyond simple tag-reader authentication to implement privacy preserving applications for the supply chain, as will be shown in Part II.

3.5 Summary

In this chapter, we have investigated the privacy and the security challenges raised by RFID systems. We surveyed the most prominent security and privacy models and analyzed some of the solutions proposed for security and privacy in an RFID-enabled environment, while describing a key recovery attack on an RFID authentication protocol called F_f.

Throughout this survey of the state of the art, we have identified a gap between the formal privacy models and the proposed RFID protocols whose main purpose is to fit the stringent computational requirements of RFID tags. In order to bridge the gap between the theoretical privacy models and practical considerations, we suggest a more realistic adversary who does not monitor all of the tags' interactions. Under this assumption, we are able to propose secure and privacy preserving solutions for supply chain management that will be presented in subsequent chapters.

Part II

Multi-party Protocols for RFID-enabled Supply Chains

RFID-enabled Supply Chains

Applications, Privacy and Security

Introduction

A supply chain is defined as a network of *partners*, who can be *"retailers, distributors, transporters, storage facilities and suppliers that participate in the sale, delivery and production of a particular product"* (1). Whereas supply chain management is defined as *"the management and the control of all materials and information in the logistic process from the acquisition of raw materials to the delivery to end users"* (115). Thus, supply chain management aims primarily to trace the movement of products to circumvent production bottlenecks, reduce product shrinkage, and to improve supply chain responsiveness to product recalls.

However when products are only equipped with optical barcodes, this renders the simplest task such as inventory labor intensive and prone to human errors. On this ground, leading retailers such as Wal-Mart and the US DoD (115, 152) endorsed the adoption of RFID technology at the pallet level to improve supply chain performances. The main advantage of RFID technology is the possibility to identify individual products without line of sight. This property enables supply chain partners to track individual products and log their history in a timely fashion without human intervention. Accordingly, it is admitted that the use of RFID tags in the supply chain is of a paramount business value as it enhances supply chain visibility which favors the regulation of production rate, counterfeit detection, enforcement of safety regulations, and targeted product recalls.

Yet, the pervasiveness of RFID technology facilitates denial of service attacks and industrial espionage as explained in Section 3.1.4. While denial of service can be tackled by increasing the physical security near RFID tags, privacy concerns are more challenging to address. Actually, tag privacy should not only be ensured against eavesdroppers (outsiders) but also against partners in the supply chain. That is, a supply chain partner must not be able to track tags that are not in his site. This privacy requirement calls for innovative solutions that rely on cryptography while taking into account the limited resources of RFID tags. Namely, any privacy preserving solution for supply chain applications has to be **1.)** *efficient*, so it does not slacken the overall performances of the supply chain, and **2.)** *implementable* in

passive tags (ideally, storage (read/write) only tags), in order not to burden the supply chain financially. Now to design supply chain applications that are *cheap*, *efficient* and *privacy preserving*, we relax the existing privacy models and we assume that *an adversary $\mathcal{A}$ cannot continuously monitor all of the tags' interactions in the supply chain*, as discussed in Section 3.4. We believe that such an assumption is fair given the distributed and the heterogeneous nature of supply chains.

By assuming a weaker yet a *realistic* adversary, we are able to design **1.)** a privacy preserving ownership transfer protocol that takes constant time while tags only compute hash functions, cf. Chapter 4, **2.)** two protocols for storage only tags that address the problem of genuineness verification of products traveling in the supply chain, see Chapter 5, and finally **3.)** a protocol for the automation of safety inspection using again storage only tags, cf. Chapter 6.

Supply Chain Requirements

Let π denote a protocol that implements a supply chain application. Without loss of generality, we assume that π outputs a bit $b \in \{0, 1\}$ after its execution. $b = 1$, if π's execution was successful; otherwise $b = 0$.

- **Security:** Loosely speaking, a protocol π is said to be secure if it is:
  - ***Complete:*** If protocol π is executed by legitimate parties, then π will output $b = 1$, meaning that the protocol execution was successful.
  - ***Sound:*** This property ensures that if an execution of protocol π outputs $b = 1$, then this implies that π was executed by legitimate parties with an overwhelming probability.

- **Privacy:** To ensure the privacy of tags, and hereby the privacy of partners in the supply chain, the protocol π has to fulfill the following two requirements.
  - ***Content privacy:*** An adversary must not be able to learn the confidential content of tags by querying them.
  - ***Location privacy:*** This property corresponds to the resistance to tracking attacks. Namely, a partner in the supply chain must not be able to trace tags that are not in his site. In this thesis, location privacy is captured by the ability of an adversary to tell tags apart based on their protocol executions.

  We note that location privacy is a stronger requirement than content privacy. In fact, if an adversary is able to disclose the private content of a tag, then he can easily track tags and violate the requirement of location privacy. Therefore, in the remainder of this

thesis, we focus on location privacy that we call hereafter tag unlinkability, and which we formalize using an indistinguishability-based definition as in (5, 92).

Target Applications

In this manuscript, we target the following supply chain applications for which we propose efficient, secure and privacy preserving solutions.

- **Tag ownership transfer:** For privacy reasons, each partner in the supply chain requires to *own* tags that are present in his site, i.e., to be the only entity that *identifies* and *authenticates* tags in his vicinity. When passing tags on to the next partner in the supply chain, ownership of tags has to be *transferred* to the new *owner*. Hence, tag ownership transfer is defined as the action of providing a new *tag owner* with the necessary information that enables him to authenticate a tag later on. The real challenge when devising tag ownership transfer protocols is to assure the privacy of tags against their previous owners and their new owners. Roughly speaking, when the ownership of some tag T is transferred from one partner P_i to another partner P_{i+1}, it must be computationally infeasible for P_i to trace T's future interactions, whereas partners P_{i+1} must not be able to link tag T to its past interactions, cf. Chapter 4.

- **Genuineness verification:** To verify the genuineness of products in the supply chain, one solution consists of verifying the path that a product took. The idea is to label each product in the supply chain with a tag that encodes the path that the product took so far. However, using RFID tags to detect counterfeits raises two challenges. The first is with respect to security, partners in the supply chain should be able to update the states of RFID tags but they must not be able to inject fake products. The second challenge regards privacy, a partner in the supply chain must not be able to trace tags once they leave his site, cf. Chapter 5.

- **Item matching:** One of the prominent applications of RFID technology is the automation of safety inspection when transporting hazardous items such as chemicals. The idea is to equip each chemical container with an RFID tag that encodes the type of the chemical. Now when two chemical containers C_i and C_j are in the range of some reader R in the supply chain, reader R reads the tags attached to C_i and C_j, and decides whether C_i and C_j can be stored close to one another or not. For safety reasons, RFID-based item matching has to be performed without revealing the private content of tags to readers in the supply chain. The only information that a reader learns after the execution of the protocol is whether a pair of tags match or not, cf. Chapter 6.

4

RFID-based Ownership Transfer with Issuer Verification

4.1 Introduction

As products travel in the supply chain, their ownership is transferred from one supply chain partner to another, and so is the ownership of their corresponding tags. Tag ownership in this setting is the capability that enables a partner in the supply chain to authenticate, access and transfer the ownership of tags that are present in his site, whereas tag ownership transfer is the action of transferring the necessary private information of some tag from one partner to another.

In order to protect the security and the privacy of tags and partners in the supply chain, a protocol for tag ownership transfer must ensure the following:

- *Secure mutual authentication* between tags and their owners (supply chain partners).
- *Exclusive ownership:* Non-authorized parties must not be able to transfer the ownership of a tag without the *consent* of the tag's owner.
- *Backward unlinkability:* A previous owner of a tag must not be able to trace a tag once he *releases* its ownership.
- *Forward unlinkability:* A new owner of a tag must not be able to link the tag to its *past* interactions.

Moreover, tag ownership transfer protocols are required to be efficient so as not to slacken the performances of the supply chain. Thus, a tag ownership transfer protocol must be built upon an efficient authentication protocol that takes into account the constrained computational resources of RFID tags: as discussed earlier, it is assumed that RFID tags can at best implement symmetric primitives such as hash functions. Yet, most symmetric authentication schemes require a linear search in the number of tags in the supply chain. We remind the

reader however that previously proposed symmetric authentication protocols are designed to be privacy preserving against a *strong* adversary who can *continuously* eavesdrop on tags' communications. As discussed in Section 3.4, we believe that such an adversary is unrealistic as it does not fit the limitations of RFID tags and the distributed and heterogeneous settings of supply chains. As a result, we believe that in order to design efficient tag ownership transfer protocols, we have to relax the privacy requirements by assuming that there is at least *one interaction* between a tag and its owner that is *unobserved* by the adversary.

To answer to the above privacy and security requirements, we introduce ROTIV, which in addition to the basic features of tag ownership transfer offers *issuer verification*. That is, any partner in the supply chain can verify the "issuer" (origin) of tags he owns. Such a feature impedes partners in the supply chain from injecting fake products that do not meet quality standards.

The main idea of ROTIV is to store in each tag T in the supply chain a symmetric key and an Elgamal encryption of its identifier signed by some trusted issuer. The public key encryption enables the owner to identify tags in constant time, while symmetric keys are used to mutually authenticate tags and owners. Also, each tag T in ROTIV is associated with a set of ownership references. T's ownership references allow T's owner to authenticate T and to transfer T's ownership. After each *successful* mutual authentication, the state of tag T and its ownership references must be updated in order to ensure both tag privacy and security. Finally, issuer verification of tag T is executed by checking whether the encrypted signature stored into tag T is a valid signature or not.

In summary, ROTIV's contributions are:

- *Constant time* mutual authentication while tags are only required to compute a hash function.
- Issuer verification that enables prospective owners of a tag T to check the identity of T's origin.
- Contrary to related work (60, 101, 117, 142), ROTIV does not require a trusted third party to perform tag ownership transfer.
- Formal definitions of privacy and security requirements of tag ownership transfer.
- Formal proofs of ROTIV security and privacy.

The sequel of this chapter is organized as follows: in Section 4.2, we introduce the notations that will be employed throughout this chapter, together with ROTIV's problem statement. In Section 4.3, we present the formal definitions that capture the security and the privacy requirements of tag ownership transfer. We move on to the protocol detailed description in Section 4.4, followed by a privacy and a security analysis in Section 4.5 and Section 4.6 respectively. Finally, we wrap-up the chapter by surveying some the previous work on tag ownership transfer in Section 4.7.

4.2 Background

An ownership transfer protocol involves the following entities:

4.2.1 Entities

- **Tags** T_i: Each tag is attached to a single product. A tag T_i has a re-writable memory representing T_i's current state $S_{T_i}^j$ at time j.

  In the remainder of this chapter, we denote $\mathcal{T}$ the set of tags T_i in the supply chain, and $n = |\mathcal{T}|$.

- **Issuer** I: The issuer I initializes tags and attaches each tag T_i to a product. At initialization I creates a set of *ownership references* denoted ref_{T_i} and writes an initial state $S_{T_i}^0$ into T_i. Finally, tag T_i and its ownership references are given to T_i's first owner denoted $O_{(T_i,1)}$.

- **Owner** $O_{(T_i,k)}$: Is the k^{th} owner of tag T_i. Owner $O_{(T_i,k)}$ stores a set of ownership references ref_{T_i} that enables him to authenticate and transfer T_i's ownership.

  We denote $\mathcal{O}$ the set of all owners $O_{(T_i,k)}$ in the supply chain and $\eta = |\mathcal{O}|$. Without loss of generality, we assume that an owner $O_{(T_i,k)}$ consists of a database DB_k and an RFID reader R_k.

4.2.2 RFID Ownership Transfer with Issuer Verification

An ownership transfer protocol raises four major requirements:

- During daily operations, the owner $O_{(T,k)}$ of some tag T in the supply chain has to be able to perform a number of *mutual authentications* with tag T.

- Eventually, $O_{(T,k)}$ has to pass T to the next owner $O_{(T,k+1)}$ in the supply chain. Therefore, the owner $O_{(T,k)}$ and $O_{(T,k+1)}$ must securely *exchange* the *ownership references* of tag T.

- Before accepting tag T, it is preferable that the prospective owner $O_{(T,k+1)}$ verifies the origin of tag T, i.e., given the ownership references of tag T, the owner $O_{(T,k+1)}$ checks whether tag T was originally initialized by the trusted issuer I or not.

- Once the ownership of tag T is transferred, the new owner $O_{(T,k+1)}$ must securely *update* any *secrets* stored in T and the corresponding *ownership references*. In this manner, $O_{(T,k+1)}$ is the only entity that can authenticate tag T and transfer its ownership.

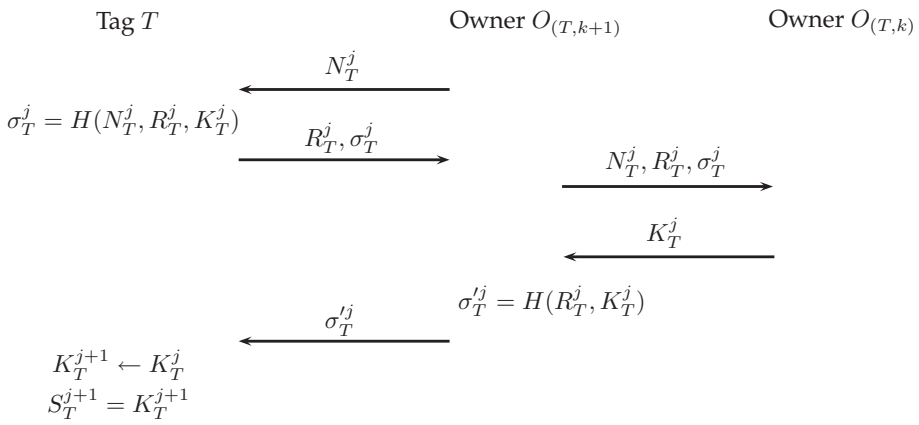

Figure 4.1: Ownership transfer protocol

4.2.3 Problem Statement

Recently proposed protocols on RFID tag ownership transfer (60, 111, 149) rely on symmetric primitives to perform privacy preserving mutual authentication and secure ownership transfer. As depicted in Figure 4.1, a tag T in these protocols:

- stores a state $S_T^j = K_T^j$. This state corresponds to a secret key which is shared between T and T's current owner $O_{(T,k)}$;
- computes a secure symmetric primitive H that is used to mutually authenticate T and $O_{(T,k)}$ using the secret key K_T^j;
- computes an update function G to refresh the secret key of T after a protocol execution.

However, such protocols suffer from inherent limitations:

- **Linear complexity:** As already explained in Section 3.3.2, a privacy preserving (in the sense of (92)) and secure symmetric tag authentication requires the owner of the tag to perform a linear search in his database to identify tags in his vicinity.
- **Denial of service:** To ensure forward unlinkability of tags, a tag is required to update its secret key using an update function G after each authentication. However such a mechanism makes the protocol prone to DoS attacks as explained in Section 3.3.2.
- **No tag issuer verification:** Without tag issuer verification, owners and therewith partners in the supply chain may be able to inject tags that were not issued by trusted parties. We claim that in the real world, the prospective owner of some tag T will require verifying the origin of T before accepting it.

We note that the previous ownership transfer protocols (60, 111, 149) are designed to be forward privacy preserving against a *strong adversary* that continuously monitors tags

in the supply chain(92, 129, 159). However, we show that by considering a more *realistic* adversary model, we can devise an ownership transfer protocol that achieves both constant time authentication and denial of service resistance while tags are only required to compute hash functions. As proposed in Section 3.4, we assume that an adversary cannot continuously monitor a tag, i.e., there is at least *one communication* between the tag and its owner that is *unobserved* by the adversary.

4.3 Adversary Model

We assume that the communication channel between owners in the supply chain is secure.

Accordingly, an adversary $\mathcal{A}$ has only access to the wireless channel between tags and their owners in the supply chain.

Now, to capture the capabilities of an adversary $\mathcal{A}$ against ROTIV, we assume that there is a challenger $\mathcal{C}$ who provides $\mathcal{A}$ with access to the following oracles:

- $\mathcal{O}_{\mathsf{Tag}}(\mathsf{param})$: When queried with parameter param, the oracle $\mathcal{O}_{\mathsf{Tag}}$ returns a tag $T \in \mathcal{T}$ that satisfies parameter param (if there is any).

  We indicate that adversary $\mathcal{A}$ can query the oracle $\mathcal{O}_{\mathsf{Tag}}$ with any combination of disjunctions or conjunctions of parameters.

- $\mathcal{O}_{\mathsf{Owner}}(\mathsf{OID})$: When queried with owner identifier OID, the oracle $\mathcal{O}_{\mathsf{Owner}}$ returns the owner $O \in \mathcal{O}$ whose identifier is OID (if there is any).

- $\mathcal{O}_{\mathsf{Execute}}(T)$: When called with tag T, the oracle $\mathcal{O}_{\mathsf{Execute}}$ starts a complete authentication session between tag T and its current owner $O_{(T,k)}$. During this authentication, adversary $\mathcal{A}$ is allowed to eavesdrop and alter the messages exchanged between tag T and owner $O_{(T,k)}$.

  At the end of the protocol execution, oracle $\mathcal{O}_{\mathsf{Execute}}$ returns a session identifier sid, a protocol transcript tran, and two bits b_T and b_O such that $b_T = 1$ ($b_O = 1$ resp.) if tag T (owner $O_{(T,k)}$ resp.) successfully authenticates owner $O_{(T,k)}$ (tag T resp.); otherwise $b_T = 0$ ($b_O = 0$ resp.)

- $\mathcal{O}_{\mathsf{Transfer}}(T, \mathsf{from}, \mathsf{to})$: When called with tag T, the oracle $\mathcal{O}_{\mathsf{Transfer}}$ invokes an ownership transfer protocol of tag T between the parties from and to.

  At the end of the protocol execution, the oracle $\mathcal{O}_{\mathsf{Transfer}}$ returns a bit b such that $b = 1$, if the ownership transfer protocol was successful, and $b = 0$ otherwise.

- $\mathcal{O}_{\mathsf{Flip}}(\mathsf{T}_0, \mathsf{T}_1)$: When queried with a pair of tags T_0 and T_1, the oracle $\mathcal{O}_{\mathsf{Flip}}$ randomly chooses $b \in \{0, 1\}$ and returns tag T_b.

4.3.1 Privacy

Inspired by previous work on ownership transfer(51, 111), we formally define using games the two major privacy requirements of ownership transfer which are: *forward unlinkability* and *backward unlinkability*.

In the setting of *tag ownership transfer*, forward unlinkability ensures that when a *new* owner $O_{(T,k+1)}$ acquires T's secrets after a successful ownership transfer at time t_{k+1}, he still cannot tell whether T has participated in protocol runs at time $t < t_{k+1}$. On the other hand, backward unlinkability, ensures that when a *previous* owner $O_{(T,k)}$ releases the ownership of tag T at time t_{k+1}, he still cannot tell whether T is involved in interactions that occurred at time $t > t_{k+1}$ or not.

4.3.1.1 Forward Unlinkability

The forward unlinkability game captures the capabilities of an owner of some tag T who has to decide whether T was already involved in *previous* protocol executions.

We recall that in scenarios where authentication is implemented using symmetric primitives, the notion of *forward* unlinkability as defined by Avoine (5), Juels and Weis (92) is achievable but at the expense of the resistance to denial of service attacks, see Section 3.3.2. Consequently, we assume that there is at least one communication between a tag T and its *previous* owner that was unobserved by T's current owner. This assumption enables us to achieve *relaxed* forward privacy, constant-time authentication and resistance to denial of service attacks.

Algorithm 4.3.1: Learning phase of the forward unlinkability game

$\mathsf{T}_0 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_0)$;
$\mathsf{T}_1 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_1)$;
for $i := 1$ **to** r **do**
  $\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_0)$;
  $\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_1)$;
for $i := 1$ **to** s **do**
  $T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}'_i)$;
  for $j := 1$ **to** t **do**
    $\mathcal{O}_{\mathsf{Execute}}(T_i)$;
  $\mathcal{O}_{\mathsf{Transfer}}(T_i, O_{(T_i,k_i)}, \mathcal{A})$;
  for $j := 1$ **to** t **do**
    $\mathcal{O}_{\mathsf{Execute}}(T_i)$;

Our forward unlinkability game is indistinguishability based, see Section 3.2.3.1. An adversary $\mathcal{A}(r, s, t, \epsilon)$ has access to tags in two phases. In the learning phase, as depicted in Algorithm 4.3.1, adversary $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{Tag}}$ to get two challenge tags T_0 and T_1 for

which he can call the oracle $\mathcal{O}_{\mathsf{Execute}}$ for a maximum of r times.

In addition to T_0 and T_1, adversary $\mathcal{A}$ is provided with s tags T_i, for which he can run mutual authentications and acquire the ownership by calling the oracles $\mathcal{O}_{\mathsf{Execute}}$ and $\mathcal{O}_{\mathsf{Transfer}}$ respectively.

Algorithm 4.3.2: Challenge phase of the forward unlinkability game

// Challenger $\mathcal{C}$ runs a mutual authentication for T_0 and T_1 outside the range of $\mathcal{A}$
$\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_0)$;
$\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_1)$;
$\mathsf{T}_b \leftarrow \mathcal{O}_{\mathsf{Flip}}\{\mathsf{T}_0, \mathsf{T}_1\}$;
// Ownership of tag T_b is transferred to $\mathcal{A}$
$\mathcal{O}_{\mathsf{Transfer}}(\mathsf{T}_b, O_{(\mathsf{T}_b,k)}, \mathcal{A})$;
for $j := 1$ **to** r **do**
  $\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_b)$;
Output b';

In the challenge phase as depicted in Algorithm 4.3.2, challenger $\mathcal{C}$ runs a mutual authentication for tags T_0 and T_1 *outside* the range of the adversary $\mathcal{A}$. Then, challenger $\mathcal{C}$ calls the oracle $\mathcal{O}_{\mathsf{Flip}}$ with the tags T_0 and T_1. $\mathcal{O}_{\mathsf{Flip}}$ selects randomly $b \in \{0, 1\}$ and returns the tag T_b to $\mathcal{A}$, who then acquires the ownership of tag T_b by calling the oracle $\mathcal{O}_{\mathsf{Transfer}}$.

After the ownership transfer, adversary $\mathcal{A}$ runs up to r mutual authentications with tag T_b and outputs his guess b' for the bit b.

Adversary $\mathcal{A}(r, s, t, \epsilon)$ is said to win the forward unlinkability game if $b = b'$.

The advantage ϵ of adversary $\mathcal{A}$ in winning the forward unlinkability game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins}) - \frac{1}{2}$$

Definition 4.1 (Forward Unlinkability)**.** ROTIV *is said to ensure forward unlinkability,* **iff** *for any adversary $\mathcal{A}(r, s, t, \epsilon)$, the advantage ϵ in winning the forward unlinkability game is negligible.*

4.3.1.2 Backward Unlinkability

Vaudenay (159) showed that it is impossible to achieve backward unlinkability without public key cryptography on tags[5]. As a result, in order to achieve at least a slightly weaker notion of backward unlinkability, we add the assumption that a previous owner $O_{(T,k)}$ of tag T *cannot continuously* monitor T after releasing T's ownership. This has been previously suggested by, e.g., Dimitriou (51), Lim and Kwon (111).

The backward unlinkability game captures the capabilities of an adversary $\mathcal{A}$ who *releases* the ownership of a tag T *during* his attack and has to tell whether tag T is involved in future

[5] Vaudenay (159) has shown that *narrow* strong privacy implies key agreement.

Algorithm 4.3.3: Learning phase of the backward unlinkability game

$\mathsf{T}_0 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_0)$;
$\mathsf{T}_1 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_1)$;
// Ownership of tags T_0 and T_1 is transferred to $\mathcal{A}$
$\mathcal{O}_{\mathsf{Transfer}}(\mathsf{T}_0, O_{(\mathsf{T}_0,k_0)}, \mathcal{A})$;
$\mathcal{O}_{\mathsf{Transfer}}(\mathsf{T}_1, O_{(\mathsf{T}_1,k_1)}, \mathcal{A})$;
for $i := 1$ **to** r **do**
  $\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_0)$;
  $\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_1)$;
for $i := 1$ **to** s **do**
  $T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}'_i)$;
  $\mathcal{O}_{\mathsf{Transfer}}(T_i, O_{(T_i,k_i)}, \mathcal{A})$;
  for $j := 1$ **to** t **do**
    $\mathcal{O}_{\mathsf{Execute}}(T_i)$;
  $O_{(T_i,k_i+2)} \leftarrow \mathcal{O}_{\mathsf{Owner}}(\mathsf{OID}_i)$;
  $\mathcal{O}_{\mathsf{Transfer}}(T_i, \mathcal{A}, O_{(T_i,k_i+2)})$;
  for $j := 1$ **to** t **do**
    $\mathcal{O}_{\mathsf{Execute}}(T_i)$;

protocol transactions or not.

In the learning phase, cf. Algorithm 4.3.3, oracle $\mathcal{O}_{\mathsf{Tag}}$ selects randomly two tags T_0 and T_1. The ownership of these two tags is transferred to $\mathcal{A}$. $\mathcal{A}$ is allowed to run up to r mutual authentications with tags T_0 and T_1.

Oracle $\mathcal{O}_{\mathsf{Tag}}$ gives $\mathcal{A}$ also an additional s tags T_i. The ownership of tags T_i is transferred to $\mathcal{A}$, who can then perform up to t mutual authentications with these tags. Again, the ownership of each tag T_i is transferred to an owner $O_{(T_i,k_i+2)}$ chosen by adversary $\mathcal{A}$ through the oracle $\mathcal{O}_{\mathsf{Owner}}$. Now, adversary $\mathcal{A}$ can execute another t mutual authentications for tags T_i.

In the challenge phase as depicted in Algorithm 4.3.4, adversary $\mathcal{A}$ transfers the ownership of the challenge tags T_0 and T_1 to owners of his choice. Then, T_0 and T_1 run a mutual authentication with their respective owners *outside* the range of the adversary $\mathcal{A}$. The oracle $\mathcal{O}_{\mathsf{Flip}}$ queried with tags T_0 and T_1, chooses randomly $b \in \{0, 1\}$ and returns tag T_b to $\mathcal{A}$. Adversary $\mathcal{A}$ is allowed to execute r mutual authentications with tag T_b.

Finally, adversary $\mathcal{A}$ outputs his guess b' for the bit b. $\mathcal{A}$ is said to win the backward unlinkability game if $b = b'$.

The advantage ϵ of adversary $\mathcal{A}$ in winning the backward unlinkability game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins}) - \frac{1}{2}$$

Definition 4.2 (Backward Unlinkability)**.** ROTIV *is said to ensure backward unlinkability,*

Algorithm 4.3.4: Challenge phase of the backward unlinkability game

// Ownership of tag T_0 is transferred from $\mathcal{A}$ to new owner $O_{(\mathsf{T}_0,k_0+2)}$
$O_{(\mathsf{T}_0,k_0+2)} \leftarrow \mathcal{O}_{\mathsf{Owner}}(\mathsf{OID}_0)$;
$\mathcal{O}_{\mathsf{Transfer}}(\mathsf{T}_0, \mathcal{A}, O_{(\mathsf{T}_0,k_0+2)})$;
// Ownership of tag T_1 is transferred from $\mathcal{A}$ to new owner $O_{(\mathsf{T}_1,k_1+2)}$
$O_{(\mathsf{T}_1,k_1+2)} \leftarrow \mathcal{O}_{\mathsf{Owner}}(\mathsf{OID}_1)$;
$\mathcal{O}_{\mathsf{Transfer}}(\mathsf{T}_1, \mathcal{A}, O_{(\mathsf{T}_1,k_1+2)})$;
// Challenger $\mathcal{C}$ runs a mutual authentication for T_0 and T_1 outside the range of $\mathcal{A}$
$\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_0)$;
$\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_1)$;
$\mathsf{T}_b \leftarrow \mathcal{O}_{\mathsf{Flip}}\{\mathsf{T}_0, \mathsf{T}_1\}$;
for $j := 1$ **to** r **do**
  $\mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_b)$;
Output b';

iff *for any adversary* $\mathcal{A}(r, s, t, \epsilon)$*, the advantage* ϵ *in winning the backward unlinkability game is negligible.*

4.3.2 Security

A secure ownership transfer with issuer verification has to fulfill the following security requirements.

4.3.2.1 Mutual Authentication

A secure ownership transfer protocol must ensure that when a tag T runs a successful mutual authentication with an owner O, then this implies that O is T's current owner. Also, when an owner O runs a successful mutual authentication with some tag T in his vicinity, it yields that T is a legitimate tag.

Algorithm 4.3.5: Learning phase of the mutual authentication game

for $i = 1$ **to** r **do**
  $T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_i)$;
  for $i = 1$ **to** s **do**
    $\mathcal{O}_{\mathsf{Execute}}(T_i)$;
for $i = 1$ **to** r **do**
  $T'_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}'_i)$;
  for $i = 1$ **to** s **do**
    $\mathcal{O}_{\mathsf{Execute}}(T'_i)$;
    $\mathsf{Read}(T'_i)$;

Algorithm 4.3.6: Challenge phase of the mutual authentication game

$\mathsf{T}_c \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_c)$;
$(\mathsf{tran}, b_T, b_O) \leftarrow \mathcal{O}_{\mathsf{Execute}}(\mathsf{T}_c)$;

We define a mutual authentication game in accordance with Lim and Kwon (111), Vaudenay (159) and Paise and Vaudenay (129). This game proceeds in two phases. During the learning phase as depicted in Algorithm 4.3.5, an adversary $\mathcal{A}(r, s, \epsilon)$ queries the oracle $\mathcal{O}_{\mathsf{Tag}}$ to get r tags T_i. Adversary $\mathcal{A}$ is allowed to execute s mutual authentications for tags T_i. Also, adversary $\mathcal{A}$ is allowed to query the oracle $\mathcal{O}_{\mathsf{Tag}}$ to get r additional tags T_i'. Adversary $\mathcal{A}$ can execute s mutual authentications with tags T_i' and to read their internal states by calling the function Read.

In the challenge phase as depicted in Algorithm 4.3.6, adversary $\mathcal{A}$ first queries the oracle $\mathcal{O}_{\mathsf{Tag}}$ to get a challenge tag T_c. Then, he interacts with tag T_c by calling the oracle $\mathcal{O}_{\mathsf{Execute}}$, which returns the tuple $(\mathsf{tran}, b_T, b_O)$ at the end of the mutual authentication.

Adversary $\mathcal{A}$ is said to win the mutual authentication game if:

i.) $b_T = 1$ or $b_O = 1$;

ii.) the internal state of tag T_c was not read by adversary $\mathcal{A}$ in the learning phase;

iii.) adversary $\mathcal{A}$ is not the current owner of tag T_c;

iv.) the owner of tag T_c and T_c did not engage in a mutual authentication with the same transcript tran.

The advantage ϵ of adversary $\mathcal{A}$ in winning the mutual authentication game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 4.3 (Mutual Authentication)**.** ROTIV *is secure with respect to mutual authentication,* **iff** *for any adversary $\mathcal{A}(r, s, \epsilon)$, the advantage ϵ in winning the mutual authentication game is negligible.*

4.3.2.2 Exclusive Ownership

Exclusive ownership ensures that an adversary $\mathcal{A}$ who does not have the ownership references ref_T of some tag T cannot transfer the ownership of T, even if he reads the internal state of tag T.

In the learning phase as shown in Algorithm 4.3.7, the oracle $\mathcal{O}_{\mathsf{Tag}}$ supplies $\mathcal{A}(r, s, \epsilon)$ with r tags T_i, then the ownership of tags T_i is transferred to adversary $\mathcal{A}$. $\mathcal{A}$ can run up to s mutual authentications with T_i by calling the oracle $\mathcal{O}_{\mathsf{Execute}}$. He can as well transfer the ownership of tags T_i to owners $O_{(T_i, k_i+2)}$ of his choice, and then executes another s mutual authentications with tags T_i.

Algorithm 4.3.7: Learning phase of the exclusive ownership game

for $i := 1$ **to** r **do**
  $T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_i)$;
  $\mathcal{O}_{\mathsf{Transfer}}(T_i, O_{(T_i,k_i)}, \mathcal{A})$;
  for $j := 1$ **to** s **do**
    $\mathcal{O}_{\mathsf{Execute}}(T_i)$;
  $O_{(T_i,k+2)} \leftarrow \mathcal{O}_{\mathsf{Owner}}(\mathsf{OID}_i)$;
  $\mathcal{O}_{\mathsf{Transfer}}(T_i, \mathcal{A}, O_{(T_i,k_i+2)})$;
  for $j := 1$ **to** s **do**
    $\mathcal{O}_{\mathsf{Execute}}(T_i)$;

Algorithm 4.3.8: Challenge phase of the exclusive ownership game

$\mathsf{T}_c \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_c)$;
$\mathsf{Read}(\mathsf{T}_c)$;
$O_c \leftarrow \mathcal{O}_{\mathsf{Owner}}(\mathsf{OID})$;
$b \leftarrow \mathcal{O}_{\mathsf{Transfer}}(\mathsf{T}_c, \mathcal{A}, O_c)$;

In the challenge phase, cf. Algorithm 4.3.8, adversary $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{Tag}}$ that supplies $\mathcal{A}$ with a challenge tag T_c. Now, adversary $\mathcal{A}$ can read T_c's internal state by calling the function Read.

At the end of the challenge phase, $\mathcal{A}$ runs an ownership transfer protocol for tag T_c with a challenge owner O_c of his choice by calling the oracle $\mathcal{O}_{\mathsf{Transfer}}$. At the end of the ownership transfer protocol, $\mathcal{O}_{\mathsf{Transfer}}$ outputs a bit b such that: $b = 1$, if the ownership transfer was successful, and $b = 0$ otherwise.

$\mathcal{A}$ is said to win the exclusive ownership game, if **i.)** $b = 1$, and if **ii.)** adversary $\mathcal{A}$ is not the owner of tag T_c.

The advantage ϵ of adversary $\mathcal{A}$ in winning the exclusive ownership game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 4.4 (Exclusive Ownership)**.** ROTIV *is said to ensure exclusive ownership,* **iff** *for any adversary* $\mathcal{A}(r, s, \epsilon)$*, the advantage* ϵ *in winning the exclusive ownership game is negligible.*

4.3.2.3 Issuer Verification

The security of issuer verification ensures that when an owner O in the supply chain accepts a tag T, then this implies that tag T was originally issued by the trusted issuer I (with an overwhelming probability).

The goal of some adversary $\mathcal{A}$ is to convince an owner O_c in the supply chain to accept the ownership of a tag T that was not actually issued by I.

The security of issuer verification is defined by a security game that proceeds as follows. In the learning phase of the issuer verification game, adversary $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{Tag}}$ that gives $\mathcal{A}$ a total of r tags T_i whose ownership is then transferred to adversary $\mathcal{A}$. Now, adversary $\mathcal{A}$ can run up to s mutual authentications for tag T_i by calling the oracle $\mathcal{O}_{\mathsf{Execute}}$. He can also call the oracle $\mathcal{O}_{\mathsf{Transfer}}$ to transfer the ownership of tags T_i to owners $O_{(T_i,k_i+2)}$ of his choice.

Algorithm 4.3.9: Learning phase of the security game of issuer verification

for $i := 1$ **to** r **do**
  $T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_i)$;
  $\mathcal{O}_{\mathsf{Transfer}}(T_i, O_{(T_i,k_i)}, \mathcal{A})$;
  for $j := 1$ **to** s **do**
    $\mathcal{O}_{\mathsf{Execute}}(T_i)$;
  $O_{(T_i,k_i+2)} \leftarrow \mathcal{O}_{\mathsf{Owner}}(\mathsf{OID}_i)$;
  $\mathcal{O}_{\mathsf{Transfer}}(T_i, \mathcal{A}, O_{(T_i,k_i+2)})$;

Algorithm 4.3.10: Challenge phase of the security game of issuer verification

$\mathsf{CreateTag}\ \mathsf{T}_c$;
$O_c \leftarrow \mathcal{O}_{\mathsf{Owner}}(\mathsf{OID}_c)$;
$b \leftarrow \mathcal{O}_{\mathsf{Transfer}}(\mathsf{T}_c, \mathcal{A}, O_c)$;

In the challenge phase, $\mathcal{A}$ creates a new tag $\mathsf{T}_c \notin \mathcal{T}$ (i.e., T_c is not a clone of some other tag). Then, adversary $\mathcal{A}$ transfers the ownership of tag T_c to some challenge owner O_c of his choice. At the end of the ownership transfer, the oracle $\mathcal{O}_{\mathsf{Transfer}}$ returns a bit b.

Adversary $\mathcal{A}$ is said to win the issuer verification game, if $b = 1$.

The advantage ϵ of adversary $\mathcal{A}$ in winning the security game of issuer verification is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 4.5 (Issuer Verification Security)**.** ROTIV *is secure with respect to issuer verification,* **iff** *for any adversary $\mathcal{A}(r, s, \epsilon)$, the advantage ϵ in winning the security game of issuer verification is negligible.*

Remark 4.1. *Issuer verification assures that an adversary cannot create and inject new tags into the supply chain. Yet, the owner of a legitimate tag T can clone T to obtain new tags that pass issuer verification. We believe that without tamper resistance or protected memory mechanisms, cloned tags will always pass issuer verification. Fortunately, cloning can be detected if tags have unique identifiers.*

4.4 ROTIV

In order to ensure tag privacy and enable issuer verification while keeping the storage requirements in tags *minimal*, each tag T in the supply chain is required to store a secret key K_T and a *short* signature (22) of its identifier encrypted using elliptic curve Elgamal.

4.4.1 Preliminaries

In this section, we describe briefly the short signature scheme used in ROTIV to sign tags' identifiers and Elgamal cryptosystem.

4.4.1.1 Short Signature

The short signature used in ROTIV consists of the following operations.

- **Key generation:** On input of a security parameter τ, the system obtains a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, H)$ where:
  - $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups such that $\mathbb{G}_1$ and $\mathbb{G}_T$ have the same prime order q;
  - g and h are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively;
  - $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an asymmetric bilinear pairing, see Section 2.3.3, Definition 2.5;
  - H is a secure hash function from $\{0,1\}^* \rightarrow \mathbb{G}_1$. This hash function will be viewed as a random oracle in the rest of this chapter.

  The system then picks up a random number x from $\mathbb{F}_q^*$. Now, the signature secret key is $\mathsf{sk} = x$, and the corresponding public key is $\mathsf{pk} = h^{\mathsf{sk}} \in \mathbb{G}_2$.
- **Signing:** On input of a message m and secret key sk, this algorithm outputs $\mathcal{S} = \mathsf{Sign}_{\mathsf{sk}}(m) = H(m)^{\mathsf{sk}}$.
- **Verification:** On input of a message m, a signature $\mathcal{S}$ and public key pk, this algorithm checks whether the following equation holds:

$$e(H(m), \mathsf{pk}) = e(\mathcal{S}, h) \tag{4.1}$$

  If so, it outputs $\mathsf{Verify}_{\mathsf{pk}}(m, \mathcal{S}) = 1$; otherwise it outputs $\mathsf{Verify}_{\mathsf{pk}}(m, \mathcal{S}) = 0$.

  Note that if $\mathcal{S} = H(m)^{\mathsf{sk}}$, then the Equation 4.1 will always hold.

Remark 4.2. *We note that the signature presented above is a modified variant of the scheme proposed by Boneh et al. (23). The difference between these two signatures lies on the fact that the scheme in (23) requires symmetric bilinear pairings and its security relies on the CDH assumption, whereas* ROTIV*'s signature requires asymmetric bilinear pairings and its security is based on the BCDH assumption as will be proven in Section 4.6.3.*

4.4.1.2 Elliptic Curve Elgamal Cryptosystem

An elliptic curve Elgamal cryptosystem provides the following usual set of operations:

- **Setup:** On input of a security parameter τ, the system outputs an elliptic curve $\mathcal{E}$ over a finite field $\mathbb{F}_p$. Let g be a point on $\mathcal{E}(\mathbb{F}_p)$ of a large prime order q such that the DDH problem is intractable in $\mathbb{G}_1 = \langle g \rangle$.
- **Key generation:** The secret key is $\mathsf{sk} \in \mathbb{F}_q^*$. The corresponding public key pk is the pair of points $(g, \tilde{g} = g^{\mathsf{sk}})$.
- **Encryption:** To encrypt a point $m \in \mathbb{G}_1$, one randomly selects $r \in \mathbb{F}_q$ and computes $\mathsf{Enc}_{\mathsf{pk}}(m) = (u, v) = (g^r, m\tilde{g}^r)$. The ciphertext is $c = (u, v)$.
- **Decryption:** To decrypt a ciphertext $c = (u, v)$, one computes $\mathsf{Dec}_{\mathsf{sk}}(c) = \dfrac{v}{u^{\mathsf{sk}}} = m$.

Remark 4.3. *Note that Elgamal cryptosystem is*

- *IND-CPA under the DDH assumption in* $\mathbb{G}_1$*;*
- *homomorphic, i.e.,* $\forall\ m_1, m_2 \in \mathbb{G}_1, \mathsf{Enc}(m_1)\mathsf{Enc}(m_2) = \mathsf{Enc}(m_1 m_2)$.

4.4.2 Protocol Overview

In ROTIV, a tag T stores a state $S_T^j = (K_T^j, c_T^j)$, where K_T^j is a shared key between tag T and its owner, and c_T^j is an Elgamal encryption of the signature of T's identifier by issuer I.

When an owner $O_{(T,k)}$ starts a mutual authentication with T, T replies with c_T^j and the MAC of c_T^j computed using T's secret key K_T^j. Upon receipt of the tag reply, owner $O_{(T,k)}$ uses his secret key to decrypt c_T^j. After decryption, $O_{(T,k)}$ checks if the resulting plaintext is in his database DB_k. If so, $O_{(T,k)}$ looks up the symmetric key K_T^j of tag T and verifies the MAC sent by T. Consequently, ROTIV enables mutual authentication in constant time, while tags are only required to compute symmetric primitives (i.e., MAC). After each successful authentication, the state of tag T is updated using Elgamal re-encryption techniques and key update mechanisms so as to ensure both forward and backward unlinkability.

Now to transfer the ownership of tag T, the current owner $O_{(T,k)}$ of T provides the prospective owner $O_{(T,k+1)}$ with the ownership references ref_T of tag T. These ownership references allow owner $O_{(T,k+1)}$ to first verify the issuer of tag T by checking whether the ciphertext c_T^j encrypts a valid signature by issuer I, then to authenticate himself to T and update the internal state of tag T.

4.4.3 Protocol Description

To ensure the privacy and the security of ROTIV, we employ bilinear groups where DDH is hard, see Definition 2.35 and Definition 2.36. More precisely, we prove the security and the

privacy of ROTIV by relying on the BCDH assumption (cf. Definition 2.32) and the XDH assumption (cf. Definition 2.35) respectively.

Remark 4.4. ROTIV *'s privacy can also rely on the SXDH assumption, see Definition 2.36.*

In the remainder of this section, we assume that each tag T can evaluate a cryptographic hash function G that is used to compute the MAC of tag T and to update its symmetric key after each successful authentication.

4.4.3.1 Setup

A trusted third party (TTP) outputs $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, h, e, H, G)$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are cyclic groups of prime order q, g and h are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an asymmetric bilinear pairing. $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ is a cryptographic hash function. To compute H, the different parties in ROTIV can use the hashing algorithm proposed by Brier et al. (28) that hashes into ordinary elliptic curves. $G : \{0,1\}^* \rightarrow \mathbb{F}_q$ is a cryptographic hash function used to compute MACs and to update the tags' keys.

The TTP chooses $x \in \mathbb{F}_q^*$ and computes h^x, and supplies issuer I with the secret key $\mathsf{sk}_I = x$ and the corresponding public key $\mathsf{pk}_I = h^x$.

Next, the TTP selects η random numbers $x_k \in \mathbb{F}_q^*$, computes $\tilde{g}_k = g^{x_k}$, and supplies each owner O_k in the supply chain with the secret key $\mathsf{sk}_k = x_k$, and the corresponding (Elgamal) public key $\mathsf{pk}_k = (g, \tilde{g}_k)$.

4.4.3.2 Tag Initialization

To initialize a tag T, issuer I picks a pair of random numbers $(K_T^0, \mathsf{ID}) \in \mathbb{F}_q \times \mathbb{F}_q$, computes the signature $\mathcal{S} = H(\mathsf{ID})^{\mathsf{sk}_I}$, and writes into tag T the state $S_T^0 = (K_T^0, c_T^0)$, where $c_T^0 = (1, \mathcal{S})$.

Finally, issuer I supplies owner $O_{(T,1)}$ with tag T and with T's ownership references:

$$\mathsf{ref}_T = (\mathcal{S}, \mathsf{id}, K^{\mathsf{old}}, K^{\mathsf{new}}, \mathsf{rand}^{\mathsf{old}}, \mathsf{rand}^{\mathsf{new}}) = (H(\mathsf{ID})^{\mathsf{sk}_I}, \mathsf{ID}, -, K_T^0, -, 1)$$

Without loss of generality, we assume that $O_{(T,1)} = O_1$.

Now, owner O_1 updates the state and the ownership references of tag T as follows: he chooses randomly $r^1 \in \mathbb{F}_q$ and computes an Elgamal encryption of $\mathcal{S}$ using his public key $\mathsf{pk}_1 = (g, \tilde{g}_1)$:

$$c_T^1 = (u_T^1, v_T^1) = (g^{r^1}, \mathcal{S}\tilde{g}_1^{r^1})$$

Then, he writes into T the state $S_T^1 = (K_T^1, c_T^1)$, and adds an entry E_T for tag T in his database DB_1:

$$\begin{aligned} E_T = \mathsf{ref}_T &= (\mathcal{S}, \mathsf{id}, K^{\mathsf{old}}, K^{\mathsf{new}}, \mathsf{rand}^{\mathsf{old}}, \mathsf{rand}^{\mathsf{new}}) \\ &= (H(\mathsf{ID})^{\mathsf{sk}_I}, \mathsf{ID}, K_T^0, K_T^1, 1, h^{r^1}) \end{aligned}$$

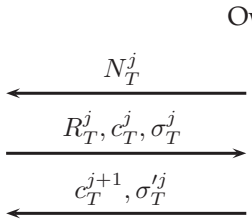

Figure 4.2: Authentication in ROTIV

4.4.3.3 Authentication Protocol

To authenticate a tag T, the current owner $O_{(T,k)}$ of tag T decrypts the ciphertext $c_T^j = (u_T^j, v_T^j)$ sent by tag T and gets $\mathcal{S}$. Using $\mathcal{S}$, $O_{(T,k)}$ identifies T and starts a MAC-based mutual authentication. If the mutual authentication succeeds, both owner $O_{(T,k)}$ and tag T update their keys. Without loss of generality, we assume that $O_{(T,k)} = O_k$.

1. To start an authentication with tag T, the owner O_k sends a random nonce N_T^j to T as depicted in Figure 4.2.

Once T receives N_T^j, it generates a random number $R_T^j \in \mathbb{F}_q$. Using its secret key K_T^j, T computes: $\sigma_T^j = \mathsf{MAC}_{K_T^j}(N_T^j, R_T^j, c_T^j)$.

2. T replies with $(R_T^j, c_T^j = (u_T^j, v_T^j), \sigma_T^j)$.

Upon receiving T's reply, the owner $O_{(T,k)}$ decrypts c_T^j using his secret key sk_k and gets $\mathcal{S} = \dfrac{v_T^j}{(u_T^j)^{\mathsf{sk}_k}}$. $O_{(T,k)}$ checks whether $\mathcal{S} \in \mathsf{DB}_k$. If not, $O_{(T,k)}$ aborts the authentication.

Otherwise, $O_{(T,k)}$ retrieves T's ownership references $\mathsf{ref}_T = (\mathcal{S}, \mathsf{id}, K^{\mathsf{old}}, K^{\mathsf{new}}, \mathsf{rand}^{\mathsf{old}}, \mathsf{rand}^{\mathsf{new}})$ in DB_k and checks whether:

$$\sigma_T^j = \mathsf{MAC}_{K^{\mathsf{new}}}(N_T^j, R_T^j, c_T^j) \text{ or } \sigma_T^j = \mathsf{MAC}_{K^{\mathsf{old}}}(N_T^j, R_T^j, c_T^j)$$

If not, $O_{(T,k)}$ aborts the authentication.

If $\mathsf{MAC}_{K^{\mathsf{old}}}(N_T^j, R_T^j, c_T^j) = \sigma_T^j$ then $K_T^j = K^{\mathsf{old}}$, otherwise $K_T^j = K^{\mathsf{new}}$.

Then, owner O_k chooses a new random number $r^{j+1} \in \mathbb{F}_q^*$ and computes:

$$\begin{aligned} c_T^{j+1} &= (u_T^{j+1}, v_T^{j+1}) = (g^{r^{j+1}}, \mathcal{S}\tilde{g_k}^{r^{j+1}}) \\ \sigma_T'^j &= \mathsf{MAC}_{K_T^j}(R_T^j, c_T^{j+1}) \end{aligned}$$

Finally, O_k updates the ownership references ref_T of tag T:

$$\begin{aligned} (K^{\mathsf{old}}, K^{\mathsf{new}}) &= (K_T^j, G(K_T^j, N_T^j, R_T^j)) \\ (\mathsf{rand}^{\mathsf{old}}, \mathsf{rand}^{\mathsf{new}}) &= (h^{r^j}, h^{r^{j+1}}) \end{aligned}$$

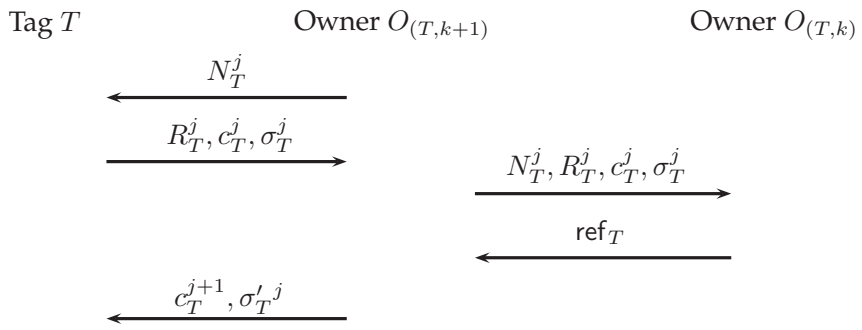

Figure 4.3: Ownership transfer in ROTIV

Where r^j and r^{j+1} are the random numbers used to compute the ciphertext c_T^j and c_T^{j+1} respectively.

3. Finally, owner O_k sends c_T^{j+1} and $\sigma_T'^j$ to T.

Once T receives $\sigma_T'^j$ and c_T^{j+1}, it checks if $\sigma_T'^j = \mathsf{MAC}_{K_T^j}(R_T^j, c_T^{j+1})$. If not, T aborts the authentication. Otherwise, T updates its internal state $S_T^{j+1} = (K_T^{j+1}, c_T^{j+1})$, where:

$$K_T^{j+1} = G(K_T^j, N_T^j, R_T^j)$$

Desynchronization If the last message of the authentication protocol is lost, tag T will not update its state, and as a result, it will not update its secret key K_T^j. However, as owner O_k keeps both keys $K^{\mathsf{old}} = K_T^j$ and $K^{\mathsf{new}} = G(K_T^j, N_T^j, R_T^j)$, owner O_k can always re-synchronize with T using K^{old}.

4.4.3.4 Ownership Transfer Protocol

The setup of ownership transfer in ROTIV consists of a current owner $O_{(T,k)}$, a prospective owner $O_{(T,k+1)}$ and a tag T as shown in Figure 4.3. The ownership transfer consists of: **i.)** a mutual authentication between T and $O_{(T,k+1)}$, **ii.)** an exchange of ownership references between $O_{(T,k)}$ and $O_{(T,k+1)}$ to perform issuer verification and to allow the authentication of $O_{(T,k+1)}$.

Without loss of generality, we assume that $O_{(T,k)} = O_k$ and that $O_{(T,k+1)} = O_{k+1}$.

The ownership transfer protocol between O_k and O_{k+1} for tag T proceeds as detailed below:

1. The owner O_{k+1} sends a nonce N_T^j to tag T.

2. T replies with $c_T^j = (u_T^j, v_T^j)$, a random number R_T^j and the MAC σ_T^j.

3. O_{k+1} sends N_T^j, R_T^j, c_T^j, σ_T^j to T's owner O_k.

Given N_T^j, R_T^j, c_T^j and σ_T^j, O_k authenticates T. If the authentication fails, O_k informs O_{k+1}. Otherwise, O_k supplies O_{k+1} with T's ownership references:

$$\begin{aligned} \mathsf{ref}_T &= (\mathcal{S}, \mathsf{id}, K^{\mathsf{old}}, K^{\mathsf{new}}, \mathsf{rand}^{\mathsf{old}}, \mathsf{rand}^{\mathsf{new}}) \\ &= (H(\mathsf{ID})_I^{\mathsf{sk}}, \mathsf{ID}, -, K_T^j, -, h^{r^j}) \end{aligned}$$

4. Provided with ref_T, O_{k+1} checks if the equation $\sigma_T^j = \mathsf{MAC}_{K_T^j}(N_T^j, R_T^j, c_T^j)$ holds. If it does, this implies that the key K_T^j provided by O_k corresponds to tag T.

Then, using the public key $\mathsf{pk}_I = h^{\mathsf{sk}_I}$ of issuer I, O_{k+1} verifies whether tag T was issued by I:

- First, O_{k+1} checks whether $e(H(\mathsf{id}), \mathsf{pk}_I) = e(\mathcal{S}, h)$ or not.
- Then, he verifies whether $e(u_T^j, h) = e(g, \mathsf{rand}^{\mathsf{new}})$ or not.
- Finally, he checks if ciphertext c_T^j encrypts the signature $\mathcal{S}$. This verification is performed by checking the following equation:

$$e(\mathcal{S}, h) = \frac{e(v_T^j, h)}{e(\tilde{g}_k, \mathsf{rand}^{\mathsf{new}})}$$

  Note that if c_T^j is the encryption of $\mathcal{S}$ using the public key pk_k, then $c_T^j = (u_T^j, v_T^j) = (g^{r^j}, \mathcal{S}\tilde{g}_k^{r^j})$, and therefore,

$$\begin{aligned} e(v_T^j, h) &= e(\mathcal{S}\tilde{g}_k^{r^j}, h) = e(\mathcal{S}, h)e(\tilde{g}_k^{r^j}, h) \\ &= e(\mathcal{S}, h)e(\tilde{g}_k, h^{r^j}) = e(\mathcal{S}, h)e(\tilde{g}_k, \mathsf{rand}^{\mathsf{new}}) \end{aligned}$$

  Also if c_T^j verifies the equations above, then c_T^j encrypts $\mathcal{S}$.

If the issuer verification fails, then O_{k+1} aborts the ownership transfer. Otherwise, O_{k+1} adds the entry ref_T into his database DB_{k+1}, and finishes its authentication as follows:

- First, owner O_{k+1} chooses a new random number $r^{j+1} \in \mathbb{F}_q^*$ and computes:

$$\begin{aligned} c_T^{j+1} &= (u_T^{j+1}, v_T^{j+1}) = (g^{r^{j+1}}, \mathcal{S}\tilde{g}_{k+1}^{r^{j+1}}) \\ \sigma_T'^j &= \mathsf{MAC}_{K_T^j}(R_T^j, c_T^{j+1}) \end{aligned}$$

  So, c_T^{j+1} is the encryption of $\mathcal{S}$ with the public key $\mathsf{pk}_k = (g, \tilde{g}_{k+1})$ of owner O_{k+1}.
- Then, O_{k+1} sends c_T^{j+1} and $\sigma_T'^j$ to T, and updates his database DB_{k+1} as in the authentication protocol presented above.
- Upon receiving c_T^{j+1} and $\sigma_T'^j$, T authenticates O_{k+1}. If the authentication succeeds T updates its state accordingly.

At the end of the ownership transfer, owner O_{k+1} queries tag T to check whether T has updated its state successfully or not. If not, owner O_{k+1} engages in mutual authentications with tag T until the latter updates its internal state.

Remark 4.5. *To prevent the old owner O_k of tag T from tracing T later in the future, the new owner O_{k+1} has to run a mutual authentication with T* outside *the range of O_k right after the ownership transfer. In this manner, tag T and owner O_{k+1} will share a symmetric key that O_k cannot retrieve without a physical access to the tag.*

4.5 Privacy Analysis

In this section, we prove that ROTIV is privacy preserving under the XDH assumption.

4.5.1 Forward Unlinkability

Theorem 4.1. ROTIV *ensures forward unlinkability under the XDH assumption.*

Proof. Assume that there is an adversary $\mathcal{A}(r, s, t, \epsilon)$ who succeeds in the forward unlinkability game with a non negligible advantage ϵ. We will now construct an adversary $\mathcal{B}$, who uses $\mathcal{A}$ as a subroutine and breaks the DDH assumption in $\mathbb{G}_1$ with a non-negligible advantage ϵ'.

Let $\mathcal{O}_{\mathsf{DDH}}$ be an oracle that when queried, selects first two random elements $x, y \in \mathbb{F}_q$ and flips a fair coin $b \in \{0, 1\}$. If $b = 1$, then $\mathcal{O}_{\mathsf{DDH}}$ sets $z = xy$; otherwise z is randomly selected from $\mathbb{F}_q$. Finally, it returns the tuple $(g, g^x, g^y, g^z) \in \mathbb{G}_1$.

To break the DDH assumption in $\mathbb{G}_1$, adversary $\mathcal{B}$ first queries the oracle $\mathcal{O}_{\mathsf{DDH}}$ to receive (g, g^x, g^y, g^z) and simulates a complete ROTIV system for $\mathcal{A}$.

- Adversary $\mathcal{B}$ selects randomly a random number $\mathsf{sk}_I \in \mathbb{F}_q$ and computes $\mathsf{pk}_I = h^{\mathsf{sk}_I}$. $(\mathsf{sk}_I, \mathsf{pk}_I)$ represents the secret and the public keys of issuer I.
- Adversary $\mathcal{B}$ picks η random numbers $x_k \in \mathbb{F}_q$, and assigns to each owner O_k in the supply chain a public key $\mathsf{pk}_k = (g, \tilde{g}_k = g^{x x_k})$.

  Although, owner O_k does not know the secret key $\mathsf{sk}_k = x x_k$ that corresponds to the public key pk_k, owner O_k can always authenticate tag T by running a MAC-based authentication. Also, owner O_k can always transfer the ownership of tags he owns, since the ownership references do not depends on the secret key sk_k.
- To issue a tag T, $\mathcal{B}$ first selects ID and $K_T^0 \in \mathbb{F}_q$, then computes $\mathcal{S} = H(\mathsf{ID})^{\mathsf{sk}_I}$, $c_T^0 = (u_T^0, v_T^0) = (1, \mathcal{S})$. Finally, he stores $S_T^0 = (K_T^0, c_T^0)$ in tag T.

Learning phase. Adversary $\mathcal{B}$ simulates challenger $\mathcal{C}$.

- $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Tag}}$ and gives $\mathcal{A}$ two challenge tags T_0 and T_1.

- $\mathcal{A}$ calls the oracle $\mathcal{O}_{\mathsf{Execute}}$ for tags T_0 and T_1. Without loss of generality, we assume that T_0 and T_1 are owned by owner O_k and O_l respectively.

  After a successful authentication, the ciphertexts $c^j_{\mathsf{T}_0}$ and $c^j_{\mathsf{T}_1}$ stored into tags T_0 and T_1 respectively are updated using the pair (g^y, g^z) as follows:

$$\begin{aligned} c^{j+1}_{\mathsf{T}_0} &= (u^{j+1}_{\mathsf{T}_0}, v^{j+1}_{\mathsf{T}_0}) = (g^{yr_0^{j+1}}, H(\mathsf{ID}_0)^{\mathsf{sk}_I} g^{zx_k r_0^{j+1}}) \\ c^{j+1}_{\mathsf{T}_1} &= (u^{j+1}_{\mathsf{T}_1}, v^{j+1}_{\mathsf{T}_1}) = (g^{yr_1^{j+1}}, H(\mathsf{ID}_1)^{\mathsf{sk}_I} g^{zx_l r_1^{j+1}}) \end{aligned}$$

  Where r_0^{j+1} and r_1^{j+1} are randomly selected in $\mathbb{F}_q$.

- $\mathcal{B}$ provides $\mathcal{A}$ with additional s tags T'_i. The ownership of tags T'_i is transferred to $\mathcal{A}$ who can run mutual authentications with tags T'_i.

Challenge phase.

- In the challenge phase, $\mathcal{B}$ picks randomly $b \in \{0,1\}$ and returns tag T_b from the pair of tags T_0 and T_1. Then, he starts a mutual authentication with tag T_b *outside the range of adversary* $\mathcal{A}$ by sending a nonce $N^{j'}_{\mathsf{T}_b}$.

  Without loss of generality, we assume that tag T_b is owned by owner O_k (i.e., $b = 0$).

- At the end of the authentication, $\mathcal{B}$ updates the state of tag T_b as follows:

$$\begin{aligned} S^{j'+1}_{\mathsf{T}_b} &= (K^{j'+1}_{\mathsf{T}_b}, c^{j'+1}_{\mathsf{T}_b}) \\ K^{j'+1}_{\mathsf{T}_b} &= G(K^{j'}_{\mathsf{T}_b}, N^{j'}_{\mathsf{T}_b}, R^{j'}_{\mathsf{T}_b}) \\ c^{j'+1}_{\mathsf{T}_b} &= (u^{j'+1}_{\mathsf{T}_b}, v^{j'+1}_{\mathsf{T}_b}) = (g^{r^{j'+1}}, H(\mathsf{ID}_b)^{\mathsf{sk}_I} \tilde{g}_k^{r^{j'+1}}) \end{aligned}$$

  Where $R^{j'}_{\mathsf{T}_b}$ is the nonce generated by tag T_b during the mutual authentication.

- $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Flip}}$ and returns tag T_b to adversary $\mathcal{A}$.

  The ownership of tag T_b is then transferred to $\mathcal{A}$.

  Notice that $\mathcal{B}$ can compute correct ownership references for tag T_b:

$$\begin{aligned} \mathsf{ref}_{\mathsf{T}_b} &= (\mathcal{S}_b, \mathsf{id}_b, K_b^{\mathsf{old}}, K_b^{\mathsf{new}}, \mathsf{rand}_b^{\mathsf{old}}, \mathsf{rand}_b^{\mathsf{new}}) \\ &= (H(\mathsf{ID}_b)^{\mathsf{sk}_I}, \mathsf{ID}_b, -, K^{j'+1}_{\mathsf{T}_b}, -, h^{r^{j'+1}}) \end{aligned}$$

  Given that $\mathcal{A}$ does not have access to $N^{j'}_{\mathsf{T}_b}$ and $R^{j'}_{\mathsf{T}_b}$, $K^{j'+1}_{\mathsf{T}_b} = G(K^{j'}_{\mathsf{T}_b}, N^{j'}_{\mathsf{T}_b}, R^{j'}_{\mathsf{T}_b})$ does not give $\mathcal{A}$ any information about T_b's past interactions. Consequently, adversary $\mathcal{A}$ has to rely on ciphertext $c^{j'+1}_{\mathsf{T}_b}$ to build his attack against ROTIV's forward unlinkability.

- At the end of the challenge phase, $\mathcal{A}$ outputs his guess b' of bit b.

If $z = xy$, then the simulation of ROTIV by adversary $\mathcal{B}$ in the learning phase does not differ from an actual ROTIV system. As a result, adversary $\mathcal{A}$ can output a correct guess b' for bit b with a non-negligible advantage ϵ.

If $z \neq xy$, then the view of adversary $\mathcal{A}$ during the learning phase of the forward unlinkability game is independent of b. Therefore, adversary $\mathcal{A}$ has only a negligible advantage in outputting a correct guess b' for the bit b.

This constructs a statistical distinguisher between the two distributions (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, g^z), $x, y, z \in \mathbb{F}_q$, which breaks the DDH assumption in $\mathbb{G}_1$. In fact, if adversary $\mathcal{A}$ outputs a correct guess b' for the bit b, then adversary $\mathcal{B}$ outputs $z = xy$; otherwise adversary $\mathcal{B}$ outputs $z \neq xy$.

Hence, if adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the forward unlinkability of ROTIV, then adversary $\mathcal{B}$ will also have a non-negligible advantage $\epsilon' = \epsilon$ in breaking the DDH assumption in $\mathbb{G}_1$. This leads to a contradiction under the XDH assumption. □

4.5.2 Backward Unlinkability

Theorem 4.2. ROTIV *ensures backward unlinkability under the XDH assumption.*

Proof. Assume that there is an adversary $\mathcal{A}(r, s, t, \epsilon)$ who succeeds in the backward unlinkability game with a non-negligible advantage ϵ. We will now construct an adversary $\mathcal{B}$, who uses $\mathcal{A}$ as a subroutine and breaks the DDH assumption in $\mathbb{G}_1$ with a non-negligible advantage ϵ'.

To break the DDH assumption in $\mathbb{G}_1$, adversary $\mathcal{B}$ first queries the oracle $\mathcal{O}_{\mathsf{DDH}}$ to receive (g, g^x, g^y, g^z) and simulates a complete ROTIV system for $\mathcal{A}$. .

- Adversary $\mathcal{B}$ selects randomly a random number $\mathsf{sk}_I \in \mathbb{F}_q$ and computes $\mathsf{pk}_I = h^{\mathsf{sk}_I}$. $(\mathsf{sk}_I, \mathsf{pk}_I)$ represents the secret and the public keys of issuer I.
- Adversary $\mathcal{B}$ picks η random numbers $x_k \in \mathbb{F}_q$, and assigns to each owner O_k in the supply chain a public key $\mathsf{pk}_k = (g, \tilde{g}_k = g^{xx_k})$.
- To issue a tag T, $\mathcal{B}$ first selects ID and $K_T^0 \in \mathbb{F}_q$, then computes $\mathcal{S} = H(\mathsf{ID})^{\mathsf{sk}_I}$, $c_T^0 = (u_T^0, v_T^0) = (1, \mathcal{S})$. Finally, he stores $S_T^0 = (K_T^0, c_T^0)$ in tag T.

Learning phase. Adversary $\mathcal{B}$ simulates challenger $\mathcal{C}$ as follows.

- $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Tag}}$ and gives $\mathcal{A}$ two challenge tags T_0 and T_1.
- The ownership of tags T_0 and T_1 is transferred to adversary $\mathcal{A}$. $\mathcal{A}$ now has full control over tags T_0 and T_1.
- $\mathcal{B}$ provides $\mathcal{A}$ with s tags T_i' whose ownership is transferred to $\mathcal{A}$.

Challenge phase.

- Adversary $\mathcal{A}$ releases the ownership of the challenge tags T_0 and T_1 by calling the oracle $\mathcal{O}_{\mathsf{Transfer}}$.

- $\mathcal{B}$ simulates challenger $\mathcal{C}$ by first picking randomly $b \in \{0,1\}$ and returning tag T_b from the pair of tags T_0 and T_1. Then, he starts a mutual authentication with tag T_b *outside the range of adversary* $\mathcal{A}$ by sending a nonce $N_{\mathsf{T}_b}^{j'}$.

  Without loss of generality, we assume that tag T_b is owned by owner O_k.

- At the end of the authentication, $\mathcal{B}$ updates the state of tag T_b using the pair (g^y, g^z) as follows:

$$\begin{aligned} S_{\mathsf{T}_b}^{j'+1} &= (K_{\mathsf{T}_b}^{j'+1}, c_{\mathsf{T}_b}^{j'+1}) \\ K_{\mathsf{T}_b}^{j'+1} &= G(K_{\mathsf{T}_b}^{j'}, N_{\mathsf{T}_b}^{j'}, R_{\mathsf{T}_b}^{j'}) \\ c_{\mathsf{T}_b}^{j'+1} &= (u_{\mathsf{T}_b}^{j'+1}, v_{\mathsf{T}_b}^{j'+1}) = (g^{yr^{j'+1}}, H(\mathsf{ID}_b)^{\mathsf{sk}_I} g^{zx_k r^{j'+1}}) \end{aligned}$$

- $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Flip}}$ and returns tag T_b to adversary $\mathcal{A}$.

  Given that $\mathcal{A}$ does not have access to $N_{\mathsf{T}_b}^{j'}$ and $R_{\mathsf{T}_b}^{j'}$, it follows that $K_{\mathsf{T}_b}^{j'+1} = G(K_{\mathsf{T}_b}^{j'}, N_{\mathsf{T}_b}^{j'}, R_{\mathsf{T}_b}^{j'})$ does not give $\mathcal{A}$ any information about tag T_b, and adversary $\mathcal{A}$ has to build his attack against the backward unlinkability of ROTIV upon the ciphertext $c_{\mathsf{T}_b}^{j'+1}$.

- At the end of the challenge phase, $\mathcal{A}$ outputs his guess b' of bit b.

Note that if $z = xy$, then the ciphertext $c_{\mathsf{T}_b}^{j'+1}$ is a correct encryption of $H(\mathsf{ID}_b)^{\mathsf{sk}_I}$, i.e., $S_{\mathsf{T}_b}^{j'+1}$ is a valid state that corresponds to tag T_b. Hence, the simulation of ROTIV by adversary $\mathcal{B}$ does not differ from an actual ROTIV system, and adversary $\mathcal{A}$ can output a correct guess b' for bit b with a non-negligible advantage ϵ.

If $z \neq xy$, then the state $S_{\mathsf{T}_b}^{j'+1}$ does not correspond to tag T_b, and the view of adversary $\mathcal{A}$ during the backward unlinkability game is independent of b. Consequently, adversary $\mathcal{A}$ has only a negligible advantage in outputting a correct guess b' for the bit b.

This leads to a statistical distinguisher between the two distributions (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, g^z), $x, y, z \in \mathbb{F}_q$, breaking hereby the DDH assumption in $\mathbb{G}_1$. In fact, if adversary $\mathcal{A}$ outputs $b' = b$, then adversary $\mathcal{B}$ outputs $z = xy$; otherwise adversary $\mathcal{B}$ outputs $z \neq xy$.

Therefore, if $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the backward unlinkability of ROTIV, then adversary $\mathcal{B}$ will have a non-negligible advantage $\epsilon = \epsilon'$ in breaking the DDH assumption in $\mathbb{G}_1$. This contradicts the XDH assumption. □

4.6 Security Analysis

4.6.1 Secure Authentication

Theorem 4.3. ROTIV *ensures secure authentication under the resistance to existential forgery of MAC.*

Proof. To simplify the proof, we assume that the key K_T shared between a tag T and its owner is not updated after each successful authentication. As the key update is only required to achieve privacy and exclusive ownership, it is irrelevant for the authentication proof.

Let $\mathcal{O}_{\mathsf{MAC}}$ be an oracle that when queried with message m returns $\sigma = \mathsf{MAC}_K(m)$, where $K \in \mathbb{F}_q$.

We show that if there is an adversary $\mathcal{A}$ who breaks the security of ROTIV's authentication with a non-negligible advantage ϵ, then we can construct an adversary $\mathcal{B}$ that breaks the resistance to existential forgery of MAC (See Definition 2.10) with a non-negligible advantage ϵ'.

To break the resistance to existential forgery of MAC, adversary $\mathcal{B}$ simulates challenger $\mathcal{C}$ and creates a complete ROTIV system as described in the following.

- $\mathcal{B}$ selects randomly $x \in \mathbb{F}_q$, and computes h^x. Here, x is the secret key sk_I of issuer I and h^x is the corresponding public key pk_I.
- $\mathcal{B}$ selects η elements $x_k \in \mathbb{F}_q$, and provides each owner in ROTIV with the secret key $\mathsf{sk}_k = x_k$ and the matching public key $\mathsf{pk}_k = (g, \tilde{g}_k = g^{x_k})$.
- To initialize the set of n tags in ROTIV, adversary $\mathcal{B}$ proceeds as follows:
  - $\mathcal{B}$ selects randomly $\mathsf{ID}_i \in \mathbb{F}_q, 1 \leq i \leq n-1$ and computes $c^0_{T_i} = (1, H(\mathsf{ID}_i)^{\mathsf{sk}_I})$. Then, $\mathcal{B}$ selects randomly $K_{T_i} \in \mathbb{F}_q$ and stores $S^0_{T_i} = (K_{T_i}, c^0_{T_i})$ into T_i, $1 \leq i \leq n-1$. Finally, $\mathcal{B}$ computes T_i's ownership references ref_{T_i}.
  - Then $\mathcal{B}$ creates a tag T_n whose secret key is K. Tag T_n stores the state $S^0_{T_n} = c^0_{T_n}$.

Learning phase. Adversary $\mathcal{B}$ simulates challenger $\mathcal{C}$ as depicted below:

- $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Tag}}$ and returns r tags T_i to adversary $\mathcal{A}$, for which $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{Execute}}$.

  Note that if $\mathcal{A}$ selects tag T_n at this step of the game, then $\mathcal{B}$ simulates both tag T_n and T_n's owner $O_{(T_n,k)}$ by querying the oracle $\mathcal{O}_{\mathsf{MAC}}$.
- Again, $\mathcal{B}$ simulates oracle $\mathcal{O}_{\mathsf{Tag}}$ in order to return r additional tags T'_i to adversary $\mathcal{A}$. This time, $\mathcal{A}$ can read the internal states of tags T'_i. We point out that if adversary $\mathcal{A}$ selects tag T_n at this step, then $\mathcal{B}$ stops the authentication game.

Challenge phase.

- In the challenge phase, adversary $\mathcal{A}$ selects a challenge tag T_c by querying the oracle $\mathcal{O}_{\mathsf{Tag}}$. If $\mathsf{T}_c \neq T_n$, then $\mathcal{B}$ stops the authentication game.
- Otherwise, adversary $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{Execute}}$ with tag T_c which starts a ROTIV's authentication.

1.) If $\mathcal{A}$ impersonates $O_{(\mathsf{T}_c,k)}$, then $\mathcal{A}$ starts the authentication by sending a nonce $N^j_{\mathsf{T}_c}$ to T_c.

$\mathcal{B}$ simulates tag T_c: he generates a random nonce $R^j_{\mathsf{T}_c}$ and queries the oracle $\mathcal{O}_{\mathsf{MAC}}$ with message $m = (N^j_{\mathsf{T}_c}, R^j_{\mathsf{T}_c}, c^j_{\mathsf{T}_c})$. The oracle $\mathcal{O}_{\mathsf{MAC}}$ returns $\sigma = \mathsf{MAC}_K(m)$, and $\mathcal{B}$ sends $R^j_{\mathsf{T}_c}$, $c^j_{\mathsf{T}_c}$ and σ to adversary $\mathcal{A}$.

Adversary $\mathcal{A}$ replies with $(c^{j+1}_{\mathsf{T}_c}, \sigma_c)$.

Since adversary $\mathcal{A}$ has a non-negligible advantage ϵ in impersonating owner $O_{(\mathsf{T}_c,k)}$, $\sigma_c = \mathsf{MAC}_K(m_c)$, where $m_c = (R^j_{\mathsf{T}_c}, c^{j+1}_{\mathsf{T}_c})$. To break the resistance to existential forgery of MAC_K, adversary $\mathcal{B}$ outputs (m_c, σ_c).

2.) If $\mathcal{A}$ impersonates T_c, then $\mathcal{B}$ sends a fresh nonce $N^j_{\mathsf{T}_c}$ to $\mathcal{A}$. Upon receiving $N^j_{\mathsf{T}_c}$, $\mathcal{A}$ generates a random number $R^j_{\mathsf{T}_c}$ and sends $R^j_{\mathsf{T}_c}$, a ciphertext $c^j_{\mathsf{T}_c}$ and σ_c to $\mathcal{B}$.

If adversary $\mathcal{A}$ has a non-negligible advantage ϵ in impersonating tag T_c, then $\sigma_c = \mathsf{MAC}_K(N^j_{\mathsf{T}_c}, R^j_{\mathsf{T}_c}, c^j_{\mathsf{T}_c})$.

Accordingly, to break the existential forgery of MAC_K, $\mathcal{B}$ outputs (m_c, σ_c), where $m_c = (N^j_{\mathsf{T}_c}, R^j_{\mathsf{T}_c}, c^j_{\mathsf{T}_c})$.

Here we quantify the advantage ϵ' of adversary $\mathcal{B}$. Adversary $\mathcal{B}$ succeeds in breaking the resistance to existential forgery of MAC, if he does not stop the authentication game when simulating challenger $\mathcal{C}$.

Let E denote the event: $\mathcal{B}$ does not stop the authentication game.

Let E_1 denote the event: $\mathcal{B}$ does not stop the authentication game in the learning phase.

Let E_2 denote the event: $\mathcal{B}$ does not stop the authentication game in the challenge phase.

Let p denote the probability that $\mathcal{A}$ selects tag T_n.

Adversary $\mathcal{B}$ does not stop the authentication game in the learning phase, if and only if, adversary $\mathcal{A}$ does not pick tag T_n in the second phase of the learning phase. Consequently, $Pr(E_1) = (1-p)^r$. Further, $\mathcal{B}$ does not stop the authentication game in the challenge phase, if and only if, adversary $\mathcal{A}$ selects tag T_n as his challenge tag T_c. Thus, $Pr(E_2) = p$.

It follows that $\pi = Pr(E) = Pr(E_1)Pr(E_2) = (1-p)^r p$, and that $\epsilon' = \pi\epsilon$.

Therefore, if adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking ROTIV's security, then $\mathcal{B}$ will have a non-negligible advantage ϵ' in breaking the resistance to existential forgery by making at most $4rs + 1$ queries to the oracle $\mathcal{O}_{\mathsf{MAC}}$. This leads to a contradiction under the security of MAC.

Note that π is maximal when $p = \frac{1}{r}$ and $\pi_{\mathsf{max}} = \frac{\left(1-\frac{1}{r}\right)^r}{r} \simeq \frac{1}{er}$. □

4.6.2 Exclusive Ownership

Theorem 4.4. ROTIV *ensures exclusive ownership under the XDH assumption.*

Proof. Assume there is an adversary $\mathcal{A}$ who succeeds in the exclusive ownership game with a non-negligible advantage ϵ. We show that there is an adversary $\mathcal{B}$ who uses adversary $\mathcal{A}$ to break the DDH assumption in $\mathbb{G}_1$ with a non-negligible advantage ϵ'.

To break the DDH assumption in $\mathbb{G}_1$, adversary $\mathcal{B}$ proceeds as follows.

First, $\mathcal{B}$ simulates challenger $\mathcal{C}$ and creates a complete ROTIV system.

- $\mathcal{B}$ selects randomly $\mathsf{sk}_I \in \mathbb{F}_q$ and computes $\mathsf{pk}_I = h^{\mathsf{sk}_I}$. Here sk_I is the secret key of issuer I and pk_I is the corresponding public key.
- $\mathcal{B}$ selects randomly η random numbers $x_k \in \mathbb{F}_q$, and provides each owner O_k in the supply chain with a pair of matching public and secret keys $\mathsf{pk_k} = (g, \tilde{g}_k = g^{x_k})$ and $\mathsf{sk}_k = x_k$.
- $\mathcal{B}$ creates n tags T_i.

Learning phase. Adversary $\mathcal{A}$ enters the learning phase.

- $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Tag}}$ and supplies $\mathcal{A}$ with r tags T_i.
- Adversary $\mathcal{A}$ is allowed to run authentication sessions with tags T_i, to acquire their ownership and to transfer this ownership to owners of his choice.

Challenge phase.

- Adversary $\mathcal{B}$ queries the oracle $\mathcal{O}_{\mathsf{DDH}}$ to receive (g, g^x, g^y, g^z).
- In the challenge phase, $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Tag}}$ and provides $\mathcal{A}$ with a challenge tag T_c. Without loss of generality, we assume that T_c's owner is O_k.

  Before giving the tag T_c to adversary $\mathcal{A}$, adversary $\mathcal{B}$ encrypts T_c's signature using g^x from the DDH challenge:

$$c^j_{\mathsf{T}_c} = (u^j_{\mathsf{T}_c}, v^j_{\mathsf{T}_c}) = (g^{xr}, H(\mathsf{ID}_c)^{\mathsf{sk}_I}(g^{xr})^{x_k}) = (g^{xr}, H(\mathsf{ID}_c)^{\mathsf{sk}_I}\tilde{g}_k^{xr})$$

  Where r is a random number in $\mathbb{F}_q$.
- $\mathcal{A}$ now can read the internal state of tag T_c.
- $\mathcal{A}$ selects a challenge owner O_c, and transfers the ownership of tag T_c to O_c.

If adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the exclusive ownership, then adversary $\mathcal{A}$ will be able to supply O_c during the ownership transfer protocol with correct ownership references:

$$\mathsf{ref}_T = (\mathcal{S}, \mathsf{id}, K^{\text{old}}, K^{\mathsf{new}}, \mathsf{rand}^{\mathsf{old}}, \mathsf{rand}^{\mathsf{new}})$$

Where $e(g^{xr}, h) = e(g, \mathsf{rand}^{\mathsf{new}})$, i.e., $\mathsf{rand}^{\mathsf{new}} = h^{xr}$.

To break the DDH assumption, adversary $\mathcal{B}$ verifies whether $e(g^z, h^r) = e(g^y, \mathsf{rand}^{\mathsf{new}}) = e(g^y, h^{xr})$. If so, $\mathcal{B}$ outputs $z = xy$, otherwise, he outputs $z \neq xy$.

As a result, if adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking exclusive ownership, then adversary $\mathcal{B}$ will have a non-negligible advantage $\epsilon' = \epsilon$ in breaking the DDH assumption.

□

4.6.3 Issuer Verification Security

To prove the security of issuer verification in ROTIV, we first show that the short signature we employ to sign tags' identifiers is secure (resistant to existential forgery).

Theorem 4.5. *The short signature presented in Section 4.4.1.1 is secure in the random oracle model under the BCDH assumption.*

Proof. Assume there is an adversary $\mathcal{A}$ who breaks the resistance to existential forgery (see Definition 2.20) of ROTIV's short signature with a non-negligible advantage ϵ, we show that there is an adversary $\mathcal{B}$ who uses $\mathcal{A}$ to break the BCDH assumption, see Definition 2.32, with a non-negligible advantage ϵ'.

Let $\mathcal{O}_{\mathsf{BCDH}}$ be an oracle that selects randomly $x, y, z \in \mathbb{F}_q$, and returns $g, g^x, g^y, g^z \in \mathbb{G}_1$, and $h, h^x, h^y \in \mathbb{G}_2$.

To break the BCDH assumption, adversary $\mathcal{B}$ simulates **1.)** a short signature scheme of secret key $\mathsf{sk} = x$ and public key $\mathsf{pk} = h^x$, and **2.)** a random oracle $\mathcal{H}$ to compute H.

Simulation of the random oracle $\mathcal{H}$. To respond to the queries of the random oracle $\mathcal{H}$, adversary $\mathcal{B}$ keeps a table T_H of tuples $(m_j, r_j, \mathsf{coin}(m_j), H(m_j))$ as explained below.

On a query $H(m_i)$, $\mathcal{B}$ replies as follows:

1.) If there is a tuple $(m_i, r_i, \mathsf{coin}(m_i), H(m_i))$ that corresponds to m_i, then $\mathcal{B}$ returns $H(m_i)$.

2.) If m_i has never been queried before, then adversary $\mathcal{B}$ picks a random number $r_i \in \mathbb{F}_q$, and flips a random coin $\mathsf{coin}(m_i) \in \{0, 1\}$ such that: $\mathsf{coin}(m_i) = 1$ with probability p, and it is equal to 0 with probability $1 - p$. If $\mathsf{coin}(m_i) = 0$, then $\mathcal{B}$ answers with $H(m_i) = g^{r_i}$. Otherwise, he answers with $H(m_i) = (g^z)^{r_i}$. Finally, adversary $\mathcal{B}$ stores the tuple $(m_i, r_i, \mathsf{coin}(m_i), H(m_i))$ in table T_H.

Learning phase. In the learning phase of the resistance to existential forgery game, adversary $\mathcal{B}$ simulates challenger $\mathcal{C}$. We recall that adversary $\mathcal{A}$ is allowed to make r_s query to the signature oracle $\mathcal{O}_{\mathsf{Sign}}$.

On query of a message m_i to the oracle $\mathcal{O}_{\mathsf{Sign}}$, $\mathcal{B}$ simulates the random oracle $\mathcal{H}$ and gets the tuple $(m_i, r_i, \mathsf{coin}(m_i), H(m_i))$.

- If $\mathsf{coin}(m_i) = 0$, then adversary $\mathcal{B}$ computes $\mathsf{Sign}_{\mathsf{sk}}(m_i) = H(m_i)^x = g^{r_i x}$.
- If $\mathsf{coin}(m_i) = 1$, then adversary $\mathcal{B}$ stops the game as he cannot compute $\mathcal{S}_i = \mathsf{Sign}_{\mathsf{sk}}(m_i)$.

Challenge phase. In the challenge phase, adversary $\mathcal{A}$ returns a challenge message m_c and a signature $\mathcal{S}_c$.

Since adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the resistance to existential forgery of ROTIV's short signature, it follows that $e(\mathcal{S}_c, h) = e(H(m_c), h^x)$, i.e., $\mathcal{S}_c = H(m_c)^x$.

Now, when receiving the pair $(m_c, \mathcal{S}_c)$, adversary $\mathcal{B}$ queries the random oracle $\mathcal{H}$ with m_c and obtains the tuple $(m_c, r_c, \mathsf{coin}(m_c), H(m_c))$.

- If $\mathsf{coin}(m_c) = 0$, adversary $\mathcal{B}$ stops the game.
- If $\mathsf{coin}(m_c) = 1$, then $H(m_c) = g^{z r_c}$, and therewith, $\mathcal{S}_c = g^{xzr_c}$. Consequently, adversary $\mathcal{B}$ breaks the BCDH assumption by outputting:

$$e(\mathcal{S}_c, h^y)^{\frac{1}{r_c}} = e(g^{xzr_c}, h^y)^{\frac{1}{r_c}} = e(g, h)^{xyz}$$

Note that adversary $\mathcal{B}$ breaks the resistance to existential forgery if he does not stop the security game.

Let E denote the event: $\mathcal{B}$ does not stop the security game.

Let E_1 denote the event: $\mathcal{B}$ does not stop the security game in the learning phase.

Let E_2 denote the event: $\mathcal{B}$ does not stop the security game in the challenge phase.

Adversary $\mathcal{B}$ does not stop the game in the learning phase, if and only if, for all the r_s queries m_i to the oracle $\mathcal{O}_{\mathsf{Sign}}$, $\mathsf{coin}(m_i) = 0$. Therefore, $Pr(E_1) = (1-p)^{r_s}$.

Additionally, $\mathcal{B}$ does not stop the authentication game in the challenge phase, if and only if, $\mathsf{coin}(m_c) = 1$, and as a result, $Pr(E_2) = p$.

We conclude that: $\pi = Pr(E) = Pr(E_1)Pr(E_2) = (1-p)^{r_s}p$, and that $\epsilon' = \pi\epsilon$.

Accordingly, if adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the resistance to existential forgery, then $\mathcal{B}$ will have a non-negligible advantage ϵ' in breaking the BCDH assumption.

We indicate that π is maximal when $p = \dfrac{1}{r_s}$ and $\pi_{\mathsf{max}} = \dfrac{\left(1 - \frac{1}{r_s}\right)^{r_s}}{r_s} \simeq \dfrac{1}{e r_s}$. □

Theorem 4.6. ROTIV *ensures issuer verification security under the resistance to existential forgery of the short signature.*

Proof. Assume there is an adversary $\mathcal{A}$ who breaks the issuer verification security of ROTIV with a non-negligible advantage ϵ, we build an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the resistance to existential forgery of ROTIV's short signature with a non-negligible advantage ϵ'.

$\mathcal{B}$ simulates challenger $\mathcal{C}$ for the issuer verification game by creating a complete ROTIV system.

- $\mathcal{B}$ selects η random numbers $x_k \in \mathbb{F}_q$ and computes $\tilde{g}_k = g^{x_k}$, then assigns to each owner O_k in the supply chain the matching pair of secret and public keys $(\mathsf{sk}_k, \mathsf{pk}_k) = (x_k, (g, \tilde{g}_k))$.
- $\mathcal{B}$ simulates issuer I whose public key is $\mathsf{pk} = h^{\mathsf{sk}}$, which is the public key of the challenge short signature, and initializes n tags T_i: he selects randomly $\mathsf{ID}_i \in \mathbb{F}_q$, then queries the oracle $\mathcal{O}_{\mathsf{Sign}}$ which returns $\mathcal{S}_i = H(\mathsf{ID}_i)^{\mathsf{sk}}$. Provided with $\mathcal{S}_i$, adversary $\mathcal{B}$ computes the ownership references of tag T_i.

Learning phase. Adversary $\mathcal{A}$ enters the learning phase.

- $\mathcal{B}$ simulates $\mathcal{O}_{\mathsf{Tag}}$ and supplies $\mathcal{A}$ with r tags T_i. Using the ownership references ref_{T_i} of tag T_i, adversary $\mathcal{B}$ transfers the ownership of T_i to $\mathcal{A}$.
- Now, $\mathcal{A}$ has full control over tag T_i, he can now run authentications with tags T_i and transfer their ownership.

Challenge phase.

- In the challenge phase, adversary $\mathcal{A}$ creates a new tag T_c (i.e., $\mathsf{ID}_i \neq \mathsf{ID}_c$, where ID_c is T_c's identifier).
- Adversary $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathsf{Owner}}$ and provides $\mathcal{A}$ with a challenge owner O_c.
- Adversary $\mathcal{A}$ calls the oracle $\mathcal{O}_{\mathsf{Transfer}}$ to transfer the ownership of tag T_c to O_c.

Since adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the security of issuer verification, it follows that adversary $\mathcal{A}$ will output valid ownership references $\mathsf{ref}_{\mathsf{T}_c}$ for tag T_c:

$$\mathsf{ref}_{\mathsf{T}_c} = (\mathcal{S}_c, \mathsf{id}_c, K_c^{\mathsf{old}}, K_c^{\mathsf{new}}, \mathsf{rand}_c^{\mathsf{old}}, \mathsf{rand}_c^{\mathsf{new}})$$

This implies that $\mathcal{S}_c$ is the signature of id_c with secret key sk.

Now to break the resistance to existential forgery of ROTIV's short signature, adversary $\mathcal{B}$ outputs $(\mathsf{id}_c, \mathcal{S}_c)$.

Hence, if there is an adversary $\mathcal{A}$ who wins the issuer verification game of ROTIV with a non-negligible advantage ϵ, then there is an adversary $\mathcal{B}$ who breaks the resistance to existential forgery of ROTIV's short signature with a non-negligible advantage $\epsilon' = \epsilon$. □

4.7 Related Work

Molnar et al. (117) address the problem of ownership transfer in RFID systems by using tag pseudonyms and relying on a trusted third party. Here, the TTP is the only entity than can identify tags. To transfer the ownership of some tag T, its current owner $O_{(T,k)}$ and its prospective owner $O_{(T,k+1)}$, contact the TTP, which then provides $O_{(T,k+1)}$ with T's identity. Once the ownership transfer of T takes place, the TTP refuses identity requests from T's previous owner $O_{(T,k)}$. Still, relying on a TTP is a drawback: in many scenarios, the availability of a trusted third party during tag ownership transfer is probably unrealistic.

Other solutions based on symmetric primitives have been proposed by Lim and Kwon (111), Fouladgar and Afifi (60), Song (149), and Kulseng et al. (101). These schemes however suffer as discussed in Section 4.2.2 from three major drawbacks: **1.)** tag identification and authentication is linear in the number of tags, **2.)** denial of service attacks and **3.)** no tag issuer verification.

Dimitriou (51) proposes a solution to ownership transfer that relies on symmetric cryptography while relaxing the privacy requirements for both backward and forward unlinkability. Unlike previous schemes on ownership transfer, this solution allows an owner of a tag to revert the tag to its original state. This is useful for after sale services where a retailer can recognize a sold tag T. Note that ROTIV offers the same feature: the unique identifier of a tag T enables any owner to verify whether he owned T before or not.

4.8 Summary

In this chapter, we presented ROTIV to address the security and the privacy issues related to RFID ownership transfer in supply chains. As part of ownership transfer, ROTIV offers a constant-time and privacy-preserving authentication while tags only evaluate a hash function. It also enables issuer verification that allows every owner in the supply chain to verify the origin of tags that he owns. ROTIV's main idea is to **1.)** combine a MAC-based authentication with Elgamal encryption to achieve constant-time and privacy preserving authentication, and **2.)** to use a short signature scheme to execute issuer verification. ROTIV is provably secure and privacy preserving under standard assumptions: MAC security, the BCDH assumption and the XDH assumption.

5

RFID-based Product Tracking in Supply Chains

5.1 Introduction

Product tracking is one of the major applications of RFID-enabled supply chains as it allows genuineness verification and replica prevention of products (56, 83, 118, 151, 160). The idea is to trace the path that products took in the supply chain by reading their attached RFID tags. However, the use of RFID tags for genuineness verification comes with new threats to security and privacy of both tags and partners in the supply chain.

With respect to security, it must be verifiable whether a product is genuine by only scanning the tag attached to the product. To this end, the supply chain has a set of verifiers that check the path that tags took in the supply chain, whereas readers along the supply chain update the states of tags in their vicinity. The main challenge is to enable readers to update tags' states while preventing them from injecting fake products.

The second challenge regards the privacy of tags. Typically, partners in the supply chain do not want to reveal any information about their internal details, strategic relationships and processes to adversaries, e.g., competitors or customers. Thus, an adversary must not be able to trace and recognize tags through subsequent steps in the supply chain.

Solutions addressing these security and privacy requirements have to be lightweight to allow wide deployment. Ideally, they should be suited for the cheapest RFID tags, namely, storage only tags. Therefore, any cryptographic computation required by the scheme should be performed by the readers. Moreover, the path verification at the readers should not be computationally heavy to avoid overloading readers and hindering supply chain performances.

Along these lines, we present in this chapter two protocols called Tracker and Checker for secure and privacy-preserving RFID-based product tracking in the supply chain. The main idea behind these two protocols is to encode paths in a supply chain using polynomials and then employ the path encoding to sign tags' identifiers. Tracker targets the product

traceability scenario where the genuineness verification is performed by a *trusted* party called *manager*, whereas CHECKER addresses the problem of on-site checking by enabling each reader in the supply chain to act as a *non-trusted* verifier.

The major contributions of the protocols proposed in this chapter are:

- They allow to determine the exact path that each tag went through in the supply chain.
- They provide provable privacy and security in the random oracle model.
- Contrary to related work such as Ouafi and Vaudenay (127) or Li and Ding (110), our protocols *do not require tags to perform any computation*. Instead, they rely on *storage only* tags .

The rest of this chapter is structured as follows: after presenting the notations that will be used throughout this chapter in Section 5.2, we introduce formal definitions that capture the security and the privacy requirements of product tracking in Section 5.3. Then in Section 5.4, we present TRACKER, our first tracking protocol that relies on a trusted party to perform the path verification for tags in the supply chain. In Section 5.5, we introduce CHECKER which implements on-site checking by allowing each reader in the supply chain to verify the genuineness of products. Then in Section 5.6, we survey some of the previous work on product tracking. Finally, Section 5.7 concludes the chapter.

5.2 Notations

A supply chain in this chapter simply denotes series of consecutive steps that a product can go through. The exact meaning or semantic of such a "step" in the supply chain depends on the particular application and will not be discussed here, one could imagine a step being a warehouse, retail store or a manufacturing unit. Each step of the supply chain is equipped with an RFID reader, and when a product moves to the subsequent step of a supply chain, an interaction takes place between the product's RFID tag and the reader associated with the step. Finally, verifiers want to know whether a product in their vicinity went through a "correct" sequence of steps in the supply chain or not.

Accordingly, a product tracking system involves the following entities:

5.2.1 Entities

- **Tags** T_i: Each tag is attached to a product or object. A tag T_i features re-writable memory representing T_i's current "state" denoted $S^j_{T_i}$.
- **Issuer** I: The issuer I prepares tags for deployment. When attaching a tag T_i to a product, I writes an initial state $S^0_{T_i}$ into T_i.

- **Readers** R_k: Representing step v_k in the supply chain, reader R_k can interact with a product's tag T_i in its range. It reads out T_i's current state $S^j_{T_i}$ and writes an updated state $S^{j+1}_{T_i}$ into T_i.

- **Verifiers** V_k: The supply chain has a set of checkpoints p_k. Each checkpoint p_k is associated with a verifier V_k. At checkpoint p_k, the verifier V_k checks the genuineness of products that are present in his site. This is carried out by verifying whether a tag T_i has passed through a valid ("correct") sequence of steps in the supply chain that leads to verifier V_k. To this effect, verifier V_k reads out the current state $S^j_{T_i}$ of T_i, and based on a set of ν_k verification keys $\mathcal{K}^k_V = \{K^1_k, K^2_k, ..., K^{\nu_k}_k\}$, verifier V_k decides whether T_i went through a valid path $\mathcal{P}_{\text{valid}_i}$ that leads to V_k or not.

Remark 5.1. *Verifiers V_k in a tracking system could either be:*

- *A single trusted party (i.e., cannot be corrupted by adversaries) called manager M, who at the end of the supply chain verifies whether tags went through valid paths or not. This scenario corresponds to product traceability by trusted party, see Section 5.4*

- *Readers R_k which are potentially malicious. In this scenario, each step v_k in the supply chain is a checkpoint p_k. This corresponds to on-site checking protocols, cf. Section 5.5.*

5.2.2 Supply Chain

Formally, a supply chain is represented by a digraph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ consisting of vertices V and edges E. Each vertex $\mathsf{v}_k \in \mathsf{V}$ is equivalent to one *step* in the supply chain. A vertex/step v_k in the supply chain is uniquely associated with a reader R_k. Each directed edge $e \in E$, $e := \overrightarrow{\mathsf{v}_j\mathsf{v}_k}$, from vertex v_j to vertex v_k, expresses that v_k is a possible next step to step v_j in the supply chain. This simply means that according to the organization of the supply chain, a product might proceed to step v_k after being at step v_j. If products must not advance from step v_j to v_k, then $\overrightarrow{\mathsf{v}_j\mathsf{v}_k} \notin \mathsf{E}$. Note that a supply chain can include loops and reflexive edges. Whenever a product in the supply chain proceeds from step v_j to step v_k, reader R_k interacts with the product's tag. The issuer I of the supply chain is represented in G by the only vertex without incoming edges v_0.

A *path* $\mathcal{P}$ is a finite sequence of steps $\mathcal{P} = \{\mathsf{v}_0, \mathsf{v}_1, \ldots, \mathsf{v}_l\}$, where $\forall k \in \{0, \ldots, l-1\}$: $\overrightarrow{\mathsf{v}_k\mathsf{v}_{k+1}} \in \mathsf{E}$, and l is the *length* of path $\mathcal{P}$. Clearly, paths can have different path lengths. Whereas a *valid* path $\mathcal{P}_{\text{valid}}$ is a particular legitimate sequence of steps that products are allowed to take. There may be up to ν multiple different valid paths $\{\mathcal{P}_{\text{valid}_1}, \mathcal{P}_{\text{valid}_2}, \ldots, \mathcal{P}_{\text{valid}_\nu}\}$, in a supply chain.

When a tag T arrives at a checkpoint p_k, the verifier V_k associated with this checkpoint checks for T's path validity. While verifier V_k might not know *all* possible paths in G, we assume in the following that each verifier V_k knows the *valid paths* that lead to him.

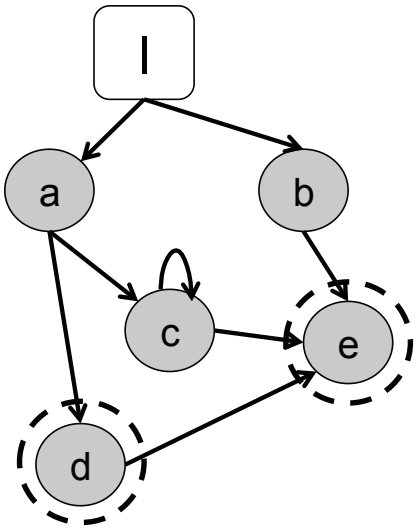

Figure 5.1: Simple supply chain, checkpoints are encircled.

Figure 5.1 depicts a sample supply chain, we note that checkpoints p_k where verifiers V_k checks the genuineness of tags/products are encircled. So, after their deployment at issuer I, tags can either start in steps a or b. Valid paths in Figure 5.1 are, for example, $\{I, a, d\}$, $\{I, a, d, e\}$ or $\{I, a, c, c, e\}$.

5.2.3 A Tracking System

Using the above definitions, a complete product tracking system consists of

- a supply chain $\mathsf{G} = (\mathsf{V}, \mathsf{E})$;
- a set $\mathcal{T}$ of n different tags;
- a set of possible states $\mathcal{S}$ that can be stored into tags;
- a total of η different readers, $\eta = |\mathsf{V}|$;
- issuer I;
- a set $\mathcal{V}$ of m verifiers ($m = 1$ or $m = \eta$);
- a set of ν valid paths;
- a set of valid states $\mathcal{S}_{\text{valid}}$ that can be stored into tags and which are accepted by the verifiers of the supply chain;
- a function $\mathsf{Read} : \mathcal{T} \rightarrow \mathcal{S}$ that reads out tag T and returns T's current state S_T^j;
- a function Write: $\mathcal{T} \times \mathcal{S} \rightarrow \mathcal{S}$ that writes a new state S_T^{j+1} into tag T;

- a function Check: $\mathcal{S} \times \mathcal{V} \rightarrow \{0,1\}$. $\mathsf{Check}(S_T^j, V_k) = 1$, if tag T went through some valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain that leads to verifier V_k, and 0 otherwise.

5.3 Adversary Model

Readers in a tracking system are supposed to read the state stored into tags and update it accordingly. We assume that readers' corruption is possible. That is, readers can try tracking tags in order to spy on other readers, as well as injecting fake products in the supply chain.

Further, we assume that the issuer I is honest and cannot be corrupted by adversaries. This implies that when tags are initialized at the beginning of the supply chain by I, these tags will definitely meet the supply chain requirements and quality standards.

As the two protocols proposed in this chapter rely on storage only tags to implement product tracking, an adversary $\mathcal{A}$ against product tracking is not only allowed to eavesdrop on tags' communication but to also tamper with tags' internal states. Adversary $\mathcal{A}$ can as well have access to the communication between tags and readers and know the steps v_k that a tag T is visiting. He can also monitor a step v_k in the supply chain by eavesdropping on tags going into or leaving the step v_k.

To capture formally these capabilities in our security and privacy definitions, a challenger $\mathcal{C}$ provides adversary $\mathcal{A}$ with access to the following oracles:

- $\mathcal{O}_{\mathsf{Tag}}(\mathsf{param})$: When queried with a parameter param, the oracle $\mathcal{O}_{\mathsf{Tag}}$ randomly selects a tag T from the n tags $\mathcal{T}$ in the supply chain that fulfills the parameter param. Then, it returns tag T to adversary $\mathcal{A}$. For example:

  1. To have access to a tag T which just entered the supply chain (i.e., tag T is at step v_0), $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{Tag}}$ with parameter param = "tag at step v_0".
  2. To have access to a tag T whose identifier is ID, $\mathcal{A}$ calls the oracle $\mathcal{O}_{\mathsf{Tag}}$ with parameter param = "tag with identifier ID". $\mathcal{O}_{\mathsf{Tag}}$ returns a tag with identifier ID if there is any.
  3. To have access to a tag T whose next step in the supply chain is the step v_k, $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{Tag}}$ with parameter param = "tag's next step is v_k".

- $\mathcal{O}_{\mathsf{Check}}(T, V_k)$: On input of tag T and verifier V_k, the oracle $\mathcal{O}_{\mathsf{Check}}$ returns the output of $\mathsf{Check}(S_T^j, V_k)$.

- $\mathcal{O}_{\mathsf{Step}}(T)$: On input of tag T, the oracle $\mathcal{O}_{\mathsf{Step}}(T)$ returns the *next* step of tag T in the supply chain to adversary $\mathcal{A}$.

- $\mathcal{O}_{\mathsf{Flip}}(T_0, T_1)$: On input of two tags T_0 and T_1, the oracle $\mathcal{O}_{\mathsf{Flip}}$ flips a coin $b \in \{0,1\}$ and returns tag T_b to adversary $\mathcal{A}$.

- $\mathcal{O}_{\mathsf{CorruptR}}(R_k)$: On input of reader R_k, the oracle $\mathcal{O}_{\mathsf{CorruptR}}$ returns the secret information Sec_k associated with reader R_k to adversary $\mathcal{A}$. We say that adversary $\mathcal{A}$ controls the step v_k associated with reader R_k.

Note that whenever adversary $\mathcal{A}$ is given access to a tag T, $\mathcal{A}$ is allowed to read from T by calling the function Read and to write into T through the function Write.

By having access to these oracles, an adversary $\mathcal{A}$ is able **1.)** to corrupt readers, **2.)** to have an arbitrary access to tags, and **3.)** to monitor readers in the supply chain.

5.3.1 Security

A secure product tracking protocol has to fulfill the following two properties:

5.3.1.1 Completeness

Completeness ensures that when a tag T stores a state $S_T^j \in \mathcal{S}_{\text{valid}}$, it follows that there is a verifier V_k in the supply chain that will accept tag T, i.e., $\mathsf{Check}(S_T^j, V_k) = 1$.

Definition 5.1 (Completeness)**.** *A product tracking protocol is said to be complete* **iff***, for any tag T storing a valid state $S_T^j \in \mathcal{S}_{\text{valid}}$, there exists a verifier $V_k \in \mathcal{V}$ such that* $\mathsf{Check}(S_T^j, V_k) = 1$.

Denial of service through malicious writing. We remind the reader that in this chapter we target storage only tags that cannot implement any reader authentication mechanisms. As a result, an adversary $\mathcal{A}$ might write any content into tags at any time to spoil their genuineness verification in the supply chain. That is, even if a tag has been through a valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain, the adversary might still replace and invalidate the state of the tag leading the Check function to output "0". This corresponds to a denial of service attack.

Still, we believe that the scope of such attacks is limited, since only partners in the supply chain can access a large number of tags. While it is reasonable to assume that these partners may try to learn sensitive information about other partners through the tags they scan, it is highly unlikely that they will invalidate the content of tags that are present in their sites.

5.3.1.2 Soundness

Soundness ensures that it is computationally infeasible for an adversary $\mathcal{A}$ to forge a valid state for a tag T that did not go through a valid path in the supply chain. This corresponds to the soundness property of the Check function. Using the notations presented in Section 5.2, this goal is stated as follows: **if** the Check function computed by some verifier V_k in the supply chain using the internal state S_T^j of some tag T returns 1, **then** this implies that tag T must have gone through some valid $\mathcal{P}_{\text{valid}_i}$ in the supply chain that leads to verifier V_k.

Algorithm 5.3.1: Learning phase of the soundness game

for $i := 1$ **to** r **do**
  $\mathsf{Sec}_i \leftarrow \mathcal{O}_{\mathsf{CorruptR}}(R_i)$;
for $i := 1$ **to** ρ **do**
  IterateSupplyChain;
  for $j := 1$ **to** s **do**
    $T_{(i,j)} \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{(i,j)})$;
    $S^i_{T_{(i,j)}}$:=Read$(T_{(i,j)})$;
    Write$(T_{(i,j)}, S'^i_{T_{(i,j)}})$;
    $b_{T_{(i,j)}} \leftarrow \mathcal{O}_{\mathsf{Check}}(T_{(i,j)}, V_{T_{(i,j)}})$;

Algorithm 5.3.2: Challenge phase of the soundness game

$\mathsf{T}_c \leftarrow \mathcal{A}$;
for $i := 1$ **to** m **do**
  $b_{(i,\mathsf{T}_c)} \leftarrow \mathcal{O}_{\mathsf{Check}}(\mathsf{T}_c, V_i)$;

It is important to note however that when we say that a tag went through the valid path $\mathcal{P}_{\text{valid}_i} = \overrightarrow{\mathsf{v}_0\mathsf{v}_1...\mathsf{v}_l}$, this means that the tag was issued by I and that its state has been updated correctly using the secrets of readers $R_1, R_2, ..., R_l$ in that order. It does not mean that the tag went actually through the steps composing the path $\mathcal{P}_{\text{valid}_i}$. If we imagine a scenario where an adversary $\mathcal{A}$ knows all the readers' secrets, adversary $\mathcal{A}$ can update the state of any tag in the supply chain and make it look as if it went through a step $\mathsf{v}_k \in \{\mathsf{v}_1, \mathsf{v}_2, ..., \mathsf{v}_\eta\}$ without the tag leaving the range of adversary $\mathcal{A}$.

Consequently, we say that a tracking protocol is *sound*, if and only if, a verifier V_k in the supply chain accepts a tag T only when the state of tag T has been updated correctly using the secrets of readers in some valid path leading to verifier V_k in an orderly fashion.

Now we formalize the soundness property of tracking schemes using a security game as depicted in Algorithm 5.3.1 and Algorithm 5.3.2.

In this game, an adversary $\mathcal{A}$ runs in two phases. First in the learning phase, adversary $\mathcal{A}$ can corrupt r readers of his choice by calling the oracle $\mathcal{O}_{\mathsf{CorruptR}}$. Then, adversary $\mathcal{A}$ is allowed to iterate the supply chain ρ times by calling the function IterateSupplyChain. Whenever called, the function IterateSupplyChain advances the tags in the supply chain to their next step. Now per iteration, $\mathcal{A}$ gets access to a set of s tags $T_{(i,j)}$ through the oracle $\mathcal{O}_{\mathsf{Tag}}$, he can then read-out and re-write their internal states with some arbitrary data. Also, adversary $\mathcal{A}$ has access to the oracle $\mathcal{O}_{\mathsf{Check}}$ which whenever queried with a tag $T_{(i,j)}$, returns the output of the CHECK function.

Finally in the challenge phase, adversary $\mathcal{A}$ selects a challenge tag T_c that he returns to the challenger $\mathcal{C}$, who then gives T_c to the oracle $\mathcal{O}_{\mathsf{Check}}$. The oracle $\mathcal{O}_{\mathsf{Check}}$ outputs a set of

m bits $b_{(i,\mathsf{T}_c)}$ such that $b_{(i,\mathsf{T}_c)} = \mathsf{Check}(S_{\mathsf{T}_c}, V_i)$.

$\mathcal{A}$ is said to win the soundness game if and only if, **i.)** $\exists$ a verifier V_i such that $\mathsf{Check}(S_{\mathsf{T}_c}, V_i) = 1$, i.e., there is a valid path $\mathcal{P}_{\text{valid}_i}$ that leads to verifier V_i and that corresponds to T_c's state; **ii.)** $\exists\ \mathsf{v}_k \in \mathcal{P}_{\text{valid}_i}$ such that the reader R_k associated with step v_k was not corrupted by $\mathcal{A}$; **iii.)** and finally, T_c did not go through step v_k.

The advantage ϵ of adversary $\mathcal{A}$ in winning the soundness game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins}) - \frac{|\mathcal{S}_{\text{valid}}|}{|\mathcal{S}|}$$

Definition 5.2 (Soundness)**.** *A product tracking protocol is said to be sound* **iff***, for any adversary $\mathcal{A}(r, s, \rho, \epsilon)$, the advantage ϵ in winning the soundness game is negligible.*

The adversary $\mathcal{A}$ captured by the definition above is a strong adversary in the sense of (159). He can access tags arbitrarily and tamper with their states. He is also allowed to access the output of the protocol and corrupt readers. In the real world, such an adversary corresponds to a malicious partner whose goal is to inject fake products into the supply chain.

Remark 5.2. *The adversary model above does not capture an adversary hijacking tags and performing "extra" steps with tags. For example, if the "extra" steps do not change the tags' state, this will go unnoticed by the verifiers. We claim that these attacks, as well as physical attacks, e.g., removing one tag from one product and attaching it to another product, cannot be tackled using cryptographic measures especially when using storage only tags.*

Cloning. As we assume cheap re-writable tags without any computational abilities, no tag/reader authentication is possible on the tag side. Any adversary can read from and write into a tag. Trivially, an adversary might "clone" a tag. This is impossible to prevent in our setup with storage only tags.

We note however that when the verification of genuineness is performed by a single trusted party (manager M in Section 5.4), the cloning can be easily detected and therewith mitigated by keeping a database DB_M on the manager M. Initially empty, DB_M will contain identifiers of tags that went through a valid path of a supply chain and were checked by manager M. Each time manager M is required to verify the genuineness of some tag, he first checks whether this tag's identifier is already in DB_M – to detect cloning. Details about identifiers and handling of DB_M will be given later in Section 5.4.3. As a result, in the presence of a centralized trusted party, an adversary cannot clone a tag more than once, and cloning cannot be performed in a large scale.

Yet, when genuineness verification is performed by potentially malicious readers along the supply chain tag cloning is trickier to address. To tackle tag cloning in this case, we suggest that each partner P_i in the supply chain keeps a database DB_i that contains the identifiers of tags present at P_i's site. Then, we divide time into epochs e_k (typically, the duration of an

epoch e_k is one day) and partners are required to update their databases at the beginning of each epoch e_k.

Now to detect clones, each pair of partners P_i and P_j invoke a protocol for privacy preserving set intersection (44, 45) at the beginning of each epoch e_k, to check whether there is an identifier ID that is present in both of their databases. At the end of the privacy preserving set intersection protocol, both partners obtain a set of identifiers $S_{(i,j)} = \mathsf{DB}_i \cap \mathsf{DB}_j$ that represent the clones in their sites. If $S_{(i,j)} \neq \emptyset$, then P_i and P_j can discard the clones and investigate where the clones come from.

5.3.2 Privacy

We say that a tracking protocol is privacy preserving if it ensures tag unlinkability. As discussed previously, tag unlinkability assures that it is computationally infeasible for an adversary $\mathcal{A}$ to distinguish between tags based on their interactions with readers in the supply chain or based on their interactions with him. In particular, tag unlinkability ensures that a reader R_k in the supply chain cannot trace tags once they leave its site (vicinity).

It is important to note that in this chapter we target passive tags that only feature storage capabilities and thereby cannot perform any computation. Consequently, tags cannot update their states after an interaction with a reader on their own. Hence, the tag state does not change in between two protocol executions. Under such circumstances, it is impossible to provide tag unlinkability against an adversary who tries to link tags in between two subsequent reader interactions. Thus, as explained in 3.4 and in line with previous work on storage-only tags, such as Ateniese et al. (3) and Sadeghi et al. (139), we assume that an adversary cannot permanently access tags or eavesdrop on tags' communications, and therefore, we conjecture that there is at least one interaction between a tag and an honest reader in the supply chain that is unobserved by the adversary.

Similar to Chapter 4, we define tag unlinkability using an indistinguishability-based game that takes place in two phases.

In the learning phase cf. Algorithm 5.3.3, adversary $\mathcal{A}(r, s, \rho, \epsilon)$ calls the oracle $\mathcal{O}_{\mathsf{CorruptR}}$ to corrupt up to r readers R_i. $\mathcal{A}$ is provided then with two challenge tags T_0 and T_1 that just entered the supply chain (tags at step v_0) from the oracle $\mathcal{O}_{\mathsf{Tag}}$. Adversary $\mathcal{A}$ starts iterating the supply chain up to ρ times. Before each iteration of the supply chain, adversary $\mathcal{A}$ reads and writes into the tags T_0 and T_1, then queries the oracle $\mathcal{O}_{\mathsf{Step}}$ to get their next steps in the supply chain. Moreover, adversary $\mathcal{A}$ can query the oracle $\mathcal{O}_{\mathsf{Tag}}$ to get access to s tags $T_{(i,j)}$ that he can read from and write into. He can also query the oracle $\mathcal{O}_{\mathsf{Step}}$ to get $T_{(i,j)}$'s next step in the supply chain. Finally, adversary $\mathcal{A}$ is allowed to iterate the supply chain and to read the states of tags $T_{(i,j)}$.

In the challenge phase, cf. Algorithm 5.3.4, adversary $\mathcal{A}$ is provided with the next step of tags T_0 and T_1, and he is allowed to read and write into T_0 and T_1 one more time. Next, the supply chain is iterated first outside the range of adversary $\mathcal{A}$. That is, tags T_0 and T_1 have

Algorithm 5.3.3: Learning phase of tag unlinkability

for $i := 1$ **to** r **do**
  $\mathsf{Sec}_i \leftarrow \mathcal{O}_{\mathsf{CorruptR}}(R_i)$;
$\mathsf{T}_0 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\text{“tag at step ”}\mathsf{v}_0)$;
$\mathsf{T}_1 \leftarrow \mathcal{O}_{\mathsf{Tag}}(\text{“tag at step ”}\mathsf{v}_0)$;
for $i := 0$ **to** $\rho - 1$ **do**
  $\mathsf{v}_{\mathsf{T}_0}^{i+1} \leftarrow \mathcal{O}_{\mathsf{Step}}(\mathsf{T}_0)$;
  $S_{\mathsf{T}_0}^{i}$:=Read(T_0);
  Write($\mathsf{T}_0, S_{\mathsf{T}_0}^{\prime i}$);
  $\mathsf{v}_{\mathsf{T}_1}^{i+1} \leftarrow \mathcal{O}_{\mathsf{Step}}(\mathsf{T}_1)$;
  $S_{\mathsf{T}_1}^{i}$:=Read(T_1);
  Write($\mathsf{T}_1, S_{\mathsf{T}_1}^{\prime i}$);
  for $j = 1$ **to** s **do**
    $T_{(i,j)} \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{(i,j)})$;
    $\mathsf{v}_{T_{(i,j)}} \leftarrow \mathcal{O}_{\mathsf{Step}}(T_{(i,j)})$;
    $S_{T_{(i,j)}}$:=Read($T_{(i,j)}$);
    Write($T_{(i,j)}, S'_{T_{(i,j)}}$);
  IterateSupplyChain;
  for $j = 1$ **to** s **do**
    Read($T_{(i,j)}$);

one interaction with an honest reader outside the range of $\mathcal{A}$. The oracle $\mathcal{O}_{\mathsf{Flip}}$ then provides adversary $\mathcal{A}$ with tag T_b, $b \in \{0, 1\}$. Now given the data stored into T_b and the result of the different readings, adversary $\mathcal{A}$ returns a guess b' for the bit b to challenger $\mathcal{C}$.

Adversary $\mathcal{A}$ is said to win the tag unlinkability game if **i.)** $b = b'$, **ii.)** the readers associated with steps $\mathsf{v}_{\mathsf{T}_0}^{k+1}$ and $\mathsf{v}_{\mathsf{T}_1}^{k'+1}$ are not corrupted by adversary $\mathcal{A}$.

The advantage ϵ of adversary $\mathcal{A}$ in winning the tag unlinkability game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins}) - \frac{1}{2}$$

Definition 5.3 (Tag Unlinkability)**.** *A product tracking protocol is said to ensure tag unlinkability,* **iff** *for any adversary $\mathcal{A}(r, s, \rho, \epsilon)$, the advantage ϵ in winning the tag unlinkability game is negligible.*

In a real world scenario, the adversary $\mathcal{A}$ against the above game corresponds to a set of r supply chain partners $\{P_1, P_2, ..., P_r\}$ that collude in order to compromise the privacy of another partner P_i by eavesdropping on and tampering with tags in the supply chain.

Remark 5.3. *The adversary $\mathcal{A}$ defined above is a narrow adversary according to Vaudenay (159). That is, $\mathcal{A}$ does not have access to the output of the* Check *function. Note that if we allow adversary $\mathcal{A}$ to access to the output of the* Check *function, then he can mount a trivial*

Algorithm 5.3.4: Challenge phase of tag unlinkability

$\mathsf{v}_{\mathsf{T}_0}^{k+1} \leftarrow \mathcal{O}_{\mathsf{Step}}(\mathsf{T}_0)$;
$S_{\mathsf{T}_0}^{k}$:=Read(T_0);
Write($\mathsf{T}_0, S_{\mathsf{T}_0}^{\prime k}$);
$\mathsf{v}_{\mathsf{T}_1}^{k'+1} \leftarrow \mathcal{O}_{\mathsf{Step}}(\mathsf{T}_1)$;
$S_{\mathsf{T}_1}^{k'}$:=Read(T_1);
Write($\mathsf{T}_1, S_{\mathsf{T}_1}^{\prime k'}$);
IterateSupplyChain; // Challenger $\mathcal{C}$ iterates the supply chain outside the range of $\mathcal{A}$
$\mathsf{T}_b \leftarrow \mathcal{O}_{\mathsf{Flip}}\{\mathsf{T}_0, \mathsf{T}_1\}$;
S_{T_b} :=Read(T_b);
Output b';

attack where he writes "dummy data" into some tag T. Now since tag T will not be accepted by any verifier V_k in the supply chain with an overwhelming probability, i.e., $\mathsf{Check}(S_T, V_k) = 0$*, it follows that adversary $\mathcal{A}$ can always distinguish T from legitimate tags.*

5.4 TRACKER: Product Tracking by a Trusted Party

Here we present our first protocol for product tracking called TRACKER. TRACKER relies on a trusted party called manager M to verify the genuineness of tags in the supply chain. Using the notations of Section 5.2, this means that $\mathcal{V} = \{M\}$. We recall that genuineness verification of tags is carried out by verifying the sequence of steps that tags have taken. Hence, a tag T in TRACKER stores a state S_T^j that encodes the path in the supply chain that T went through. The underpinning idea of TRACKER is to encode different paths in the supply chain using different *polynomials*. More precisely, a path $\mathcal{P}$ in the supply chain is represented by the evaluation of unique polynomial $Q_{\mathcal{P}} \in \mathbb{F}_q[X]$ in a fixed value x_0, offering thus a compact and efficient encoding of paths.

Now, TRACKER relies on the property that for any two *different* paths $\mathcal{P} \neq \mathcal{P}'$, valid or not, the equation $Q_{\mathcal{P}}(x_0) = Q_{\mathcal{P}'}(x_0)$ holds only with negligible probability when q is large enough and x_0 is a generator of $\mathbb{F}_q^*$. Two different paths will result in two different polynomial evaluations, and therefore, the state of a tag T at the end of the supply chain can be uniquely mapped to one single (valid) path.

However, the path representation as introduced above does not prevent path cloning, i.e., copying the path of a valid tag into a fake tag and then injecting the fake tag in the supply chain. To tackle this issue, tags in TRACKER stores a path signature $\sigma_{\mathcal{P}}(\mathsf{ID})$ defined as $\sigma_{\mathcal{P}}(\mathsf{ID}) = H(\mathsf{ID})^{Q_{\mathcal{P}}(x_0)}$ instead of $Q_{\mathcal{P}}(x_0)$, where H is some cryptographic hash function. The path signature corresponds hence to the tag's identifier signed by the path encoding. By construction, valid path signatures prove that tags are issued by a legitimate authority, and that they went through valid paths in the supply chain.

TRACKER can be structured into three parts: **1.)** Issuer I writes an initial state S_T^0 into a new tag T. **2.)** Readers R_k in the supply chain update the path signature stored into tag T by applying simple arithmetic operations represented by an update function denoted f_{R_k} on T's current state S_T^j. Eventually, this results in the evaluation of the $\sigma_{\mathcal{P}_{\text{valid}_i}} = H(\mathsf{ID})^{Q_{\mathcal{P}_{\text{valid}_i}}(x_0)}$. **3.)** Finally, manager M checks whether T's state S_T^j matches one of the ν evaluations of valid polynomials $Q_{\mathcal{P}_{\text{valid}_i}}(x_0)$. If so, manager M accepts tag T and identifies the valid path that tag T has taken.

Privacy and security overview. On the one hand, to protect tag privacy in TRACKER, each tag stores probabilistic elliptic curve Elgamal encryptions of its state $S_T = (\mathsf{ID}, H(\mathsf{ID}), \sigma_{\mathcal{P}}($ $\mathsf{ID}))$, and readers use homomorphic (re-)encryption techniques to update the path signature stored in tags without decryption. At the end of the supply chain, the manager M can then decrypt and verify the validity of the path.

On the other hand, security of TRACKER relies on the computational Diffie-Hellman assumption (cf. Definition 2.27). In fact, we show that if there is an adversary $\mathcal{A}$ who is able to compute a valid encrypted state $S_T = (\mathsf{ID}, H(\mathsf{ID}), \sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID}))$, then this adversary will be able to break the computational Diffie-Hellman (CDH) assumption, see Definition 2.27.

Before the detailed protocol description in Section 5.4.3, we first provide an overview of TRACKER's polynomial path encoding.

5.4.1 Path Encoding

TRACKER's polynomial path encoding is based on techniques for software fault detection that were proposed by Noubir et al. (121). The idea is to map each path $\mathcal{P}$ in the supply chain to some polynomial $Q_{\mathcal{P}} \in \mathbb{F}_q[X]$, where q is a prime number. To this end, each step $\mathsf{v}_k, 0 \leq k \leq \eta$, in the supply chain is associated with a unique random number $a_k \in \mathbb{F}_q$.

Now each path in the supply chain is represented by a polynomial in $\mathbb{F}_q$. The polynomial corresponding to path $\mathcal{P} = \overrightarrow{\mathsf{v}_0\mathsf{v}_1 \ldots \mathsf{v}_l}$ is defined as follows:

$$Q_{\mathcal{P}}(x) = a_0 x^l + \sum_{k=1}^{l} a_k x^{l-k} \tag{5.1}$$

To have a more compact representation of paths, a path $\mathcal{P}$ is encoded as the evaluation of $Q_{\mathcal{P}}(x)$ in x_0, where x_0 is a generator of $\mathbb{F}_q^*$. We denote $\phi(\mathcal{P}) = Q_{\mathcal{P}}(x_0)$ the polynomial-based path encoding of path $\mathcal{P}$.

It is noteworthy that when the coefficient $a_k \in \mathbb{F}_q$ are randomly chosen and q is large enough, the above path encoding has the desired property that for any two different paths $\mathcal{P}$ and $\mathcal{P}'$, $\phi(\mathcal{P}) \neq \phi(\mathcal{P}')$ with an overwhelming probability. In fact, Noubir et al. (121) proved

the following result.

$$\forall \mathcal{P}, \mathcal{P}' \text{ with } \mathcal{P} \neq \mathcal{P}', \text{ the equation } \phi(\mathcal{P}) = \phi(\mathcal{P}') \text{ holds with probability } \frac{1}{q}.$$

We also note that for any path $\mathcal{P}$ and for any step v_k in the supply chain, the following equation always holds.

$$\phi(\overrightarrow{\mathcal{P}\mathsf{v}_k}) = x_0\phi(\mathcal{P}) + a_k$$

5.4.2 Path Signature

Let T be a tag that took path $\mathcal{P}$. We define T's *path signature* as:

$$\sigma_{\mathcal{P}}(\mathsf{ID}) = H(\mathsf{ID})^{\phi(\mathcal{P})}$$

Where ID is T's unique identifier and H is a cryptographic hash function. Therefore, the path signature defined above depends on tags' ID to prevent an adversary from copying the path signature of one tag into another one.

Note that $\sigma_{\mathcal{P}}(\mathsf{ID})$ is a signature of ID using the secret key $\phi(\mathcal{P})$. More precisely, it is an aggregate signature using the secret coefficients a_k of readers R_k in the path $\mathcal{P}$.

5.4.2.1 Reader Computation

A reader that is visited by some tag T, reads T's current path signature, updates it, and writes the updated path signature into T. To eventually achieve the evaluation of path signature $\sigma_{\mathcal{P}_l}(\mathsf{ID})$ of path $\mathcal{P}_l = \overrightarrow{\mathsf{v}_0\mathsf{v}_1 \ldots \mathsf{v}_{k-1}\mathsf{v}_k\mathsf{v}_{k+1} \ldots \mathsf{v}_l}$, the per reader effort is quite low. Assume that T arrives at reader R_k, i.e., step v_k in the supply chain. So far, T went through (sub-)path $\mathcal{P}_{k-1} = \overrightarrow{\mathsf{v}_0\mathsf{v}_1 \ldots \mathsf{v}_{k-1}}$, and stores ID, $H(\mathsf{ID})$, and path signature $\sigma_{\mathcal{P}_{k-1}}(\mathsf{ID})$.

To get $\sigma_{\mathcal{P}_k}(\mathsf{ID})$, reader R_k simply computes its state transition function f_{R_k} defined as:

$$f_{R_k}(x, y) := x^{x_0} + y^{a_k}$$

In fact,

$$\begin{aligned} f_{R_k}(\sigma_{\mathcal{P}_{k-1}}(\mathsf{ID}), H(\mathsf{ID})) &= \sigma_{\mathcal{P}_{k-1}}(\mathsf{ID})^{x_0}\ H(\mathsf{ID})^{a_k} = H(\mathsf{ID})^{\phi(\mathcal{P}_{k-1})x_0}\ H(\mathsf{ID})^{a_k} \\ &= H(\mathsf{ID})^{x_0\phi(\mathcal{P}_{k-1})+a_k} = H(\mathsf{ID})^{\phi(\overrightarrow{\mathcal{P}_{k-1}\mathsf{v}_k})} = H(\mathsf{ID})^{\phi(\mathcal{P}_k)} \\ &= \sigma_{\mathcal{P}_k}(\mathsf{ID}) \end{aligned} \tag{5.2}$$

Reader R_k then writes $\sigma_{\mathcal{P}_k}(\mathsf{ID})$ in tag T.

5.4.2.2 Tag State Decoding

This operation corresponds to the Check function of the TRACKER protocol.

To verify the genuineness of tags in the supply chain manager M stores a list of all possible valid paths $\mathcal{P}_{\text{valid}_i}$ together with their corresponding verification keys $K^i = \phi(\mathcal{P}_{\text{valid}_i})$.

Now when manager M reads the state $S_T^j = (\mathsf{ID}, H(\mathsf{ID}), \sigma_{\mathcal{P}}(\mathsf{ID}))$ of some tag T in the supply chain that went through a path $\mathcal{P}$, he first computes $H(\mathsf{ID})$ and verifies the second element of T's state. If T passes the verification, manager M checks whether there exists a verification key $K^i \in \mathcal{K}_V$ that verifies the following equation:

$$\sigma_{\mathcal{P}}(\mathsf{ID}) = H(\mathsf{ID})^{K^i} = H(\mathsf{ID})^{\phi(\mathcal{P}_{\text{valid}_i})}$$

5.4.3 TRACKER

TRACKER consists of an initial setup phase, the preparation of new tags entering the supply chain, reader and tag interaction as part of the supply chain, and finally a path verification conducted by manager M.

- **Setup:** A trusted third party (TTP) sets up an *elliptic curve Elgamal cryptosystem* and generates the secret key sk and the corresponding public key $\mathsf{pk} = (g, \tilde{g} = g^{\mathsf{sk}})$ such that the order of g is a large prime q, ($|q| = 160$ bits). Without loss of generality, we denote $\mathbb{G} = \langle g \rangle$.

  Then, it selects a generator x_0 of the finite field $\mathbb{F}_q$, and generates $\eta+1$ random numbers $a_k \in \mathbb{F}_q,\ 0 \leq k \leq \eta$.

  Through a secure channel, the TTP sends to each reader R_k, representing step v_k the tuple (x_0, a_k, pk), while providing issuer I with the tuple (x_0, a_0, pk). Finally, it supplies manager M with the secret key sk, the generator x_0 and the tuples (k, a_k).

  Now, manager M is informed which reader R_k at step v_k knows which a_k. As manager M knows which paths in the supply chain will be valid, he now computes his set of verification keys $\mathcal{K}_V = \{K^1, K^2, ..., K^\nu\}$. Each verification key K^i is computed as the encoding of a valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain using Equation (5.1). That is,

  $$K^i = \phi(\mathcal{P}_{\text{valid}_i})$$

  Finally, manager M stores the pairs (K^i, steps), where steps is the sequence of steps composing the path $\mathcal{P}_{\text{valid}_i}$. Accordingly, manager M can verify the validity of the path that a tag took, and if the path is valid he can identify it.

- **Tag initialization:** For each new tag T entering the supply chain, issuer I draws a random point $\mathsf{ID} \in \mathcal{E}$ which is T's unique identifier. Now, let $H : \{0,1\}^* \rightarrow \mathbb{G}$ be a cryptographic hash function[6]. H will be viewed in the rest of this section as a random oracle.

[6]The hash function H can be computed using the algorithm proposed by Brier et al. (28).

Provided with the secret coefficient a_0, issuer I computes

$$\sigma_{\mathsf{v}_0} = H(\mathsf{ID})^{a_0}$$

Next, he selects three random numbers $r^0_{\mathsf{ID}}, r^0_H, r^0_\sigma \in \mathbb{F}_q$ to compute the following ciphertexts:

$$\begin{aligned} c^0_{\mathsf{ID}} &= \mathsf{Enc}_{\mathsf{pk}}(\mathsf{ID}) = (u^0_{\mathsf{ID}}, v^0_{\mathsf{ID}}) = (g^{r^0_{\mathsf{ID}}}, \mathsf{ID}\tilde{g}^{r^0_{\mathsf{ID}}}) \\ c^0_H &= \mathsf{Enc}_{\mathsf{pk}}(H(\mathsf{ID})) = (u^0_H, v^0_H) = (g^{r^0_H}, H(\mathsf{ID})\tilde{g}^{r^0_H}) \\ c^0_\sigma &= \mathsf{Enc}_{\mathsf{pk}}(\sigma_{\mathsf{v}_0}) = (u^0_\sigma, v^0_\sigma) = (g^{r^0_\sigma}, H(\mathsf{ID})^{a_0}\tilde{g}^{r^0_\sigma}) \end{aligned}$$

Finally, he writes the state $S^0_T = (c^0_{\mathsf{ID}}, c^0_H, c^0_\sigma)$ into tag T which can now enter the supply chain.

- **Tag state update by readers:** Assume a tag T arrives at reader R_k that is associated with step v_k in the supply chain. Reader R_k reads out T's current state $S^j_T = (c^j_{\mathsf{ID}}, c^j_H, c^j_\sigma)$. Without loss of generality, we assume that the path that tag T took so far is $\mathcal{P}$.

  Given the ciphertexts $c^j_H = (u^j_H, v^j_H)$, $c^j_\sigma = (u^j_\sigma, v^j_\sigma)$, generator x_0, and a_k, reader R_k computes $c^{j+1}_\sigma = (u^{j+1}_\sigma, v^{j+1}_\sigma)$ as follows:

$$\begin{aligned} u^{j+1}_\sigma &= f_{R_k}(u^j_\sigma, u^j_H) = (u^j_\sigma)^{x_0}(u^j_H)^{a_k} \\ &= g^{x_0 r^j_\sigma} g^{a_k r^j_H} = g^{x_0 r^j_\sigma + a_k r^j_H} = g^{r^{j+1}_\sigma} \\ v^{j+1}_\sigma &= f_{R_k}(v^j_\sigma, v^j_H) = (v^j_\sigma)^{x_0}(v^j_H)^{a_k} \\ &= \left(H(\mathsf{ID})^{\phi(\mathcal{P})}\tilde{g}^{r^j_\sigma}\right)^{x_0}\left(H(\mathsf{ID})\tilde{g}^{r^j_H}\right)^{a_k} \\ &= H(\mathsf{ID})^{x_0\phi(\mathcal{P})}\tilde{g}^{x_0 r^j_\sigma} H(\mathsf{ID})^{a_k}\tilde{g}^{a_k r^j_H} \\ &= H(\mathsf{ID})^{x_0\phi(\mathcal{P})} H(\mathsf{ID})^{a_k}\tilde{g}^{x_0 r^j_\sigma}\tilde{g}^{a_k r^j_H} \\ &= H(\mathsf{ID})^{(x_0\phi(\mathcal{P})+a_k)}\tilde{g}^{x_0 r^j_\sigma + a_k r^j_H} \\ &= H(\mathsf{ID})^{\phi(\overrightarrow{\mathcal{P}\mathsf{v}_k})}\tilde{g}^{r^{j+1}_\sigma} \\ &= \sigma_{\overrightarrow{\mathcal{P}\mathsf{v}_k}}(\mathsf{ID})\tilde{g}^{r^{j+1}_\sigma} \end{aligned}$$

  To get c^{j+1}_{ID} and c^{j+1}_H, reader R_k re-encrypts c^j_{ID} and c^j_H respectively: it picks randomly two numbers r'_{ID} and $r'_H \in \mathbb{F}_q$ and outputs two new ciphertexts $c^{j+1}_{\mathsf{ID}} = (u^{j+1}_{\mathsf{ID}}, v^{j+1}_{\mathsf{ID}}) = (g^{r'_{\mathsf{ID}}} u^j_{\mathsf{ID}}, \tilde{g}^{r'_{\mathsf{ID}}} v^j_{\mathsf{ID}})$ and $c^{j+1}_H = (u^{j+1}_H, v^{j+1}_H) = (g^{r'_H} u^j_H, \tilde{g}^{r'_H} v^j_H)$.

  The reader also re-encrypts c^{j+1}_σ. It picks randomly $r'_\sigma \in \mathbb{F}_q$ and outputs: $c'^{j+1}_\sigma = (u'^{j+1}_\sigma, v'^{j+1}_\sigma) = (g^{r'_\sigma} u^{j+1}_\sigma, \tilde{g}^{r'_\sigma} v^{j+1}_\sigma)$. Finally, reader R_k writes the new state $S^{j+1}_T = (c^{j+1}_{\mathsf{ID}}, c^{j+1}_H, c'^{j+1}_\sigma)$ into tag T.

- **Path verification by manager M:** This operation corresponds to TRACKER's realization of the Check function. Upon reading the state $S_T^l = (c_{\mathsf{ID}}^l, c_H^l, c_\sigma^l)$ stored into tag T, manager M decrypts c_{ID}^l and gets $\mathsf{ID} \in \mathbb{G}$. Manager M checks first for cloning by looking up ID in his database DB_M. If $\mathsf{ID} \in \mathsf{DB}_M$, then manager M outputs 0 and rejects tag T.

  Otherwise, manager M decrypts c_H^l, gets a point $g' \in \mathcal{E}$ and verifies whether the equation $g' = H(\mathsf{ID})$ holds. If it does not, manager M outputs 0 and rejects tag T. If $g' = H(\mathsf{ID})$, then manager M decrypts c_σ^l which results in another point $\tilde{\sigma}$. Given $H(\mathsf{ID})$, manager M verifies whether there exists $K^i \in \mathcal{K}_V$ such that

$$\tilde{\sigma} = H(\mathsf{ID})^{K^i} = H(\mathsf{ID})^{\phi(\mathcal{P}_{\text{valid}_i})}$$

  If it is not the case, manager M outputs 0 and rejects the tag T. Otherwise, manager M outputs 1 and adds ID to his database DB_M.

5.4.4 Security Analysis

In this section, we present the main security theorems regarding TRACKER.

Theorem 5.1. TRACKER *is complete.*

Proof. We note that if a tag T went through a valid path $\mathcal{P}_{\text{valid}_i}$, then T will store a state $S_T = (c_{\mathsf{ID}}, c_H, c_\sigma)$ such that:

$$\begin{aligned} c_{\mathsf{ID}} &= \mathsf{Enc}_{\mathsf{pk}}(\mathsf{ID}) \\ c_H &= \mathsf{Enc}_{\mathsf{pk}}(H(\mathsf{ID})) \\ c_\sigma &= \mathsf{Enc}_{\mathsf{pk}}(\sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID})) = \mathsf{Enc}_{\mathsf{pk}}(H(\mathsf{ID})^{\phi(\mathcal{P}_{\text{valid}_i})}) \end{aligned}$$

When manager M decrypts the state S_T, he obtains the tuple $(\mathsf{ID}, H(\mathsf{ID}), \sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID}))$. Now it is clear that for $K^i = \phi(\mathcal{P}_{\text{valid}_i})$, the equation $H(\mathsf{ID})^{K^i} = \sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID})$ holds, leading the check function to output "1". □

Theorem 5.2. TRACKER *is sound under the CDH assumption in* $\mathbb{G}$ *in the random oracle model.*

Proof. Assume there is an adversary $\mathcal{A}$ who breaks the security of TRACKER with a non-negligible advantage ϵ, we build an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to break the CDH assumption with a non-negligible advantage ϵ'.

Let $\mathcal{O}_{\mathsf{CDH}}$ be an oracle that selects randomly $x, y \in \mathbb{F}_q$, and returns $g, g^x, g^y \in \mathbb{G}$.

Proof overview. If adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the security of TRACKER, then adversary $\mathcal{A}$ will be able to output a challenge tag T_c that stores an encrypted state S_{T_c}, such that:

i.) $\mathsf{Check}(S_{\mathsf{T}_c}, M) = 1$, i.e., there is a valid path $\mathcal{P}_{\text{valid}_i}$ that corresponds to T_c's state;

ii.) $\exists\ \mathsf{v}_k \in \mathcal{P}_{\text{valid}_i}$ such that step v_k is not corrupted by adversary $\mathcal{A}$;

iii.) T_c did not go through step v_k.

To break the CDH assumption, adversary $\mathcal{B}$ simulates a TRACKER system for $\mathcal{A}$ where he creates a step v_k in the supply chain such that $\mathsf{Sec}_k = (x_0, g^x)$ instead of $\mathsf{Sec}_k = (x_0, a_k)$.

Without loss of generality, we assume in the rest of the proof that $\mathsf{v}_k = \mathsf{v}_0$ and that adversary $\mathcal{A}$ corrupts all readers in the supply chain.

Now, adversary $\mathcal{B}$ must convince adversary $\mathcal{A}$ that v_0 is associated with secret coefficient $a_0 = x$ that corresponds to g^x received from the oracle $\mathcal{O}_{\mathsf{CDH}}$. That is, adversary $\mathcal{B}$ has to be able to compute $H(\mathsf{ID})^x$ only by knowing g^x. To this end, adversary $\mathcal{B}$ simulates a random oracle $\mathcal{H}$ to compute the hash function H.

When $\mathcal{H}$ is queried in the learning phase with identifier ID_j, $\mathcal{B}$ picks a random number r_j and computes $H(\mathsf{ID}_j) = g^{r_j}$.

When adversary $\mathcal{A}$ queries the random oracle $\mathcal{H}$ with the identifier ID_c of the challenge tag T_c, adversary $\mathcal{B}$ simulates $\mathcal{H}$ by picking a random number r_c and computing $H(\mathsf{ID}_c) = g^{yr_c}$.

In the challenge phase, adversary $\mathcal{A}$ returns the challenge tag T_c to $\mathcal{B}$.

As adversary $\mathcal{A}$ has a non-negligible advantage in winning the soundness game, it follows that the challenge tag T_c stores an encrypted valid state that corresponds to the tuple $(\mathsf{ID}_c, H(\mathsf{ID}_c), \sigma_c)$ such that $\sigma_c = H(\mathsf{ID}_c)^{\phi(\mathcal{P}_{\text{valid}_i})}$, while T_c did not go through the step v_0.

We assume that tag T_c stores a state S_{T_c} that corresponds to the valid path $\mathcal{P}_{\text{valid}_i} = \overrightarrow{\mathsf{v}_0 \mathcal{P}'_{\text{valid}_i}}$, and we denote l the length of path $\mathcal{P}_{\text{valid}_i}$.

By definition, $\phi(\mathcal{P}_{\text{valid}_i}) = a_0 x_0^l + \phi(\mathcal{P}'_{\text{valid}_i}) = x x_0^l + \phi(\mathcal{P}'_{\text{valid}_i})$, and given σ_c and the encoding $\phi(\mathcal{P}'_{\text{valid}_i})$ of the sub-path $\mathcal{P}'_{\text{valid}_i}$, adversary $\mathcal{B}$ computes:

$$\begin{aligned} \frac{\sigma_c}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} &= \frac{H(\mathsf{ID}_c)^{\phi(\mathcal{P}_{\text{valid}_i})}}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} = H(\mathsf{ID}_c)^{x x_0^l} \\ H(\mathsf{ID}_c)^x &= \left(\frac{\sigma_c}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} \right)^{\frac{1}{x_0^l}} \end{aligned}$$

Adversary $\mathcal{B}$ thus has access to $H(\mathsf{ID}_c)^x = (g^{yr_c})^x = g^{xyr_c}$, and he can compute $(g^{xyr_c})^{\frac{1}{r_c}} = g^{xy}$. This breaks the CDH assumption leading to a contradiction.

Simulation of the random oracle $\mathcal{H}$. To respond to the queries of the random oracle $\mathcal{H}$, the adversary $\mathcal{B}$ keeps a table T_H of tuples $(\mathsf{ID}_j, r_j, \mathsf{coin}(\mathsf{ID}_j), H(\mathsf{ID}_j))$ as explained below.

On a query $H(\mathsf{ID}_i)$, adversary $\mathcal{B}$ replies as follows:

1. If there is a tuple $(\mathsf{ID}_i, r_i, \mathsf{coin}(\mathsf{ID}_i), H(\mathsf{ID}_i))$ that corresponds to ID_i, then $\mathcal{B}$ returns $H(\mathsf{ID}_i)$.

2. If ID_i has never been queried before, then $\mathcal{B}$ picks a random number $r_i \in \mathbb{F}_q$ and flips a random coin $\mathsf{coin}(\mathsf{ID}_i) \in \{0,1\}$ such that: $\mathsf{coin}(\mathsf{ID}_i) = 1$ with probability p, and it equals to 0 with probability $1-p$. If $\mathsf{coin}(\mathsf{ID}_i) = 0$, then $\mathcal{B}$ answers with $H(\mathsf{ID}_i) = g^{r_i}$. Otherwise, he answers with $H(\mathsf{ID}_i) = (g^y)^{r_i}$. Finally, he stores the tuple $(\mathsf{ID}_i, r_i, \mathsf{coin}(\mathsf{ID}_i), H(\mathsf{ID}_i))$ in table T_H.

Construction. First, adversary $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{CDH}}$ to receive $g, g^x, g^y \in \mathbb{G}$. Then, adversary $\mathcal{B}$ simulates the challenger $\mathcal{C}$:

- Adversary $\mathcal{B}$ generates a pair of matching Elgamal public and secret keys $(\mathsf{sk}, \mathsf{pk})$. Then, he generates η random coefficients a_k.
- Next, he provides each reader R_k in TRACKER with the pair $\mathsf{Sec}_k = (x_0, a_k)$.
- He provides the issuer I with the pair (x_0, g^x), as if $a_0 = x$.
- Instead of computing the verification keys K^i as the encoding of valid paths in the supply chain $\phi(\mathcal{P}_{\text{valid}_i})$, adversary $\mathcal{B}$ computes $K^i = g^{\phi(\mathcal{P}_{\text{valid}_i})}$.

  Without loss of generality, a valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain could be represented as $\mathcal{P}_{\text{valid}_i} = \overrightarrow{\mathsf{v}_0 \mathcal{P}'_{\text{valid}_i}}$. Thus, $g^{\phi(\mathcal{P}_{\text{valid}_i})} = g^{x x_0^l + \phi(\mathcal{P}'_{\text{valid}_i})}$, where l is the length of path $\mathcal{P}_{\text{valid}_i}$.

  Once K^i are computed for all the valid paths in the supply chain, $\mathcal{B}$ provides the pairs (K^i, steps) to the manager M.
- $\mathcal{B}$ simulates the issuer I and creates n tags T_j of TRACKER. For each tag T_j, $\mathcal{B}$ selects randomly $\mathsf{ID}_j \in \mathbb{G}$ and simulates the random oracle $\mathcal{H}$ to get the tuple $(\mathsf{ID}_j, r_j, \mathsf{coin}(\mathsf{ID}_j), H(\mathsf{ID}_j))$.

  If $\mathsf{coin}(\mathsf{ID}_j) = 1$, i.e., $H(\mathsf{ID}_j) = g^{y r_j}$, then $\mathcal{B}$ cannot compute $H(\mathsf{ID}_j)^x = g^{xyr_j}$ as he does not know both x and y. Consequently, $\mathcal{B}$ stops the soundness game.

  Otherwise, using r_j, adversary $\mathcal{B}$ computes $H(\mathsf{ID}_j)^x = (g^x)^{r_j}$.

  Finally, adversary $\mathcal{B}$ encrypts the tuple $(\mathsf{ID}_j, H(\mathsf{ID}_j), \sigma_{\mathsf{v}_0}(\mathsf{ID}_j))$ using the public key pk of Elgamal cryptosystem. $\mathcal{B}$ stores the resulting ciphertexts $(c^0_{\mathsf{ID}_j}, c^0_{H_j}, c^0_{\sigma_j})$ into tag T_j.

Learning phase. $\mathcal{B}$ then calls adversary $\mathcal{A}$ and simulates the challenger $\mathcal{C}$ as follows.

- Adversary $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathsf{CorruptR}}$ for $\mathcal{A}$. For ease of understanding, we assume that adversary $\mathcal{A}$ corrupts all readers R_k in the supply chain.
- Adversary $\mathcal{B}$ simulates readers R_k along the supply chain. Let T_j be a tag which arrives at step v_k. $\mathcal{B}$ updates the state of tag T_j using the secret coefficient a_k and Elgamal public key pk.

- Adversary $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathsf{Check}}$. Let T_j be a tag that went through some path $\mathcal{P}$ in the supply chain. Tag T_j stores a state $S_{T_j} = (c_{\mathsf{ID}_j}, c_{H_j}, c_{\sigma_j})$.

  $\mathcal{B}$ first decrypts the state of tag T_j and gets a tuple of points $(\mathsf{ID}_j, g'_j, \tilde{\sigma}_j)$. He then looks up ID_j in T_H to retrieve $(\mathsf{ID}_j, r_j, \mathsf{coin}(\mathsf{ID}_j), H(\mathsf{ID}_j))$, verifies whether $H(\mathsf{ID}_j) = g'_j$, and finally, checks whether there is a valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain such that $\tilde{\sigma}_j = (K^i)^{r_j}$ and $K^i = g^{\phi(\mathcal{P}_{\text{valid}_i})}$.

  Note. Here, we assume that $\mathsf{coin}(\mathsf{ID}_j) = 0$ for ease of understanding. Otherwise, adversary $\mathcal{B}$ has to stop the soundness game whenever $\mathsf{coin}(\mathsf{ID}_j) = 1$, as he cannot verify the validity of the path that tag T_j took.

Challenge phase. Adversary $\mathcal{A}$ outputs a tag T_c.

Since adversary $\mathcal{A}$ has a non-negligible advantage in the soundness game, it follows that **i.)** $\mathsf{Check}(S_{\mathsf{T}_c}, M) = 1$, and **ii.)** T_c did not go through step v_0.

Without loss of generality, we assume that the state of tag T_c corresponds to the tuple $(\mathsf{ID}_c, H(\mathsf{ID}_c), \sigma_c)$, and that T_c's path signature σ_c corresponds to path $\mathcal{P}_{\text{valid}_i} = \overrightarrow{\mathsf{v}_0 \mathcal{P}'_{\text{valid}_i}}$.

First, $\mathcal{B}$ checks whether $\mathsf{coin}(\mathsf{ID}_c) = 1$ or not.

If $\mathsf{coin}(\mathsf{ID}_c) = 0$, then $\mathcal{B}$ stops the game. Notice that if $H(\mathsf{ID}_c) = g^{r_c}$, $\mathcal{B}$ will not be able to break the CDH assumption.

If $\mathsf{coin}(\mathsf{ID}_c) = 1$, i.e., $H(\mathsf{ID}_c) = g^{yr_c}$, then $\mathcal{B}$ continues the game, and computes g^{xy}.

Let l denote the length of path $\mathcal{P}_{\text{valid}_i}$. Accordingly,

$$\begin{aligned}
\phi(\mathcal{P}_{\text{valid}_i}) &= a_0 x_0^l + \phi(\mathcal{P}'_{\text{valid}_i}) = x x_0^l + \phi(\mathcal{P}'_{\text{valid}_i}) \\
H(\mathsf{ID}_c)^{x x_0^l} &= \frac{\sigma_c}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} = \frac{H(\mathsf{ID}_c)^{\phi(\mathcal{P}_{\text{valid}_i})}}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} \\
H(\mathsf{ID}_c)^{x} &= \left(\frac{\sigma_c}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} \right)^{\frac{1}{x_0^l}} = (g^{yr_c})^x = g^{xyr_c}
\end{aligned}$$

Provided with the random number r_c, adversary $\mathcal{B}$ finally computes g^{xy}.

Here we compute the advantage ϵ' of $\mathcal{B}$. We indicate that without knowing the value of x, adversary $\mathcal{B}$ cannot identify the valid path that the state of the challenge tag T_c encodes. As a result, $\mathcal{B}$ picks randomly a valid path $\mathcal{P}_{\text{valid}_i}$ from his set of ν valid paths, and he succeeds in breaking the CDH assumption only if, **1.)** his guess of the valid path that the state of tag T_c encodes is correct and if **2.)** he does not stop the soundness game.

1.) Adversary $\mathcal{B}$ makes a correct guess of the valid path that the state of tag T_c encodes with probability $\frac{1}{\nu}$.

2.) Adversary $\mathcal{B}$ stops the soundness game in the learning phase, if during the initialization phase of the n tags in TRACKER, there is a tag T_j with identifier ID_j such that $\mathsf{coin}(\mathsf{ID}_j) =$

1. This event occurs with probability p. Hence, the probability that $\mathcal{B}$ does not stop the soundness game in the learning phase is: $(1-p)^n$.

3.) Adversary $\mathcal{B}$ does not stop the game during the challenge phase, if $\mathsf{coin}(\mathsf{ID}_c) = 1$, which occurs with probability p.

Let E denote the event: $\mathcal{B}$ does not stop the soundness game.

Let E_1 denote the event: $\mathcal{B}$ does not stop the soundness game in the learning phase, $Pr(E_1) = (1-p)^n$.

Let E_2 denote the event: $\mathcal{B}$ does not stop the soundness game in the challenge phase, $Pr(E_2) = p$. Hence,

$$\begin{aligned} \pi &= Pr(E) = Pr(E_1)Pr(E_2) \\ &= p(1-p)^n \end{aligned}$$

Now, if adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the security of Tracker, then adversary $\mathcal{B}$ can break the CDH assumption with a non-negligible advantage $\epsilon' = \frac{\pi}{\nu}\epsilon$, leading to a contradiction.

Note that π is maximal when $p = \frac{1}{n}$ and $\pi_{\mathsf{max}} = \frac{\left(1-\frac{1}{n}\right)^n}{n} \simeq \frac{1}{en}$. □

5.4.5 Privacy Analysis

In this section, we prove that Tracker ensures tag unlinkability under the DDH assumption (see Definition 2.29).

Theorem 5.3 (Tag Unlinkability)**.** Tracker *ensures tag unlinkability under the DDH assumption.*

Proof. Assume there is an adversary $\mathcal{A}$ whose advantage ϵ in winning the tag unlinkability game is non-negligible. We below construct a new adversary $\mathcal{B}$ that executes $\mathcal{A}$ and breaks the DDH assumption in $\mathbb{G} = \langle g \rangle$ with a non-negligible advantage ϵ'.

Let $\mathcal{O}_{\mathsf{DDH}}$ be an oracle that when queried selects two random elements $x, y \in \mathbb{F}_q$ and flips a fair coin $b \in \{0,1\}$. If $b = 1$, then $\mathcal{O}_{\mathsf{DDH}}$ sets $z = xy$; otherwise it randomly selects z from $\mathbb{F}_q$. Finally, it returns the tuple (g, g^x, g^y, g^z).

To break the DDH assumption in $\mathbb{G}$, adversary $\mathcal{B}$ proceeds as follows:

He queries the oracle $\mathcal{O}_{\mathsf{DDH}}$ and gets the tuple (g, g^x, g^y, g^z). Then, he simulates challenger $\mathcal{C}$ and creates a supply chain for the Tracker protocol where the public key of Elgamal is defined as $\mathsf{pk} = (g, \tilde{g} = g^x)$.

Learning phase. He calls adversary $\mathcal{A}$ who enters the learning phase of the tag unlinkability game.

- Adversary $\mathcal{A}$ queries the oracle $\mathcal{O}_{\mathsf{CorruptR}}$ with the identity of r readers R_k in the supply chain. $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathsf{CorruptR}}$ and returns to adversary $\mathcal{A}$ the secret information of readers R_k defined as $\mathsf{Sec}_k = (x_0, a_k)$.

- Simulating the oracle $\mathcal{O}_{\mathsf{Tag}}$, adversary $\mathcal{B}$ supplies adversary $\mathcal{A}$ with two challenge tags T_0 and T_1 that have just been issued by issuer I (i.e., T_0 and T_1 have just entered the supply chain).

- Adversary $\mathcal{A}$ iterates the supply chain ρ times. Before each iteration j of the supply chain:

  1.) $\mathcal{A}$ reads and writes into tags T_0 and T_1.

  2.) Simulating the oracle $\mathcal{O}_{\mathsf{Step}}$, adversary $\mathcal{B}$ provides $\mathcal{A}$ with the next step of tags T_0 and T_1.

  3.) $\mathcal{B}$ simulates the oracles $\mathcal{O}_{\mathsf{Tag}}$ and $\mathcal{O}_{\mathsf{Step}}$ and supplies $\mathcal{A}$ with s tags $T_{(i,j)}$ together with their next step $\mathsf{v}_{T_{(i,j)}}$ in the supply chain.

Challenge phase.

- Adversary $\mathcal{B}$ simulates the oracles $\mathcal{O}_{\mathsf{Step}}$ and provides adversary $\mathcal{A}$ with the next steps of tags T_0 and T_1. Then, he iterates the supply chain for tags T_0 and T_1 outside the range of adversary $\mathcal{A}$, updates the path signature and re-encrypts the states of tags T_0 and T_1 according to TRACKER. Finally, adversary $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathsf{Flip}}$ as follows.

  1.) He first picks randomly $b \in \{0,1\}$ and returns tag T_b from the pair of tags T_0 and T_1. We assume that T_b at this point of the game stores the state $S_{\mathsf{T}_b} = (c_{\mathsf{ID}_b}, c_{H_b}, c_{\sigma_b})$.

  2.) He re-encrypts the state $S_{\mathsf{T}_b} = (c_{\mathsf{ID}_b}, c_{H_b}, c_{\sigma_b})$ using (g^y, g^z) to obtain a new state $S'_{\mathsf{T}_b} = (c'_{\mathsf{ID}_b}, c'_{H_b}, c'_{\sigma_b})$:

$$\begin{aligned} c'_{\mathsf{ID}_b} &= (u'_{\mathsf{ID}_b}, v'_{\mathsf{ID}_b}) = (g^{yr_{\mathsf{ID}}} u_{\mathsf{ID}_b}, g^{zr_{\mathsf{ID}}} v_{\mathsf{ID}_b}) \\ c'_{H_b} &= (u'_{H_b}, v'_{H_b}) = (g^{yr_H} u_{H_b}, g^{zr_H} v_{H_b}) \\ c'_{\sigma_b} &= (u'_{\sigma_b}, v'_{\sigma_b}) = (g^{yr_\sigma} u_{\sigma_b}, g^{zr_\sigma} v_{\sigma_b}) \end{aligned}$$

- Now, adversary $\mathcal{B}$ returns tag T_b to adversary $\mathcal{A}$.

Notice that if $z = xy$, then the state S'_{T_b} is a correct re-encryption of the state S_{T_b}, i.e., S'_{T_b} is a valid state that corresponds to tag T_b. Consequently, the simulation of TRACKER by adversary $\mathcal{B}$ does not differ from an actual TRACKER system, and adversary $\mathcal{A}$ can output a correct guess b' for the value of b with a non-negligible advantage ϵ.

If $z \neq xy$, then the state S'_{T_b} does not correspond to tag T_b, and adversary $\mathcal{A}$'s view of the tag unlinkability game is independent of b. Therefore, adversary $\mathcal{A}$ has only a negligible advantage in outputting a correct guess b' for the bit b.

This leads to a statistical distinguisher between the two distributions (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, g^z), $x, y, z \in \mathbb{F}_q$, breaking hereby the DDH assumption in $\mathbb{G}$.

If adversary $\mathcal{A}$ outputs $b' = b$, then adversary $\mathcal{B}$ outputs $z = xy$; otherwise adversary $\mathcal{B}$ outputs $z \neq xy$.

In conclusion, if there is an adversary $\mathcal{A}(r, s, \rho, \epsilon)$ who breaks the tag unlinkability of TRACKER, then there is an adversary $\mathcal{B}$ who breaks the DDH assumption in $\mathbb{G}$ with a non-negligible advantage $\epsilon' = \epsilon$. □

5.4.6 Evaluation

TRACKER can be implemented using today's available RFID tags. It requires tags to only store data, i.e, the encrypted ID, the encrypted hash and the encrypted path signature. Consequently, the tag stores three Elgamal ciphertexts $c_{\mathsf{ID}} = (g^{r_{\mathsf{ID}}}, \mathsf{ID}\tilde{g}^{r_{\mathsf{ID}}})$, $c_H = (g^{r_H}, H(\mathsf{ID})\tilde{g}^{r_H})$ and $c_\sigma = (g^{r_\sigma}, \sigma_{\mathcal{P}}(\mathsf{ID})\tilde{g}^{r_\sigma})$, which results in an overall storage of $2 \cdot 3 \cdot 160 = 960$ bits. Storing only 1 Kbit of data is feasible for today's EPC Class 1 Gen 2 UHF tags, for example Alien Technology's Higgs 3 tags (2).

Complexity for readers is also low in TRACKER. A reader R_k at step v_k is required to store the pair $(x_0, a_k) \in \mathbb{F}_q$ and the public key of Elgamal $\mathsf{pk} = (g, \tilde{g})$. So, the total storage per reader is approximately 80 bytes. Regarding computation, R_k is required to update the path signature of the tags passing by and to re-encrypt three ciphertexts: this sums up to a total of eight exponentiations in $\mathbb{G}$. Based on previous research (17), we conjecture this to be feasible even for lightweight embedded readers.

The manager M is required to maintain two lookup tables. The first table stores the list of valid paths in the supply chain, while the second corresponds to the manager M's database DB_M that contains the identifiers of tags that he has read. Therefore, the storage required in M is linear in the number of valid paths, and the number of tags in the supply chain $O(\nu+n)$. The path verification on the other hand, requires the manager M to **1.)** decrypt three elliptic curve ciphertexts to get ID, $H(\mathsf{ID})$ and $\sigma_{\mathcal{P}}(\mathsf{ID})$. Then, **2.)** to parse its database DB_M for clone detection. Finally, if no cloning is detected, **3.)** manager M is required to check for each valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain whether the equation $H(\mathsf{ID})^{K^i} = H(\mathsf{ID})^{\phi(\mathcal{P}_{\text{valid}_i})} = \sigma_{\mathcal{P}}(\mathsf{ID})$ holds or not, which results in performing $O(\nu)$ exponentiation in $\mathbb{G}$.

However, we note that the path verification can be optimized to reach a constant time complexity $O(1)$ by trading off computation load on the manager and the storage on tags. The main idea is to store into tags the encryption of the tuple $(\mathsf{ID}, H(\mathsf{ID}), H(\mathsf{ID})^{\phi(\mathcal{P}_{\text{valid}_i})}$, $g^{\phi(\mathcal{P}_{\text{valid}_i})})$. Now, the verification key K^i of the valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain is defined as $K^i = (\phi(\mathcal{P}_{\text{valid}_i}), g^{\phi(\mathcal{P}_{\text{valid}_i})}) \in \mathbb{F}_q \times \mathbb{G}$. When a tag arrives at manager M, the latter decrypts the tag's state and retrieves the tuple $(\mathsf{ID}, g', \sigma_{\mathcal{P}}(\mathsf{ID}), \tilde{\sigma})$. Manager M first

checks for clones using ID. Then, he verifies whether $g' = H(\mathsf{ID})$ and whether there is an entry in his set of verification keys that matches $\tilde{\sigma}$. If so, manager M verifies the path that the tag took using the path encoding that corresponds to $\tilde{\sigma}$. The manager thus verifies the paths of tags in constant time while tags are required to store an encryption of $g^{\phi(\mathcal{P}_{\text{valid}_i})}$ which counts for an additional 320 bits.

5.5 CHECKER: On-site Checking in Supply Chains

Although TRACKER allows for efficient, secure and privacy preserving product tracking in the supply chain, it suffers from two major drawbacks. **1.)** It requires a centralized, *trusted party* called *"manager"* to carry out the path verification; otherwise, the manager is able to inject fake products into the supply chain. **2.)** The verification can only be performed once the tags arrive at the manager, but not before. This limits the wide deployment of such a solution, especially in a context where partners do not trust each other and demand to be able to verify product genuineness in real-time "on-site".

Therefore, we propose in this section another solution for product tracking and hence genuineness verification called CHECKER. CHECKER addresses the problem of on-site checking by enabling each reader R_k in the supply chain to verify the validity of the path taken by the tag, instead of a global path verification performed by a trusted party that only takes place at the end of the supply chain. Using the notations of Section 5.2, this corresponds to a tracking system, where each step in the supply chain is a checkpoint, and each reader in the supply chain is a verifier.

Accordingly in CHECKER, each tag stores an identifier ID along with the path signature of ID computed using the polynomial path encoding presented in Section 5.4.1. The main idea behind CHECKER is to use a combination of polynomial path encoding and mechanisms of public key signatures to allow readers in the supply chain to verify the path that tags went through while preventing these same readers from injecting fake products. By verifying the signature in the tag, each reader thus validates the path taken that far, and by signing the ID the reader updates the path encoding. To protect tag privacy against readers in the supply chain, we encrypt tag identifiers and the corresponding path signature using an IND-CCA (see Definition 2.17) encryption, namely elliptic curve Cramer-Shoup encryption (41).

5.5.1 Overview

In CHECKER, a tag T going through a valid path $\mathcal{P}_{\text{valid}_i}$ stores a randomly encrypted state $S_T^j = (\mathsf{Enc}(\mathsf{ID}), \mathsf{Enc}(\sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID})))$, such that ID is T's identifier and $\sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID})$ is the path signature defined as $\sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID}) = H(\mathsf{ID})^{\phi(\mathcal{P}_{\text{valid}_i})}$.

At initialization, the issuer I writes into a tag T an initial encrypted state $S_T^0 = (\mathsf{Enc}_{\mathsf{pk}_1}(\mathsf{ID}), \mathsf{Enc}_{\mathsf{pk}_1}(\sigma_{\mathsf{v}_0}(\mathsf{ID})))$, where pk_1 is the public key of T's next step in the supply chain.

Without loss of generality, we assume that whenever tag T visits a reader R_k, the latter reads the encrypted state S_T^j stored into T and decrypts it using its own secret key sk_k to get the pair $(\mathsf{ID}, \sigma_{\mathcal{P}}(\mathsf{ID}))$. Reader R_k uses its set of verification keys $\mathcal{K}_V^k = \{K_k^1, K_k^2, ..., K_k^{\nu_k}\}$ to verify whether T went through a valid path leading to R_k or not. After the path verification, reader R_k computes the function f_{R_k} to update the state stored into tag T as depicted in Equation 5.2. Finally, it encrypts the new state of tag T using the public key of T's next step.

Privacy and security overview. To protect the *privacy* of tags against readers in the supply chain, tags store an IND-CCA secure encryption of their states. As CHECKER takes place in subgroups of elliptic curves that support bilinear pairings, we note that any IND-CCA secure scheme that takes place in DDH-hard groups [7] can be used to encrypt the tag state. For ease of presentation, we use Cramer-Shoup's scheme (CS for short) (41) as the underlying encryption. Also, readers in the supply chain do not share the same CS pair of keys, instead each reader R_k is equipped with a matching pair of CS public and secret keys $(\mathsf{sk}_k, \mathsf{pk}_k)$.

Similar to TRACKER, security is ensured by storing in tags a signature of their identifiers using the polynomial-based encoding of the path they took so far in the supply chain. The difference between CHECKER and TRACKER lies in the fact that CHECKER takes place in bilinear groups, which enables us to compute the verification key K^i for any valid path $\mathcal{P}_{\text{valid}_i}$ as $K^i = h^{\phi(\mathcal{P}_{\text{valid}_i})}$, instead of $K^i = \phi(\mathcal{P}_{\text{valid}_i})$. This property allows CHECKER to offer readers the possibility to verify product genuineness using relatively *short signatures* without jeopardizing the security of the entire supply chain. In fact, we show that without having access to the polynomial-based encoding of valid paths, an adversary cannot forge a valid state; otherwise he will be able to break the bilinear computational Diffie-Hellman (BCDH) assumption (cf. Definition 2.32).

Remark 5.4. *We use an IND-CCA cryptosystem to encrypt tags' states in order to ensure tag unlinkability against readers which can perform genuineness verification and therewith decrypt the encrypted states of tags.*

5.5.2 CHECKER

Before presenting the details of CHECKER, we first introduce the Cramer-Shoup cryptosystem that is used to encrypt the tags' states.

5.5.2.1 Cramer-Shoup Encryption

An elliptic curve Cramer-Shoup encryption consists of the following operations:

[7] CHECKER can take place either in bilinear groups where the XDH assumption holds or in bilinear groups where the SXDH assumption holds.

- **Setup:** The system outputs an elliptic curve $\mathcal{E}$ over a finite field $\mathbb{F}_p$. Let $\mathbb{G}_1$ be a subgroup of $\mathcal{E}$ of a large prime order q ($|q| = 160$ bits), where DDH is intractable. Let (g_1, g_2) be a pair of generators of the group $\mathbb{G}_1$.

- **Key generation:** The secret key is the random tuple $\mathsf{sk} = (x_1, x_2, y_1, y_2, z) \in \mathbb{F}_q^5$. The system computes then $(c, d, f) = (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}, g_1^z)$. Let G be a cryptographic hash function. The public key is $\mathsf{pk} = (g_1, g_2, c, d, f, G)$.

- **Encryption:** Given a message $m \in \mathbb{G}_1$, the encryption algorithm chooses $r \in \mathbb{F}_q$ at random. Then it computes $u_1 = g_1^r, u_2 = g_2^r, u = mf^r, \alpha = G(u_1, u_2, u), v = c^r d^{r\alpha}$. The encryption algorithm outputs the ciphertext $\mathsf{Enc}_{\mathsf{pk}}(m) = (u_1, u_2, u, v)$.

- **Decryption:** On input of a ciphertext $C = (u_1, u_2, u, v)$, the decryption algorithm first computes $\alpha = G(u_1, u_2, u)$, and tests if $v = u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$. If this condition does not hold, the decryption algorithm outputs $\perp$; otherwise, it outputs $\mathsf{Dec}_{\mathsf{sk}}(C) = \dfrac{u}{u_1^z}$.

5.5.2.2 Protocol Description

CHECKER consists of an initial setup phase, the initialization of tags by the issuer, and finally the path verification and tag state update by the readers.

- **Setup:** A trusted third party (TTP) outputs $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, h, H, G, e)$, where $\mathbb{G}_1$, $\mathbb{G}_T$ are subgroups of prime order q, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an asymmetric bilinear pairing, cf. Section 2.3.3, Remark 2.5. g_1 and g_2 are random generators of $\mathbb{G}_1$, while h is a generator of $\mathbb{G}_2$. $H : \{0,1\}^* \to \mathbb{G}_1$[8] and $G : \{0,1\}^* \to \mathbb{F}_q$ are secure hash functions.

  The TTP generates $\eta+1$ pairs of matching public and secret keys for the Cramer-Shoup encryption: $\mathsf{sk}_k = (x_{(1,k)}, x_{(2,k)}, y_{(1,k)}, y_{(2,k)}, z_k) \in \mathbb{F}_q^5$ and $\mathsf{pk}_k = (g_1, g_2, c_k, d_k, f_k, G)$, $0 \le k \le \eta$. The TTP generates as well $\eta + 1$ random coefficients $a_k \in \mathbb{F}_q$. Then, it selects a generator x_0 of $\mathbb{F}_q$.

  Through a secure channel, the TTP sends to each reader $R_k, 1 \le k \le \eta$, the tuple $(x_0, a_k, \mathsf{sk}_k, \mathsf{pk}_k, H)$ and sends the tuple $(x_0, a_0, \mathsf{sk}_0, \mathsf{pk}_0, H)$ to issuer I.

  The TTP computes the verification keys for each reader R_k in the supply chain. Let $\mathcal{P}_{\text{valid}_i}$ be a valid path leading to reader R_k. To obtain the verification key K_k^i corresponding to path $\mathcal{P}_{\text{valid}_i}$, the TTP computes the path encoding $\phi(\mathcal{P}_{\text{valid}_i})$ and outputs

$$K_k^i = h^{\phi(\mathcal{P}_{\text{valid}_i})} \in \mathbb{G}_2$$

  Once all the verification keys are computed, the TTP provides each reader R_k with its set $\mathcal{K}_V^k$ of verification keys.

[8] The hash function H will be viewed as a random oracle in the rest of this section.

We assume that the public keys $\mathsf{pk}_k, 0 \leq k \leq \eta$, are known to all parties in the system.

- **Tag initialization:** For each new tag T in the supply chain, I chooses a random identifier $\mathsf{ID} \in \mathbb{G}_1$. The issuer computes the hash $H(\mathsf{ID})$, and using his secret coefficient a_0, he computes $H(\mathsf{ID})^{a_0}$. Provided with the public key of T's next step, the issuer computes a CS encryption of both ID and $\sigma_{\mathsf{v}_0}(\mathsf{ID}) = H(\mathsf{ID})^{a_0}$. Without loss of generality, we assume that T's next step is v_1. The public key of step v_1 is $\mathsf{pk}_1 = (g_1, g_2, c_1, d_1, f_1, G)$.

  Issuer I draws two random number r_{ID} and r_σ in $\mathbb{F}_q$ and computes the following ciphertexts:

$$\begin{aligned}
c^0_{\mathsf{ID}} &= \mathsf{Enc}_{\mathsf{pk}_1}(\mathsf{ID}) = (u_{(1,\mathsf{ID})}, u_{(2,\mathsf{ID})}, u_{\mathsf{ID}}, v_{\mathsf{ID}}) \\
&= (g_1^{r_{\mathsf{ID}}}, g_2^{r_{\mathsf{ID}}}, \mathsf{ID}\ f_1^{r_{\mathsf{ID}}}, c_1^{r_{\mathsf{ID}}} d_1^{r_{\mathsf{ID}}\alpha_{\mathsf{ID}}}) \\
\alpha_{\mathsf{ID}} &= G(u_{(1,\mathsf{ID})}, u_{(2,\mathsf{ID})}, u_{\mathsf{ID}}) \\
c^0_\sigma &= \mathsf{Enc}_{\mathsf{pk}_1}(\sigma_{\mathsf{v}_0}(\mathsf{ID})) = (u_{(1,\sigma)}, u_{(2,\sigma)}, u_\sigma, v_\sigma) \\
&= (g_1^{r_\sigma}, g_2^{r_\sigma}, \sigma_{\mathsf{v}_0}(\mathsf{ID})\ f_1^{r_\sigma}, c_1^{r_\sigma} d_1^{r_\sigma \alpha_\sigma}) \\
\alpha_\sigma &= G(u_{(1,\sigma)}, u_{(2,\sigma)}, u_\sigma)
\end{aligned}$$

  Finally, I writes the state $S^0_T = (c^0_{\mathsf{ID}}, c^0_\sigma) \in \mathbb{G}_1^8$ into tag T. T then enters the supply chain.

- **Path verification by readers:** Assume a tag T arrives at steps v_k in the supply chain. The reader R_k associated with step v_k reads the state $S^j_T = (c^j_{\mathsf{ID}}, c^j_\sigma)$ stored in tag T. Without loss of generality, we assume that T went through path $\mathcal{P}$. R_k using its secret key sk_k decrypts the CS ciphertexts c^j_{ID} and c^j_σ and gets respectively the pair $(\mathsf{ID}, \sigma_{\mathcal{P}}(\mathsf{ID}))$.

  Let $\mathcal{K}^k_V$ denote the set of verification keys $\mathcal{K}^k_V = \{K^1_k, K^2_k, ..., K^{\nu_k}_k\} = \{h^{\phi(\mathcal{P}^1_{\text{valid}_k})}, h^{\phi(\mathcal{P}^2_{\text{valid}_k})}, ..., h^{\phi(\mathcal{P}^{\nu_k}_{\text{valid}_k})}\}$ corresponding to the valid paths leading to step v_k.

  To verify whether tag T went through a valid path or not, R_k computes the hash $H(\mathsf{ID})$ and checks whether there exists $i \in \{1, 2, ..., \nu_k\}$, such that:

$$\begin{aligned}
e(\sigma_{\mathcal{P}}(\mathsf{ID}), h) &= e(H(\mathsf{ID}), K^i_k) \\
&= e\left(H(\mathsf{ID}), h^{\phi(\mathcal{P}^i_{\text{valid}_k})}\right)
\end{aligned}$$

  If so, then this implies that T went through a valid path leading to step v_k. Otherwise, the reader concludes that tag T is illegitimate and rejects it.

- **Tag state update by readers:** If the verification succeeds, then reader R_k in the supply chain is required to update the state of tag T. Using the update function f_{R_k}, the reader computes the new path signature $\sigma_{\overrightarrow{\mathcal{P}\mathsf{v}_k}}(\mathsf{ID})$ using Equation 5.2

Without loss of generality, we assume that the tag's next step is $\mathsf{v_{k+1}}$. The reader R_k prepares tag T for reader R_{k+1} by encrypting the pair $(\mathsf{ID}, \sigma_{\overrightarrow{\mathcal{P}\mathsf{v}_k}}(\mathsf{ID}))$ using the public key $\mathsf{pk}_{k+1} = (g_1, g_2, c_{k+1}, d_{k+1}, f_{k+1}, G)$ of reader R_{k+1}. Reader R_k obtains therefore, two ciphertexts c_{ID}^{j+1} and c_{σ}^{j+1}.

Finally, R_k writes the state $S_T^{j+1} = (c_{\mathsf{ID}}^{j+1}, c_{\sigma}^{j+1})$ into T.

5.5.3 Security Analysis

Theorem 5.4. CHECKER *is complete.*

Proof. Similar to the proof of Theorem 5.1 □

Theorem 5.5. CHECKER *is sound under the BCDH assumption in the random oracle model.*

Proof. Assume there is an adversary $\mathcal{A}$ who breaks the security of CHECKER with a non-negligible advantage ϵ, we build an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine to break the BCDH assumption with a non-negligible advantage ϵ'.

Let $\mathcal{O}_{\mathsf{BCDH}}$ be an oracle that selects randomly $x, y, z \in \mathbb{F}_q$, and returns $g, g^x, g^y, g^z \in \mathbb{G}_1$, and $h, h^x, h^y \in \mathbb{G}_2$.

Proof overview. If $\mathcal{A}$ has a non-negligible advantage in breaking the security of CHECKER, then $\mathcal{A}$ will be able to output a challenge tag T_c that stores a valid encrypted state S_{T_c} that fulfills the following:

i.) $\exists\ R_i$ such that $\mathsf{Check}(S_{\mathsf{T}_c}, R_i) = 1$, i.e., there is a path $\mathcal{P}_{\text{valid}_i}$ that corresponds to T_c's state;

ii.) $\exists\ \mathsf{v}_k \in \mathcal{P}_{\text{valid}_i}$ such that the step v_k is not corrupted by $\mathcal{A}$;

iii.) T_c did not go through step v_k.

To break BCDH, adversary $\mathcal{B}$ simulates a CHECKER system for $\mathcal{A}$ where he provides a step v_k in the supply chain with the tuple $(x_0, g^x, \mathsf{sk}_k, \mathsf{pk}_k)$ instead of the tuple $(x_0, a_k, \mathsf{sk}_k, \mathsf{pk}_k)$.

Without loss of generality, we assume in the rest of the proof that $\mathsf{v}_k = \mathsf{v}_0$ and that adversary $\mathcal{A}$ corrupts all readers in the supply chain.

Now, adversary $\mathcal{B}$ must convince adversary $\mathcal{A}$ that v_0 is associated with the secret coefficient $a_0 = x$ that corresponds to the pair (g^x, h^x) received from the oracle $\mathcal{O}_{\mathsf{BCDH}}$. Accordingly, $\mathcal{B}$ has to be able to compute $H(\mathsf{ID})^x$ only by knowing (g^x, h^x). To this effect, adversary $\mathcal{B}$ simulates a random oracle $\mathcal{H}$ that computes the hash function H.

When $\mathcal{H}$ is queried in the learning phase with identifier ID_j, $\mathcal{B}$ picks a random number r_j and computes $H(\mathsf{ID}_j) = g^{r_j}$.

Before the challenge phase, adversary $\mathcal{A}$ queries the random oracle $\mathcal{H}$ with an identifier ID_c, where ID_c is the identifier of the challenge tag T_c. Simulating $\mathcal{H}$, adversary $\mathcal{B}$ picks a random number r_c, computes $H(\mathsf{ID}_c) = g^{zr_c}$, and returns $H(\mathsf{ID}_c)$ to adversary $\mathcal{A}$.

At the end of the challenge phase, adversary $\mathcal{A}$ supplies adversary $\mathcal{B}$ with the challenge tag T_c.

Since adversary $\mathcal{A}$ has a non-negligible advantage in winning the soundness game, it follows that the challenge tag T_c stores an encrypted valid state that corresponds to some valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain. That is, tag T_c stores the encrypted pair $(\mathsf{ID}_c, \sigma_c = \sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID}))$ while T_c did not go through step v_0.

Using σ_c and CHECKER's verification keys, adversary $\mathcal{B}$ can identify the path $\mathcal{P}_{\text{valid}_i}$ that corresponds to the state of tag T_c. We assume that $\mathcal{P}_{\text{valid}_i} = \overrightarrow{\mathsf{v}_0 \mathcal{P}'_{\text{valid}_i}}$, and we denote l the length of path $\mathcal{P}_{\text{valid}_i}$.

By definition, $\phi(\mathcal{P}_{\text{valid}_i}) = a_0 x_0^l + \phi(\mathcal{P}'_{\text{valid}_i}) = x x_0^l + \phi(\mathcal{P}'_{\text{valid}_i})$. Given σ_c and the encoding $\phi(\mathcal{P}'_{\text{valid}_i})$ of the sub-path $\mathcal{P}'_{\text{valid}_i}$, $\mathcal{B}$ computes:

$$\begin{aligned} \frac{\sigma_c}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} &= \frac{H(\mathsf{ID}_c)^{\phi(\mathcal{P}_{\text{valid}_i})}}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} = H(\mathsf{ID}_c)^{x x_0^l} \\ H(\mathsf{ID}_c)^x &= \left(\frac{\sigma_c}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} \right)^{\frac{1}{x_0^l}} \end{aligned}$$

Now adversary $\mathcal{B}$ has access to $H(\mathsf{ID}_c)^x = (g^{z r_c})^x = g^{x z r_c}$, which he can use to compute $(g^{x z r_c})^{\frac{1}{r_c}} = g^{xz}$, and finally $e(g^{xz}, h^y) = e(g,h)^{xyz}$, breaking thus the BCDH assumption.

Simulation of the random oracle $\mathcal{H}$. To respond to the queries of the random oracle $\mathcal{H}$, adversary $\mathcal{B}$ keeps a table T_H of tuples $(\mathsf{ID}_j, r_j, \mathsf{coin}(\mathsf{ID}_j), H(\mathsf{ID}_j))$ as explained below.

On a query $H(\mathsf{ID}_i)$, $\mathcal{B}$ replies as follows:

1.) If there is a tuple $(\mathsf{ID}_i, r_i, \mathsf{coin}(\mathsf{ID}_i), H(\mathsf{ID}_i))$ that corresponds to ID_i, then $\mathcal{B}$ returns $H(\mathsf{ID}_i)$.

2.) If ID_i has never been queried before, then adversary $\mathcal{B}$ picks a random number $r_i \in \mathbb{F}_q$, and flips a random coin $\mathsf{coin}(\mathsf{ID}_i) \in \{0,1\}$ such that: $\mathsf{coin}(\mathsf{ID}_i) = 1$ with probability p, and it is equal to 0 with probability $1-p$. If $\mathsf{coin}(\mathsf{ID}_i) = 0$, then $\mathcal{B}$ answers with $H(\mathsf{ID}_i) = g^{r_i}$. Otherwise, he answers with $H(\mathsf{ID}_i) = (g^z)^{r_i}$. Finally, adversary $\mathcal{B}$ stores the tuple $(\mathsf{ID}_i, r_i, \mathsf{coin}(\mathsf{ID}_i), H(\mathsf{ID}_i))$ in table T_H.

Construction. First, adversary $\mathcal{B}$ queries $\mathcal{O}_{\mathsf{BCDH}}$ to receive $g, g^x, g^y, g^z \in \mathbb{G}_1$ and $h, h^x, h^y \in \mathbb{G}_2$. Then, $\mathcal{B}$ simulates the challenger $\mathcal{C}$ and creates a complete CHECKER system.

- Adversary $\mathcal{B}$ generates $\eta + 1$ pairs of matching CS public and secret keys $(\mathsf{sk}_k, \mathsf{pk}_k)$. Then, he generates η random coefficients a_k.
- He provides each reader R_k in CHECKER with the tuple $(x_0, a_k, \mathsf{sk}_k, \mathsf{pk}_k)$.
- He provides the issuer I with the tuple $(x_0, g^x, \mathsf{sk}_0, \mathsf{pk}_0)$, as if $a_0 = x$.

- He computes the verification keys for each reader R_k in the supply chain. Without loss of generality, a valid path $\mathcal{P}_{\text{valid}_i}$ in the supply chain could be represented as $\mathcal{P}_{\text{valid}_i} = \overrightarrow{\mathsf{v}_0\mathcal{P}'_{\text{valid}_i}}$. Thus, the corresponding verification key K^i is computed as: $K^i = (h^x)^{x_0^l} h^{\phi(\mathcal{P}'_{\text{valid}_i})} = h^{\phi(\mathcal{P}_{\text{valid}_i})}$, where l is the length of path $\mathcal{P}_{\text{valid}_i}$.

  Once the verification keys are computed for all the readers R_k, $\mathcal{A}$ provides each reader R_k with its set $\mathcal{K}_V^k$ of verification keys.

- Adversary $\mathcal{B}$ simulates the issuer I and creates n tags T_j for CHECKER.

  He selects randomly $\mathsf{ID}_j \in \mathbb{G}_1$, simulates the random oracle $\mathcal{H}$ and gets the tuple $(\mathsf{ID}_j, r_j, \mathsf{coin}(\mathsf{ID}_j), H(\mathsf{ID}_j))$.

  If $\mathsf{coin}(\mathsf{ID}_j) = 1$, i.e., $H(\mathsf{ID}_j) = g^{zr_j}$, then $\mathcal{B}$ cannot compute $H(\mathsf{ID}_j)^x = g^{xzr_j}$ as he does not know both x and z. Consequently, adversary $\mathcal{B}$ stops the soundness game. Otherwise using r_j, adversary $\mathcal{B}$ computes $H(\mathsf{ID}_j)^x = (g^x)^{r_j}$.

  Finally, adversary $\mathcal{B}$ encrypts both ID_j and $\sigma_{\mathsf{v}_0}(\mathsf{ID}_j)$ using the public key of T_j's next step. $\mathcal{B}$ stores the resulting ciphertexts $(c^0_{\mathsf{ID}_j}, c^0_{\sigma_j})$ into tag T_j.

Learning phase. Adversary $\mathcal{B}$ calls adversary $\mathcal{A}$ and simulates the learning phase of the soundness game.

- Adversary $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathsf{CorruptR}}$ for $\mathcal{A}$. For ease of understanding, we assume that $\mathcal{A}$ corrupts all readers R_k in the supply chain.

- Adversary $\mathcal{B}$ simulates readers R_k along the supply chain. Let T_j be a tag which went through path $\mathcal{P}$ and arrives at step v_k.

  Adversary $\mathcal{B}$ decrypts the state of tag T_j using CS secret key sk_k of reader R_k and gets the pair $(\mathsf{ID}_j, \sigma_{\mathcal{P}}(\mathsf{ID}_j))$. He verifies the path of tag T_j using $\mathcal{K}_V^k$. Then $\mathcal{B}$ updates the path of tag T_j using the secret coefficient a_k.

  Finally, using the public key of T_j's next step, $\mathcal{B}$ encrypts T_j's identifier and T_j's path signature.

Challenge phase. Adversary $\mathcal{A}$ outputs a tag T_c.

Since adversary $\mathcal{A}$ has a non-negligible advantage in the soundness game, it follows that **i.)** $\exists$ R_i such that $\mathsf{Check}(R_i, \mathsf{T}_c) = 1$, and **ii.)** T_c did not go through step v_0.
We assume without loss of generality that T_c's state corresponds to the pair $(\mathsf{ID}_c, \sigma_c)$.

- $\mathcal{B}$ first checks whether $\mathsf{coin}(\mathsf{ID}_c) = 1$ or not.

  If $\mathsf{coin}(\mathsf{ID}_c) = 0$, then adversary $\mathcal{B}$ stops the game. Notice that if $H(\mathsf{ID}_c) = g^{r_c}$, adversary $\mathcal{B}$ will not be able to break the BCDH assumption.

  If $\mathsf{coin}(\mathsf{ID}_c) = 1$, i.e., $H(\mathsf{ID}_c) = g^{zr_c}$, then adversary $\mathcal{B}$ continues the game, and computes $e(g,h)^{xyz}$.

- Using the verification keys, adversary $\mathcal{B}$ identifies the path $\mathcal{P}_{\text{valid}_i} = \overrightarrow{\mathsf{v}_0\mathcal{P}'_{\text{valid}_i}}$ that matches T_c's path signature σ_c.

Let l denote the length of path $\mathcal{P}_{\text{valid}_i}$. We have:

$$\begin{aligned}
\phi(\mathcal{P}_{\text{valid}_i}) &= a_0 x_0^l + \phi(\mathcal{P}'_{\text{valid}_i}) = x x_0^l + \phi(\mathcal{P}'_{\text{valid}_i}) \\
H(\mathsf{ID}_c)^{x x_0^l} &= \frac{\sigma_c}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} = \frac{H(\mathsf{ID}_c)^{\phi(\mathcal{P}_{\text{valid}_i})}}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}} \\
H(\mathsf{ID}_c)^{x} &= \left(\frac{\sigma_c}{H(\mathsf{ID}_c)^{\phi(\mathcal{P}'_{\text{valid}_i})}}\right)^{\frac{1}{x_0^l}} = (g^{z r_c})^x = g^{x z r_c}
\end{aligned}$$

Provided with the random number r_c, $\mathcal{B}$ finally computes:

$$e\left(H(\mathsf{ID}_c)^x, h^y\right)^{\frac{1}{r_c}} = (e(g,h)^{xyz r_c})^{\frac{1}{r_c}} = e(g,h)^{xyz}$$

Here we compute the advantage ϵ' of adversary $\mathcal{B}$. Notice that adversary $\mathcal{B}$ succeeds in breaking the BCDH assumption if he does not stop the soundness game.

1.) $\mathcal{B}$ halts the game, if during the initialization of the n tags in CHECKER, there is a tag T_j such that $\mathsf{coin}(\mathsf{ID}_j) = 1$. This event occurs with probability p. Hence, the probability that $\mathcal{B}$ does not stop the game during the learning phase is: $(1-p)^n$.

2.) $\mathcal{B}$ stops the game during the challenge phase, if $\mathsf{coin}(\mathsf{ID}_c) = 0$. As a result, $\mathcal{B}$ does not stop the game in the challenge phase with probability p.

Let E denote the event: adversary $\mathcal{B}$ does not stop the soundness game.

Let E_1 denote the event: adversary $\mathcal{B}$ does not stop the soundness game in the learning phase, $Pr(E_1) = (1-p)^n$.

Let E_2 denote the event: adversary $\mathcal{B}$ does not stop the soundness game in the challenge phase, $Pr(E_2) = p$. Hence,

$$\begin{aligned}
\pi &= Pr(E) = Pr(E_1) Pr(E_2) \\
&= p(1-p)^n
\end{aligned}$$

Now, if adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the security of CHECKER, then $\mathcal{B}$ can break the BCDH assumption with a non-negligible advantage $\epsilon' = \pi\epsilon$, leading to a contradiction.

Note that π is maximal when $p = \frac{1}{n}$ and $\pi_{\mathsf{max}} = \frac{\left(1-\frac{1}{n}\right)^n}{n} \simeq \frac{1}{en}$. □

5.5.4 Privacy Analysis

Theorem 5.6. CHECKER *ensures tag unlinkability under the XDH assumption.*

Proof. To prove tag unlinkability, we use the IND-CCA property of Cramer-Shoup encryption ensured under the XDH assumption, see Definition 2.35.

Assume there is an adversary $\mathcal{A}$ who breaks the tag unlinkability of CHECKER with a non-negligible advantage ϵ, we show that there is an adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a subroutine and breaks the IND-CCA property of Cramer-Shoup encryption with a non-negligible advantage ϵ'.

Let $\mathcal{O}_{\mathsf{Dec}}$ be the oracle that, on input of a ciphertext c encrypted with public key pk, outputs the underlying plaintext m.

Let $\mathcal{O}_{\mathsf{Enc}}$ be the oracle that, provided with two messages m_0 and m_1 and public key pk, randomly chooses $b \in \{0,1\}$, encrypts m_b using public key pk, and returns the challenge ciphertext c_b.

Proof overview. The idea of the proof is to build a CHECKER system such that there is a step v_k in the supply chain that is associated with public key pk, where pk is the challenge public key from the IND-CCA security game.

In the learning phase, adversary $\mathcal{B}$ is required to simulate reader R_k. This implies that $\mathcal{B}$ has to decrypt the state of tags arriving at step v_k. Hence the need to a decryption oracle and therewith to an IND-CCA secure encryption. Now, whenever a tag T arrives at step v_k, $\mathcal{B}$ first calls the decryption oracle for the Cramer-Shoup encryption $\mathcal{O}_{\mathsf{Dec}}$ that returns the underlying plaintexts, i.e., ID and $\sigma_{\mathcal{P}}(\mathsf{ID})$. Then, $\mathcal{B}$ verifies the validity of the pair and updates the state of tag T.

In the challenge phase, adversary $\mathcal{A}$ returns the challenge tags T_0 and T_1 to adversary $\mathcal{B}$. Adversary $\mathcal{B}$ decrypts the state of tags T_0 and T_1 and gets their identifiers ID_0 and ID_1 respectively. Then, adversary $\mathcal{B}$ queries the encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ with messages ID_0 and ID_1. The encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ returns the challenge ciphertext $c_b = \mathsf{Enc}_{\mathsf{pk}}(\mathsf{ID}_b), b \in \{0,1\}$. Adversary $\mathcal{B}$ iterates the supply chain outside the range of $\mathcal{A}$, and simulates the oracle $\mathcal{O}_{\mathsf{Flip}}$ by returning T_b which stores the ciphertext c_b along with an encryption of T_b's path signature. As $\mathcal{B}$ makes a guess to choose the path signature that corresponds to tag T_b, it follows that the path signature stored into T_b will be correct with probability $\frac{1}{2}$.

If adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the tag unlinkability game, then he outputs a correct guess for the value of b. If adversary $\mathcal{A}$ outputs $b = 0$, then this implies that T_b stores an encryption of ID_0 and thus $c_b = \mathsf{Enc}_{\mathsf{pk}}(\mathsf{ID}_0)$; otherwise, $c_b = \mathsf{Enc}_{\mathsf{pk}}(\mathsf{ID}_1)$.

Construction. To break the IND-CCA property of Cramer and Shoup encryption, $\mathcal{B}$ proceeds as follows:

Adversary $\mathcal{B}$ creates a supply chain for the CHECKER protocol and simulates the challenger $\mathcal{C}$ of the tag unlinkability game.

Learning phase.

- Adversary $\mathcal{B}$ calls adversary $\mathcal{A}$ who queries the oracle $\mathcal{O}_{\mathsf{CorruptR}}$ with the identity of r readers R_i. Adversary $\mathcal{B}$ simulates the oracle $\mathcal{O}_{\mathsf{CorruptR}}$ and assigns to each reader R_i a tuple $(x_0, a_i, \mathsf{sk}_i, \mathsf{pk}_i)$ that he returns to adversary $\mathcal{A}$.

- Now, $\mathcal{B}$ selects a reader R_k from the set of uncorrupted readers and assigns to reader R_k the tuple $(x_0, a_k, \mathsf{pk}_k = \mathsf{pk})$. Without loss of generality, we assume that step v_k in the supply chain is associated with reader R_k.

- Simulating the oracle $\mathcal{O}_{\mathsf{Tag}}$, adversary $\mathcal{B}$ supplies $\mathcal{A}$ with two challenge tags T_0 and T_1 that have just been issued by issuer I (i.e., just entered the supply chain).

- Adversary $\mathcal{A}$ iterates the supply chain ρ times. Before each iteration j of the supply chain:

  1.) Adversary $\mathcal{A}$ reads and writes into tags T_0 and T_1.

  2.) Simulating the oracle $\mathcal{O}_{\mathsf{Step}}$, adversary $\mathcal{B}$ provides adversary $\mathcal{A}$ with the next step of tags T_0 and T_1.

  3.) $\mathcal{B}$ simulates the oracles $\mathcal{O}_{\mathsf{Tag}}$ and $\mathcal{O}_{\mathsf{Step}}$ and supplies $\mathcal{A}$ with s tags $T_{(i,j)}$ together with their next step $\mathsf{v}_{T_{(i,j)}}$ in the supply chain. Then $\mathcal{A}$ iterates the supply chain and reads the states stored into tags $T_{(i,j)}$.

- When a tag T in the learning phase arrives at step v_k, then adversary $\mathcal{B}$ simulates reader R_k:

  1.) Adversary $\mathcal{B}$ reads the state stored into tag T and gets two CS ciphertexts c_{ID} and c_σ.

  2.) He queries the decryption oracle $\mathcal{O}_{\mathsf{Dec}}$ with the ciphertexts c_{ID} and c_σ. The oracle $\mathcal{O}_{\mathsf{Dec}}$ returns the corresponding plaintexts ID and σ.

  3.) He checks then if the pair (ID, σ) corresponds to a valid path leading to step v_k.

  4.) Finally, he updates the path signature of T and encrypts both the identifier ID and the path signature using the public key of T's next step.

Challenge phase. Adversary $\mathcal{B}$ simulates the oracles $\mathcal{O}_{\mathsf{Step}}$ and provides adversary $\mathcal{A}$ with the next steps of tags T_0 and T_1. Then, he iterates the supply chain for tags T_0 and T_1 outside the range of adversary $\mathcal{A}$.

- Adversary $\mathcal{B}$ decrypts the states stored into T_0 and T_1, and gets ID_0 and ID_1 respectively.

- $\mathcal{B}$ queries the oracle $\mathcal{O}_{\mathsf{Enc}}$ with messages ID_0 and ID_1. The encryption oracle $\mathcal{O}_{\mathsf{Enc}}$ returns $c_{\mathsf{ID}_b} = \mathsf{Enc}_{\mathsf{pk}}(\mathsf{ID}_b)$.

- $\mathcal{B}$ prepares the challenge tag T_b for adversary $\mathcal{A}$:

  1.) Adversary $\mathcal{B}$ updates the path of tags T_0 and T_1 and encrypts the path signature using the public key pk. He obtains two ciphertexts c_{σ_0} and c_{σ_1}.

  2.) He randomly selects $b' \in \{0,1\}$ and stores the state $S_{\mathsf{T}_b} = (c_{\mathsf{ID}_b}, c_{\sigma_{b'}})$ in T_b. Therefore, T_b's next step is step v_k associated with public key pk.

- Simulating the oracle $\mathcal{O}_{\mathsf{Flip}}$, adversary $\mathcal{B}$ provides adversary $\mathcal{A}$ with the challenge tag T_b.

Notice that if $b = b'$, then the state $S_{\mathsf{T}_b} = (c_{\mathsf{ID}_b}, c_{\sigma_{b'}})$ computed by $\mathcal{B}$ when simulating CHECKER corresponds to a well formed pair $(\mathsf{ID}_b, \sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID}_b))$, and consequently, the simulation of CHECKER by $\mathcal{B}$ does not differ from an actual CHECKER system. $\mathcal{A}$ can accordingly output a correct guess for the tag corresponding to the challenge tag T_b with a non-negligible advantage ϵ.

If adversary $\mathcal{A}$ outputs $b = 0$, this means that T_b stores an encryption of ID_0, and adversary $\mathcal{B}$ outputs 0. If $\mathcal{A}$ outputs $b = 1$, then this means that T_b stores an encryption of ID_1, and $\mathcal{B}$ outputs 1.

If $b \neq b'$, then the probability that $\mathcal{B}$ breaks the IND-CCA property of CS is at worst a random guess, i.e., $\frac{1}{2}$.

Now, we quantify the advantage of adversary $\mathcal{B}(r_e, 0, r_d, 0, \epsilon')$ in breaking the IND-CCA property of CS. We note that $r_e \leq s\rho + 2\rho + 2$ and $r_d \leq s\rho + 2\rho + 2$.

– Let E_1 be the event that $\mathcal{B}$ breaks the IND-CCA property of CS.

– Let E_2 be the event that $b = b'$.

Since b' is selected randomly, the probability that $b = b'$ is $\frac{1}{2}$. Hence,

$$\begin{aligned} Pr(E_1) &= Pr(E_1|E_2) \cdot Pr(E_2) + Pr(E_1|\overline{E_2}) \cdot Pr(\overline{E_2}) \\ &= \frac{1}{2} Pr(E_1|E_2) + \frac{1}{2} Pr(E_1|\overline{E_2}) \\ &= \frac{1}{2}\left(\frac{1}{2} + \epsilon\right) + \frac{1}{2} Pr(E_1|\overline{E_2}) \\ &\geq \frac{1}{2}\left(\frac{1}{2} + \epsilon + \frac{1}{2}\right) = \frac{1}{2} + \frac{\epsilon}{2} \end{aligned}$$

Thus, the advantage ϵ' of adversary $\mathcal{B}$ in breaking the IND-CCA property of CS is at least $\frac{\epsilon}{2}$.

We conclude that if $\mathcal{A}$ has a non-negligible advantage ϵ to break CHECKER, then $\mathcal{B}(r_e, 0, r_d, 0, \epsilon')$ will have a non-negligible advantage ϵ' to break the IND-CCA property of Cramer and Shoup encryption, which leads to a contradiction under the XDH assumption. □

5.5.5 Evaluation

A tag in CHECKER is required to store a pair of IND-CCA encryptions of its identifier ID and its path signature $\sigma_{\mathcal{P}_{\text{valid}_i}}(\mathsf{ID}) = H(\mathsf{ID})^{\phi(\mathcal{P}_{\text{valid}_i})}$. Since we use Cramer-Shoup's scheme as the underlying encryption, tags are required to store $2 \cdot 4 \cdot 160 = 1280$ bits. We emphasize that any IND-CCA1 secure encryption in DDH-hard subgroups of elliptic curve is sufficient to implement CHECKER. One possible choice of encryption scheme is CS-lite (41), a light variant of CS encryption which is IND-CCA1 secure and costs 480 bits per encryption instead of 640 bits. Also, there is a variant of Elgamal proposed by Fujisaki and Okamoto (62) which is IND-CCA2 secure in the random oracle model, and whose storage requirements are comparable to Elgamal's. We believe that CHECKER can be implemented in current ISO 18000-3 HF tags, such as UPM RFID MiniTrack tags (155) that feature 1 Kbit of memory.

Moreover, a reader R_k in the supply chain is required to decrypt the state stored into tags using its secret key sk_k, then to verify the validity of the paths that tags went through, and finally, to update and encrypt the states of tags. This amounts to performing: **1.)** two decryptions in $\mathbb{G}_1$ where $|\mathbb{G}_1| = 160$ bits, **2.)** the computation of ν_k bilinear pairings in $\mathbb{G}_T$, where ν_k is the number of verification keys of reader R_k and $|\mathbb{G}_T| = 1024$ bits, **3.)** two exponentiations in $\mathbb{G}_1$ to update the path signature, and finally **4.)** two encryptions in $\mathbb{G}_1$. The costly operation at reader R_k is the verification of the path signature which is linear in the number of valid paths leading to reader R_k. As in TRACKER, we can further decrease the computation load at the readers by allowing tags to store a pointer to the verification key that corresponds to the path that they took in the supply chain.

The idea is that instead of storing the encrypted pair $(\mathsf{ID}, H(\mathsf{ID})^{\phi(\mathcal{P})})$, a tag in the supply chain stores the encrypted tuple $(\mathsf{ID}, H(\mathsf{ID})^{\phi(\mathcal{P})}, g^{\phi(\mathcal{P})})$. Now the verification key K^i of the valid path $\mathcal{P}_{\text{valid}_i}$ is defined as $K^i = (g^{\phi(\mathcal{P}_{\text{valid}_i})}, h^{\phi(\mathcal{P}_{\text{valid}_i})}) \in \mathbb{G}_1 \times \mathbb{G}_2$. When tag T arrives at step v_k, reader R_k decrypts the tag's state and gets a tuple $(\mathsf{ID}, \sigma_{\mathcal{P}}(\mathsf{ID}), \tilde{\sigma})$. First, R_k checks whether $\tilde{\sigma}$ corresponds to a pair in its set of verification keys $\mathcal{K}_V^k$ or not. If so, R_k verifies the path signature $\sigma_{\mathcal{P}}(\mathsf{ID})$. Consequently, the cost of the verification of the path signature at the readers is constant. We note that a reader in the supply chain is required to perform an additional table lookup, one decryption, two exponentiations and one encryption in $\mathbb{G}_1$, and to store an additional 160 bits for each valid path in the supply chain that lead to it. Tags on the other hand have to store three encryptions of size 640 bits each in the case of Cramer-Shoup, and of size 480 bits in the case of CS-lite.

5.6 Related Work

Ouafi and Vaudenay (127) address counterfeiting of products using cryptographic hash functions on RFID tags. To protect against malicious state updates, tags authenticate readers at every step in the supply chain. Only if readers are successfully authenticated, tags will update their internal states. Ouafi and Vaudenay (127) require tags to evaluate a cryptographic

hash function twice: for reader authentication and for the state update. A similar approach with tags evaluating cryptographic hash functions is proposed by Li and Ding (110). While such setups using cryptography-enabled tags might lead to a secure and privacy-preserving solution of the counterfeiting problem, tags will always be more expensive than storage only tags.

Chawla et al. (39) check for covert channels that leak information about a supply chain's internal details. Therefore, tags are frequently synchronized with a backend-database. If a tag's state contains "extra" data that is not in the database, the tag is rejected. Also, Shuihua and Chu (148) detect malicious tampering of a tag's state in a supply chain using watermarks. Both of these schemes nonetheless do not protect tag privacy in the supply chain.

Burbridge and Soppera (31) suggest the use of proxy re-signature to allow path segment verification while using storage only tags. The tag stores a signature of the last trusted party it has visited. To prevent product injection in the supply chain, partners in the supply chain do not have secret keys to sign tags' identifiers, but rather secret proxy keys that only allow partners to transform a valid signature of one partner to their own signature. This scheme however does not address the problem of implementing practical proxy re-signatures without trusted third party. Further, it does not protect the privacy of tags in the supply chain; a tag always sends its identifier in clear when replying to readers' queries.

Other solutions exist that rely on physical properties of a "tag". For example, TAGSYS produces holographic "tags" that are expensive to clone (151). Verayo produces tags with Physically Unclonable Functions (PUF) (160). While these approaches solve product genuineness verification, they do not support the protection of tag privacy.

Our construction based on polynomial path encoding might resemble other (cryptographic) constructions based on, e.g., Rabin fingerprints (134), aggregated messages authentication codes (96) or aggregated signatures (24). However, we stress that our design focuses on **1.)** preserving both the order or sequence of steps in the supply chain and the privacy of tags, **2.)** at the same time putting only minimal computational burden on the verifiers ($O(1)$ complexity with low overhead), and **3.)** being provable. While alternative constructions might be envisioned, this is far from being straightforward.

5.7 Summary

In this chapter, we presented two protocols that are Tracker and Checker to address security and privacy challenges of product tracking in RFID-enabled supply chains. The main idea of these protocols is to verify the genuineness of products by verifying the paths that they took in the supply chain. Accordingly, paths in the supply chain are encoded using polynomials, then the resulting path encoding is used to sign tags' identifiers. Readers representing steps in the supply chain update the path encoding successively by signing tags' identifiers, while verifiers check the genuineness of products by verifying the signature stored

in tags. The security of both protocols relies on standard assumptions, namely CDH and BCDH, whereas the privacy of tags relies on the DDH assumption. Contrary to related work, our protocols do not require any computational complexity on tags and they can be implemented in current storage only tags.

6

RFID-based Item Matching in Supply Chains

6.1 Introduction

One prominent application of RFID technology is the automation of safety inspections when transporting hazardous goods such as highly reactive chemicals in supply chains. Here, it is dangerous to place specific reactive chemicals close to each other, because small leaks can already result in a threat to the life of workers managing these chemicals.

Some recent solutions to enforce safety regulations when storing or transporting chemicals in supply chains rely on equipping each chemical container with an RFID tag that stores information that identifies the chemical in the container as highlighted by EU project CoBIs (40). Before two tags are placed next to each other, their tags are wirelessly "scanned" using an RFID reader. Each tag sends its content in cleartext to a server. The server performs chemicals' matching based on a set Ref of *matching references* that it knows beforehand. Each matching reference identifies a pair of chemicals that react. Now, when two reactive chemicals are detected, the server triggers an alarm.

However, the above solution suffers from several shortcomings that may lead to security and privacy threats. The fact that tags transmit their contents in cleartext allows any malicious entity with proper wireless equipment to learn the content of a container, to infer information about reactive chemicals, and finally to track their location.

Consequently, RFID-based protocols for tag matching require a careful design that takes into account both the security and the privacy threats to RFID tags and the consequences thereof on the security and safety of users managing matched items.

A privacy preserving RFID-based tag matching must assure that tag matching is performed without disclosing the content of tags. That is, the only information revealed after executing the protocol to *readers in the supply chain* is a bit b indicating whether the tags involved in the protocol execution "match" or not. It must also ensure *location privacy* so as

to prevent tracking attacks by eavesdroppers. Ideally, an eavesdropper must not be able to distinguish between tags based on the traces of the matching protocol, in accordance with previous chapters this requirement will be called hereafter tag unlinkability.

With respect to security, it is mandatory to ensure that a matching protocol is correct (almost) all the time. Namely, it is required to detect all incompatible items (reactive chemicals). This corresponds to the *completeness* property: the protocol must always trigger an alarm when two reactive chemicals are put next to each other. Moreover, the protocol has to be efficient: an alarm is triggered only when necessary. When a match is detected by the protocol, one can safely derive that the tags involved in the protocol are attached to reactive chemicals. This second requirement corresponds to the *soundness* property of the protocol.

Note that solutions to answer the above security and privacy problems are strongly constrained by the limitations of RFID environment. While tag privacy against eavesdroppers can be achieved by using re-encryption techniques, tag privacy against readers is more difficult to address especially when using cheap RFID storage only tags unable to perform any computation. Traditional security and privacy solutions based on heavyweight secret matching protocols between two parties , cf. Ateniese et al. (4), Balfanz et al. (9), cannot be implemented in an RFID setting.

Accordingly, we design T-MATCH, a new tag matching protocol that involves tags T_i attached to "containers" (barrels) of chemicals traveling in a supply chain, multiple readers R_k and a backend server S. T-MATCH targets storage only tags only featuring storage and no computational capabilities so as to allow for the deployment of such an application with a reasonable cost.

Overview: In T-MATCH, a reader R_k in the supply chain reads out the content of a pair of tags T_i and T_j, cooperates with backend server S to perform tag matching, and finally outputs the outcome of the matching while assuring various privacy properties in the face of curious readers R_k and curious backend server S.

Reader R_k and backend server S are required to evaluate securely a *boolean* function Check for any pair of tags T_i and T_j, such that Check outputs $b = 1$, if T_i and T_j match. To this effect, each tag T_i in T-MATCH stores a homomorphic IND-CPA encryption Enc of its attribute a_{T_i}. When two tags T_i and T_j are in the range of reader R_k, reader R_k reads both tags and retrieves the encryptions $\mathsf{Enc}(a_{T_i})$ and $\mathsf{Enc}(a_{T_j})$ of T_i and T_j's attributes respectively. To protect the privacy of tags, reader R_k re-encrypts the ciphertexts stored into tags T_i and T_j. Now to evaluate the Check function, reader R_k uses the homomorphic property of Enc to compute an encryption $\mathsf{Enc}(f(a_{T_i}, a_{T_j}))$ of a function f of T_i and T_j's attributes. Then, reader R_k and backend server S engage in a two party protocol for a *modified* privacy preserving plaintext equality test (84) to check whether $f(a_{T_i}, a_{T_j}) \in \mathsf{Ref}$, where Ref is the set of matching references of backend server S. If so, Check outputs $b = 1$; otherwise, Check outputs $b = 0$.

To summarize, T-MATCH's major contributions are:

- T-Match proposes a novel solution for item matching that targets storage only tags. A tag T_i in T-Match does not perform any computation, it is only required to store a state that is updated at every protocol execution by readers R_k.

- T-Match is provably privacy preserving: T-Match relies on techniques of secure two-party computation to ensure that neither readers R_k nor backend server S can disclose the content of a tag or learn its attribute.

- T-Match is provably secure: readers R_k raise an alarm only when they interact with a pair of matching tags.

This chapter is organized as follows: we first introduce the problem statement and T-Match's setup in Section 6.2. In Section 6.3, we formalize our privacy and security requirements by presenting an adversary model that is suited for RFID-based item matching applications. Then, we present T-Match in Section 6.4, followed by a security and privacy analysis in Section 6.5 and Section 6.6 respectively. In Section 6.7, we provide a quick evaluation of T-Match, and we survey some of the previous work in Section 6.8. Section 6.9 concludes the chapter.

6.2 Preliminaries

In this section, we introduce T-Match's problem statement and T-Match's entities.

6.2.1 Problem Statement

A storage only tag T_i in T-Match stores a state that encodes its attribute a_{T_i}. By solely relying on the states of any pair of tags T_i and T_j, a reader R_k in the supply chain has to decide whether tags T_i and T_j match or not.

A first solution to tackle this problem could be encrypting the state of tags. When two tags T_i and T_j are in the range of an authorized reader R_k, reader R_k decrypts the content of tags T_i and T_j. Finally, based on a set of matching references Ref, reader R_k decides whether T_i and T_j match or not.

However, the solution above has two limitations: **first,** if the underlying encryption is not IND-CPA, tags will be sending the same ciphertexts whenever queried. This enables any eavesdropper to track tags, and consequently, enables eavesdroppers to violate tag unlinkability. **Second,** it does not ensure tag privacy against readers R_k. The solution relies on disclosing the tags' attributes to readers R_k in the supply chain.

Although, the first limitation can be tackled by using an IND-CPA encryption, the second limitation is difficult to address, as tags cannot perform any computation.

We recall that our main goal is to enable readers R_k to perform tag matching for any pair of tags T_i and T_j while preserving the privacy of tags. That is, at the end of the matching

protocol, *a reader R_k only gets the outcome of a boolean function* Check *which outputs a bit $b = 1$ if tags T_i and T_j match, otherwise, it outputs $b = 0$.*

A straightforward solution to address the problem above is to use homomorphic encryption. Homomorphic encryption enables readers R_k to compute the encrypted value $\mathsf{Enc}(\mathsf{Check}(T_i, T_j))$ using the encrypted value $\mathsf{Enc}(a_{T_i})$ of attribute a_{T_i} stored in tag T_i and the encrypted value $\mathsf{Enc}(a_{T_j})$ of attribute a_{T_j} stored in tag T_j.

However, a limitation of this approach arises when we allow readers to decrypt the ciphertext $\mathsf{Enc}(\mathsf{Check}(T_i, T_j))$: if a reader R_k is allowed to decrypt $\mathsf{Enc}(\mathsf{Check}(T_i, T_j))$, then by the same means, it can decrypt $\mathsf{Enc}(a_{T_i})$ and $\mathsf{Enc}(a_{T_j})$, leading to the potential disclosure of the tag attributes to readers in the supply chain.

An idea to overcome this limitation consists of preventing readers from decrypting ciphertexts by themselves. This calls for the use of secret sharing techniques (145). We identify two methods to implement secret sharing: the **first** method relies on distributing secret shares to readers and tags. The idea would be to allow a reader R_k to decrypt only when it reads a pair of tags T_i and T_j that match. Yet, such a solution requires that tags T_i in the system are either active and able to perform cryptographic operations, or synchronized by readers. The **second** method relies on an additional third-party component that is a backend server S. S possesses the set Ref of matching references. Readers R_k and backend server S hold secret shares of some secret key sk that allows backend server S and any reader R_k to evaluate securely $\mathsf{Check}(T_i, T_j)$.

T-MATCH relies on the second method to implement item matching. That is, in addition to readers R_k which read and re-encrypt the content of tags, T-MATCH involves a backend server S that stores the set Ref of matching references for any pair of attributes that match. Despite the fact that this approach requires backend server S to be always online with readers R_k, it remains realistic. We stress that today, even handheld RFID readers can establish continuous connection with backend server S using wireless technologies such as Bluetooth, ZigBee, WiFi or even 3G. Furthermore, having a backend server S allows for using techniques of secure multi-party computation to ensure that at the end of an execution of T-MATCH, readers R_k and backend server S learn at most the output of the Check function.

Now to check whether a pair of tags T_i and T_j match, a reader R_k reads first the encrypted states stored into T_i and T_j, then R_k contacts backend server S in order to securely evaluate the Check function for T_i and T_j. The Check function has as input the encrypted states of tags T_i and T_j along with the matching references Ref of backend server S. At the end of a T-MATCH's execution, reader R_k gets the output of the Check function.

6.2.2 T-MATCH's Setup

T-MATCH involves the following entities:

- **Tags T_i:** Each tag is attached to an item (container, barrel, ...). A tag T_i is equipped

with a re-writable memory storing T_i's current "state" denoted $S^j_{T_i}$. The state $S^j_{T_i}$ encodes and *encrypts* an attribute $a_{T_i} \in \mathbb{A}$, where $\mathbb{A}$ is the set of valid attributes in T-MATCH. We denote $\mathcal{T}$ the set of tags in T-MATCH, and we assume that $|\mathbb{A}| = l$ and $|\mathcal{T}| = n$.

- **Issuer I:** The issuer I initializes tags. It chooses an attribute $a_{T_i} \in \mathbb{A}$, then computes an initial state $S^0_{T_i}$, and finally writes the state $S^0_{T_i}$ into T_i.

- **Readers R_k:** A reader R_k in the supply chain interacts with tags T_i in its vicinity. R_k reads the states $S^{k_i}_{T_i}$ and $S^{k_j}_{T_j}$ stored into tags T_i and T_j respectively by calling the function Read, and updates the states $S^{k_i}_{T_i}$ and $S^{k_j}_{T_j}$ accordingly. Next, R_k writes the new states $S^{k_i+1}_{T_i}$ and $S^{k_j+1}_{T_j}$ into T_i and T_j by calling the function Write. Finally, R_k engages in a *two party protocol* with backend server S to compute securely a boolean function Check. R_k's input to Check is the states $S^{k_i}_{T_i}$, $S^{k_j}_{T_j}$, and its secret share α_{R_k}. If Check outputs $b = 1$, then reader R_k raises an alarm meaning that T_i and T_j match. Otherwise, T_i and T_j do not match and reader R_k does nothing. Without loss of generality, we assume that the supply chain comprises η readers R_k.

- **Backend server S:** Backend server S stores a set of ν matching references $\mathsf{Ref} = \{\mathsf{ref}_1, \mathsf{ref}_2, ..., \mathsf{ref}_\nu\}$. Backend server S is required to compute a boolean function Check jointly with reader R_k. Backend server S's input to the Check function is its set of matching references Ref and its secret share α_S.

6.3 Adversary Models

We recall that in secure multiparty computation protocols, two adversary models are identified: *semi-honest* and *malicious* in compliance with the work of Goldreich (70).

- ***Semi-honest model*:** Readers R_k and backend server S are assumed to act according to the protocol with the exception that each party keeps a record of all its computations.

- ***Malicious model*:** An adversary $\mathcal{A} \in \{R_k, S\}$ in this model may act arbitrarily. Adversary $\mathcal{A}$ may **i.)** refuse to participate in the protocol when the protocol is first invoked. $\mathcal{A}$ may as well **ii.)** substitute its local input: this corresponds for instance to a reader R_k providing an input that does not match the states of tags it has just read, or to backend server S submitting a set of bogus matching references as its local input. $\mathcal{A}$ may also **iii.)** abort the protocol before sending its last message.

In (70), Goldreich established the following result: if trapdoor permutations exist, then any secure and privacy preserving protocol against semi-honest adversaries can be compiled into a secure and a privacy preserving protocol against malicious adversaries. The idea is

to force the parties participating in the protocol to behave in a protocol compliant manner using namely commitment schemes and zero knowledge proofs.

We point out however that it is infeasible to force readers R_k and backend server S to behave according to the protocol when interacting with tags in the supply chain, as tags cannot perform any computation. Yet we believe that in the real world, it is hard for readers R_k and backend server S to deviate from the protocol arbitrarily without being detected. Note that it is always feasible to verify whether a reader R_k raises an alarm when it should or not. Whereas, it is hard to prevent readers R_k and backend server S from keeping records of their previous protocol executions or from eavesdropping on tags in the system.

Hence, in the sequel of this chapter, we assume that *readers R_k and backend server S are semi-honest*, i.e., they behave in compliance with T-MATCH. We assume as well that *issuer I is honest*, meaning that when I initializes some tag, then this tag correctly encodes the attribute of the item to which it is attached.

Now, to formally capture the capabilities of an adversary $\mathcal{A}$ against the security and the privacy of T-MATCH, a challenger $\mathcal{C}$ provides adversary $\mathcal{A}$ with access to the following oracles:

- $\mathcal{O}_{\mathsf{Tag}}(\mathsf{param})$: When queried with a parameter param, the oracle $\mathcal{O}_{\mathsf{Tag}}(\mathsf{param})$ returns a tag based on the value of the parameter chosen by $\mathcal{A}$. For instance, if $\mathsf{param} = a_i \in \mathbb{A}$, then $\mathcal{O}_{\mathsf{Tag}}$ returns a tag that encodes attribute a_i.
- $\mathcal{O}_{\mathsf{Check}}(T_i, T_j)$: When queried with a pair of tags T_i and T_j, the oracle $\mathcal{O}_{\mathsf{Check}}$ returns a bit $b = \mathsf{Check}(T_i, T_j)$. If $b = 1$, then this entails that T_i and T_j store a pair of attributes that match; otherwise, they do not.
- $\mathcal{O}_{\mathsf{Flip}}(\mathsf{T}_0, \mathsf{T}_1)$: When queried with two tags T_0 and T_1, $\mathcal{O}_{\mathsf{Flip}}$ flips a fair coin $b \in \{0, 1\}$. If $b = 1$, then $\mathcal{O}_{\mathsf{Flip}}$ returns tag T_1; otherwise, it returns tag T_0.

6.3.1 Security

In the following, we introduce the security requirements of T-MATCH.

6.3.1.1 Completeness

Completeness ensures that if two tags T_i and T_j store a pair of matching attributes, then $\mathsf{Check}(T_i, T_j)$ outputs $b = 1$.

Definition 6.1 (Completeness)**.** T-MATCH *is complete* $\Leftrightarrow$ *For any pair of tags* (T_i, T_j) *that store a pair of matching attributes,* $\mathsf{Check}(T_i, T_j) = 1$.

Denial of service. Similarly to the tracking protocols proposed in Chapter 5, an adversary $\mathcal{A}$ against T-MATCH can spoil the "completeness" property by writing any content "garbage" into tags. As discussed previously, RFID protocols that rely on storage only tags are vulnerable to denial of service attacks, since these tags do not implement any reader authentication

mechanism. However, if T-MATCH is used in an application scenario where denial of service attacks may result in real physical threats to the supply chain, then the partners in the supply chain may decide to use more "intelligent" and "expensive" tags that can implement T-MATCH on top of a reader authentication protocol. It is clear that there is a trade-off between tags' cost and resistance to denial of service, and that is depending on the nature of the items participating in the matching protocol and the trust level between the partners of the supply chain, these partners can decide whether to use "cheap" storage only tags or to use more "expensive" tags.

6.3.1.2 Soundness

Soundness assures that if the Check function outputs $b = 1$, then this entails that the tags T_i and T_j presented to reader R_k encode a pair of attributes a_{T_i} and a_{T_j} that match with an overwhelming probability.

We formalize soundness using a game-based definition as depicted in Algorithm 6.3.1 and Algorithm 6.3.2. In the learning phase, challenger $\mathcal{C}$ calls the oracle $\mathcal{O}_{\mathsf{Tag}}$ that supplies $\mathcal{A}$ with r tags T_i. $\mathcal{A}$ is allowed to read and write into tags T_i. He can also query the oracle $\mathcal{O}_{\mathsf{Check}}$ with any tag from the set of r tags T_i for a maximum of s times.

Algorithm 6.3.1: Learning phase of the soundness game

for $i := 1$ **to** r **do**
  $T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_i)$;
  for $j := 1$ **to** s **do**
    $S^j_{T_i} = \mathsf{Read}(T_i)$;
    $\mathsf{Write}(T_i, S'^j_{T_i})$;
    $T_{(i,j)} \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{(i,j)})$;
    $S_{T_{(i,j)}} = \mathsf{Read}(T_{(i,j)})$;
    $\mathsf{Write}(T_{(i,j)}, S'_{T_{(i,j)}})$;
    $b_{(i,j)} \leftarrow \mathcal{O}_{\mathsf{Check}}(T_i, T_{(i,j)})$;

Algorithm 6.3.2: Challenge phase of the soundness game

$(\mathsf{T}_0, \mathsf{T}_1) \leftarrow \mathcal{A}$; // $\mathcal{A}$ submits tags T_0 and T_1 to challenger $\mathcal{C}$
$b \leftarrow \mathcal{O}_{\mathsf{Check}}(\mathsf{T}_0, \mathsf{T}_1)$;

In the challenge phase, adversary $\mathcal{A}$ submits two challenge tags T_0 and T_1 to challenger $\mathcal{C}$ who queries the oracle $\mathcal{O}_{\mathsf{Check}}$ with tags T_0 and T_1. Finally, the oracle $\mathcal{O}_{\mathsf{Check}}$ outputs a bit b.

Adversary $\mathcal{A}$ is said to win the soundness game, if **i.)** $b = 1$ and if **ii.)** T_0 and T_1 encode two attributes a_{T_0} and a_{T_1} that do not match.

The advantage ϵ of adversary $\mathcal{A}$ in winning the soundness game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins})$$

Definition 6.2 (Soundness)**.** T-MATCH *is sound,* **iff** *for any adversary* $\mathcal{A}(r, s, \epsilon)$*, the advantage* ϵ *in winning the soundness game is negligible.*

The definition above captures the capabilities of an active adversary $\mathcal{A}$, who in addition to being able to read tags, can re-write their internal states. The adversarial goal of $\mathcal{A}$ is to provide a pair of tags T_0 and T_1 which do not store matching attributes, yet $\mathsf{Check}(\mathsf{T}_0, \mathsf{T}_1)$ outputs 1.

6.3.2 Privacy

T-MATCH is said to be privacy preserving, with respect to tags in the supply chain *if the only information learned by an adversary* $\mathcal{A}$ *after executing* T-MATCH *with a pair of tags* T_i *and* T_j *is the output of* $\mathsf{Check}(T_i, T_j)$. That is, adversary $\mathcal{A}$ only learns whether tags T_i and T_j match or not.

Along these lines, we define first T-MATCH's privacy against readers R_k and backend server S, so as to measure information leakage through reader and backend server interaction. Second, we define T-MATCH's privacy against an *outsider adversary* $\mathcal{A} \notin \{R_k, S\}$ to quantify information leakage through the wireless channel between tags and readers R_k in the supply chain.

6.3.2.1 Privacy against Readers and Backend Server

In accordance with previous work on secure two-party computation (70), we define privacy of T-MATCH against readers R_k and backend server S in the semi-honest model by considering, first an ideal model in which both parties communicate their inputs to a TTP that computes the output of the Check function for reader R_k and backend server S. Then, we consider an execution of T-MATCH which evaluates the Check function in the real model without a TTP as depicted in Figure 6.3.2.1.

T-MATCH is said to be privacy preserving against readers R_k and backend server S, if for every semi-honest behavior of one of the parties (reader R_k or backend server S), the joint view of both parties can be simulated by a computation of the Check function in the ideal model, where also one party is semi-honest and the other is honest. That is, T-MATCH does not leak information about the private inputs of readers R_k and backend server S.

Definition 6.3 (Privacy against reader R_k and backend server S (70))**.** *Let* $\bar{\mathcal{A}} = (\mathcal{A}_1, \mathcal{A}_2)$ *be an admissible pair representing adversarial behavior by reader* R_k *and backend server* S *in the real model. Such a pair is admissible if at least one party* $\mathcal{A}_i$ *is honest.*

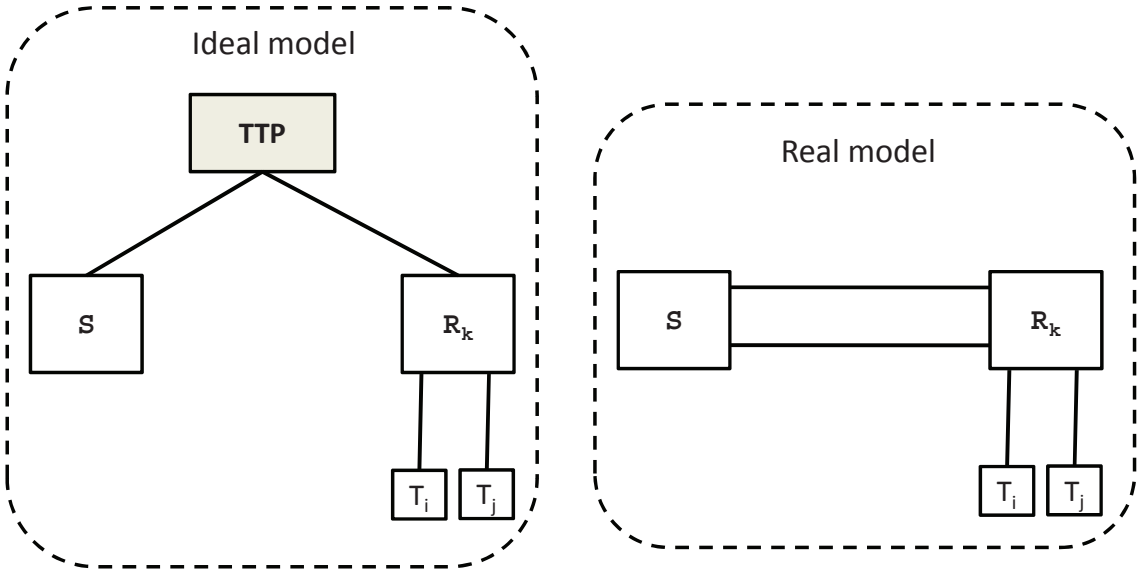

Figure 6.1: Computing the Check function in both the ideal model and the real model

- *On input pair* (X, Y) *(*X *is* R_k*'s input and* Y *is* S*'s input), let* $\mathsf{View}_1 = (X, r, M_1, ..., M_p, \mathsf{Check}(X, Y))$ *denote the view of reader* R_k*, where* r *is the outcome of* R_k*'s internal randomness, and* M_i *is the* i^{th} *message that* R_k *has received.*

- *Let* $\mathsf{View}_2 = (Y, r', M'_1, ..., M'_q, \perp)$ *denote the view of backend server* S*, where* r' *is the outcome of* S*'s internal randomness, and* M'_i *is the* i^{th} *message that* S *has received.*

We denote the joint execution under $\bar{\mathcal{A}}$ *in the real model on input pair* (X, Y) $\mathsf{Real}_{\bar{\mathcal{A}}}(X, Y)$*, and it is defined as* $(\mathcal{A}_1(\mathsf{View}_1), \mathcal{A}_2(\mathsf{View}_2))$*.*

Let $\bar{\mathcal{B}} = (\mathcal{B}_1, \mathcal{B}_2)$ *be an admissible pair representing adversarial behavior by reader* R_k *and backend server* S *in the ideal model.*

We denote the joint execution under $\bar{\mathcal{B}}$ *in the ideal model on input pair* (X, Y) $\mathsf{Ideal}_{\bar{\mathcal{B}}}(X, Y)$*, and it is defined as* $(\mathcal{B}_1(X, \mathsf{Check}(X, Y)), \mathcal{B}_2(Y, \perp))$*.*

T-Match *is said to be privacy preserving with respect to reader* R_k *and backend server* S *in the semi-honest model, if there is a transformation of pairs of admissible adversaries* $\bar{\mathcal{A}} = (\mathcal{A}_1, \mathcal{A}_2)$ *in the real model, into pairs of admissible adversaries* $\bar{\mathcal{B}} = (\mathcal{B}_1, \mathcal{B}_2)$ *in the ideal model, so that the distributions* $\{\mathsf{Real}_{\bar{\mathcal{B}}}(X, Y)\}_{X,Y}$ *and* $\{\mathsf{Ideal}_{\bar{\mathcal{B}}}(X, Y)\}_{X,Y}$ *are computationally indistinguishable.*

Remark 6.1. *Using the notations of Section 6.2.2, we indicate that:*

- *the input of* X *of reader* R_k *to* T-Match *is defined as its secret share* α_{R_k} *and the states* $S_{T_i}^{k_i}$ *and* $S_{T_i}^{k_j}$ *of tags* T_i *and* T_j *respectively;*

- *the input Y of backend server S to* T-MATCH *is its set of matching references* Ref *and its secret share α_S;*
- *at the end of* T-MATCH*'s execution, only reader R_k gets the bit $b = \mathsf{Check}(T_i, T_j)$.*

Remark 6.2 (Readers and backend server collusion)**.** *In the definition above of the privacy of* T-MATCH *against readers R_k and backend server S, it is assumed that at least one party is honest. This implies that we implicitly assume that readers R_k and backend server S do not collude against tags participating in the protocol. Notice that if readers R_k and backend server S collude against tags in* T-MATCH*, then tag privacy cannot be ensured. Readers R_k and backend server S can use their respective secret shares α_{R_k} and α_S to reveal tags' attributes without invoking* T-MATCH*.*

6.3.2.2 Privacy against Outsiders

Ideally, a privacy preserving protocol for tag matching against an *outsider* adversary $\mathcal{A}$ should provide tag unlinkability. As discussed in previous chapters, tag unlinkability is the privacy property that ensures that it is computationally infeasible for an adversary $\mathcal{A}$ to tell two tags T_i and T_j apart.

However, we note that any adversary $\mathcal{A}$ who has access to the output of the Check function can mount a trivial attack against tag unlinkability. In fact, to break tag unlinkability for a pair of tags (T_i, T_j), all $\mathcal{A}$ has to do is to run T-MATCH, first with a pair of tags (T_i, T_k) and then with another pair of tag (T_j, T_k). Next, if $\mathsf{Check}(T_i, T_k) \neq \mathsf{Check}(T_j, T_k)$, then $\mathcal{A}$ concludes that T_i and T_j encode different attributes, and by the same token, he concludes that T_i and T_j are different tags, breaking hereby tag unlinkability.

Also, as in Chapter 5, it is impossible to ensure *tag unlinkability* against an adversary who monitors all of tags' interactions. We recall that T-MATCH targets storage only tags and therewith it relies on readers R_k to update tags' states, and as a result, a tag's state does not change in between two protocol executions. Accordingly, we relax again the definition of tag unlinkability, by assuming that there is at least one unobserved interaction between tags and an honest reader R_k outside the range of adversary $\mathcal{A}$.

Now in accordance with previous chapters, we use an indistinguishability based definition to formalize tag unlinkability.

In the learning phase as depicted in Algorithm 6.3.3, challenger $\mathcal{C}$ provides adversary $\mathcal{A}$ with access to the oracle $\mathcal{O}_{\mathsf{Tag}}$ that $\mathcal{A}$ can query to get a set of r tags which he can read from and write into, and for which he can query the oracle $\mathcal{O}_{\mathsf{Check}}$ for a maximum of s times.

In the challenge phase, cf. Algorithm 6.3.4, $\mathcal{A}$ generates two challenge tags T_0 and T_1 that he submits to challenger $\mathcal{C}$. These two tags are read outside the range of adversary $\mathcal{A}$, then they are submitted to the oracle $\mathcal{O}_{\mathsf{Flip}}$. Next, the oracle $\mathcal{O}_{\mathsf{Flip}}$ supplies $\mathcal{A}$ with tag T_b, $b \in \{0, 1\}$. Finally, $\mathcal{A}$ outputs his guess b' for the value of b.

$\mathcal{A}$ is said to win the tag unlinkability game if $b = b'$.

Algorithm 6.3.3: Learning phase of the tag unlinkability game

for $i := 1$ **to** r **do**
  $T_i \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_i)$;
  for $j := 1$ **to** s **do**
    $S^j_{T_i} = \mathsf{Read}(T_i)$;
    $\mathsf{Write}(T_i, S'^j_{T_i})$;
    $T_{(i,j)} \leftarrow \mathcal{O}_{\mathsf{Tag}}(\mathsf{param}_{(i,j)})$;
    $S_{T_{(i,j)}} = \mathsf{Read}(T_{(i,j)})$;
    $\mathsf{Write}(T_{(i,j)}, S'_{T_{(i,j)}})$;
    $\mathcal{O}_{\mathsf{Check}}(T_i, T_{(i,j)})$;

Algorithm 6.3.4: Challenge phase of the tag unlinkability game

$(\mathsf{T}_0, \mathsf{T}_1) \leftarrow \mathcal{A}$; // $\mathcal{A}$ submits T_0 and T_1 to challenger $\mathcal{C}$
// T_0 and T_1 are read outside the range of $\mathcal{A}$ by some reader R_k in the supply chain
$\mathsf{T}_b \leftarrow \mathcal{O}_{\mathsf{Flip}}(\mathsf{T}_0, \mathsf{T}_1)$;
$\mathsf{Read}(\mathsf{T}_b)$;
Output b';

The advantage ϵ of adversary $\mathcal{A}$ in wining the tag unlinkability game is defined as:

$$\epsilon = Pr(\mathcal{A} \text{ wins}) - \frac{1}{2}$$

Definition 6.4 (Tag Unlinkability)**.** T-MATCH *is said to ensure tag unlinkability,* **iff** *for any adversary $\mathcal{A}(r, s, \epsilon)$, the advantage ϵ in winning the tag unlinkability game is negligible.*

Roughly speaking, the above definition of tag unlinkability ensures that if a pair of tags T_i and T_j interact with an honest reader outside the range of a *narrow* adversary[9] $\mathcal{A}$ at least once, then it is computationally infeasible for adversary $\mathcal{A}$ to distinguish between tags T_i and T_j.

6.4 Protocol

To perform tag matching in T-MATCH, we store into each tag T_i an IND-CPA homomorphic encryption $\mathsf{Enc}(a_{T_i})$ of its attribute a_{T_i}. When reader R_k reads a pair of tags T_i and T_j, it uses the homomorphic property of Enc to compute an encryption $C_{(i,j)}$ of a function f of T_i and T_j's attributes, i.e., $C_{(i,j)} = \mathsf{Enc}(f(a_{T_i}, a_{T_j}))$.

Now, the matching reference of any pair of attributes (a_i, a_j) is computed as $\mathsf{ref}_{(i,j)} = f(a_i, a_j)$. To evaluate the Check function, reader R_k and backend server S rely on a two

[9] An adversary who does not always access the oracle $\mathcal{O}_{\mathsf{Check}}$ (159).

party privacy preserving *plaintext equality test* (84) (PET for short) to decide whether $C_{(i,j)}$ encrypts one of S's matching references or not.

Although, it may seem that any IND-CPA homomorphic encryption such as Elgamal or Paillier could suit the privacy and the security requirements of T-MATCH when readers R_k in the supply chain and backend server S are semi-honest, not all of them prevent backend server S from forging new matching references from its initial set Ref. We recall that Elgamal is multiplicatively homomorphic and thus the function f is going to be expressed as $f(a_i, a_j) = \psi(a_i)\psi(a_j) = \mathsf{ref}_{(i,j)}$, where ψ is the attribute encoding in T-MATCH. We note also that Paillier is additively homomorphic, and as a consequence: $f(a_i, a_j) = \psi(a_i) + \psi(a_j) = \mathsf{ref}_{(i,j)}$.

Therefore, neither the use of Elgamal nor Paillier as the underlying encryption technique can stop backend sever S from forging a new matching reference ref from its set Ref.

To prevent forgery of matching references, we use Boneh-Goh-Nissim (BGN) encryption (26). In addition to being multiplicatively homomorphic, BGN encryption allows computing an encryption of a bilinear pairing of two plaintexts from their ciphertexts. Consequently, a matching reference of two attributes a_i and a_j in T-MATCH is computed as: $\mathsf{ref}_{(i,j)} = f(a_i, a_j) = f(a_j, a_i) = e(\psi(a_i), \psi(a_j))$, where ψ is the attribute encoding in T-MATCH. We show that in this case, forging a new matching reference ref from Ref is as hard as the bilinear computational Diffie-Hellman problem, see Appendix A.

Now, we introduce the definitions and the notations that will be used in this chapter.

6.4.1 Tools

T-MATCH uses the BGN cryptosystem which takes place in subgroups of finite composite order that support bilinear pairings of type 1, (see Section 2.3.3, Remark 2.5) as in previous work of Katz et al. (97).

6.4.1.1 Boneh-Goh-Nissim (BGN) Cryptosystem

We now describe Boneh-Goh-Nissim (BGN) cryptosystem that we employ to encrypt tags' attributes in T-MATCH.

- **Key generation:** On input of a security parameter τ, the system obtains a tuple $(q_1, q_2, \mathbb{G}, \mathbb{G}_T, e)$ such that:

  1. q_1 and q_2 are two random primes. Typically, $|q_1| = |q_2| = 512$ bits;
  2. $\mathbb{G}$ is a bilinear group of composite order $N = q_1 q_2$;
  3. $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear pairing of type 1.

  The system then picks up two random generators $g, u \in \mathbb{G}$ and sets $\tilde{g}_1 = u^{q_2}$. Finally, the system outputs the public key $\mathsf{pk} = (N, \mathbb{G}, \mathbb{G}_T, e, g, \tilde{g}_1)$ and the secret key $\mathsf{sk} = q_1$.

- **Encryption:** The encryption algorithm is defined in both groups $\mathbb{G}$ and $\mathbb{G}_T$.

- *Encryption in* $\mathbb{G}$: On input of a message $m \in \mathbb{G}$, the encryption algorithm selects a random number $r \in \mathbb{Z}_N$ and computes $c = \mathsf{Enc}_{\mathbb{G}}(m) = m\tilde{g}_1^r$.
- *Encryption in* $\mathbb{G}_T$: On input of a message $M \in \mathbb{G}_T$, the encryption algorithm picks a random number $r \in \mathbb{Z}_N$ and computes $C = \mathsf{Enc}_{\mathbb{G}_T}(M) = Me(g, \tilde{g}_1)^r$.

- **Decryption:** Decryption in BGN relies on computing discrete logarithm in a finite group of order N. Thus, decryption takes $O(\sqrt{N})$ steps, and consequently, BGN is only suitable for encrypting short messages. However, in T-MATCH we do not decrypt any ciphertext C. For completeness purposes, we detail below the decryption algorithm of BGN.

  - *Decryption in* $\mathbb{G}$: On input of a ciphertext $c \in \mathbb{G}$ and secret key $\mathsf{sk} = q_1$, the decryption algorithm computes: $\mathcal{C} = c^{q_1} = m^{q_1}\tilde{g}_1^{rq_1}$. Since the order of $\tilde{g}_1$ is q_1, it follows that $\mathcal{C} = m^{q_1}$.

    As g is a generator of $\mathbb{G}$, there exists $x_m \in \mathbb{Z}_N$ such that: $m = g^{x_m}$. x_m is computed as $\mathsf{log}_{g^{q_1}}(\mathcal{C})$ and $\mathsf{Dec}_{\mathbb{G}}(c) = g^{x_m} = m$.
  - *Decryption in* $\mathbb{G}_T$: On input of a ciphertext $C \in \mathbb{G}_T$ and secret key $\mathsf{sk} = q_1$, the decryption algorithm computes: $\mathcal{C} = C^{q_1} = M^{q_1}e(g, \tilde{g}_1)^{rq_1} = M^{q_1}$, since the order of $e(g, \tilde{g}_1)$ is q_1.

    As $e(g, g)$ is a generator of $\mathbb{G}_T$, then there exists $x_M \in \mathbb{Z}_N$ such that: $M = e(g, g)^{x_M}$. Therefore, $\mathcal{C} = (e(g, g)^{q_1})^{x_M}$ and x_M is computed as $\mathsf{log}_{e(g,g)^{q_1}}(\mathcal{C})$. Finally, $\mathsf{Dec}_{\mathbb{G}_T}(C) = e(g, g)^{x_M} = M$.

Remark 6.3. *The Boneh-Goh-Nissim encryption takes place in supersingular curves.*

We refer to the work of Boneh et al. (26) for more details on how to construct subgroups of elliptic curves of order N that support symmetric bilinear pairings.

The BGN cryptosystem is IND-CPA under the subgroup decision assumption.

Definition 6.5 (The Subgroup Decision Assumption (26, 125))**.** *Let $\mathcal{G}$ be a group of order N where $N = q_1q_2$ is the product of two primes q_1 and q_2. The subgroup decision assumption is said to hold in $\mathcal{G}$, if given a random element u in $\mathcal{G}$, it is computationally hard to decide whether u is in the subgroup of $\mathcal{G}$ of order q_1 or not.*

Moreover, the following homomorphic properties hold:

$$\forall\ m_1, m_2 \in \mathbb{G},\ \mathsf{Enc}_{\mathbb{G}}(m_1)\mathsf{Enc}_{\mathbb{G}}(m_2) = \mathsf{Enc}_{\mathbb{G}}(m_1m_2)$$

$$e(\mathsf{Enc}_{\mathbb{G}}(m_1), \mathsf{Enc}_{\mathbb{G}}(m_2)) = \mathsf{Enc}_{\mathbb{G}_T}(e(m_1, m_2))$$

6.4.1.2 Attribute Encoding

Let $\mathbb{G}$ be a group of composite order $N = q_1 q_2$ and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear pairing.

We denote $\mathbb{G}_1$ and $\mathbb{G}_2$ the subgroups of $\mathbb{G}$ of order q_1 and q_2 respectively.

We also denote $\mathbb{G}_{T_1}$ and $\mathbb{G}_{T_2}$ the subgroups of $\mathbb{G}_T$ of order q_1 and q_2 respectively.

Let g, u be two random generators of $\mathbb{G}$. By construction, $\tilde{g}_1 = u^{q_2}$ is a generator of $\mathbb{G}_1$ and $\tilde{g}_2 = g^{q_1}$ is a generator of $\mathbb{G}_2$.

Let $x_I = q_1 {x'}_I \bmod N$ be the issuer's secret key, where x'_I is randomly selected in $\mathbb{Z}_N^*$.

An attribute a_i in T-MATCH is encoded as $\psi(a_i) = H(a_i)^{x_I}$, where $H : \{0,1\}^* \rightarrow \mathbb{G}$ is a cryptographic hash function.

To evaluate H, issuer I can use the algorithm proposed by Icart (81) that hashes into elliptic curves.

We note that:

$$\begin{aligned} \forall a_i \in \mathbb{A},\ \exists x_i \in \mathbb{Z}_N^* \text{ such that: } \psi(a_i) &= H(a_i)^{x_I} = (g^{x_i})^{x_I} = g^{x_i x_I} \\ &= g^{x_i q_1 x'_I} = (g^{q_1})^{x_i x'_I} = \tilde{g}_2^{x_i x'_I} \in \mathbb{G}_2, \end{aligned}$$

And it follows that:

$$\forall (a_i, a_j) \in \mathbb{A}^2, e(\psi(a_i), \psi(a_j)) \in \mathbb{G}_{T_2}$$

6.4.2 T-MATCH Overview

Here we provide an overview of T-MATCH that summarizes how the matching protocol works.

Each tag T_i stores a state $S_{T_i}^{k_i}$ that consists of a BGN encryption $c_{T_i}^{k_i} = \mathsf{Enc}_{\mathbb{G}}(\psi(a_{T_i})) = \mathsf{Enc}_{\mathbb{G}}(H(a_{T_i})^{x_I})$ of T_i's attribute a_{T_i} (where $H : \{0,1\}^* \rightarrow \mathbb{G}_T$ is a cryptographic hash function, and x_I is the secret key of issuer I), together with a MAC $\sigma_{T_i}^{k_i} = \mathsf{MAC}_K(c_{T_i}^{k_i})$, i.e., $S_{T_i}^{k_i} = (c_{T_i}^{k_i}, \sigma_{T_i}^{k_i})$. Whereas backend server S stores a set Ref of ν matching references. Each matching reference $\mathsf{ref}_{(i,j)}$ corresponds to two attributes a_i and a_j in $\mathbb{A}$ that match and it is computed as:

$$\mathsf{ref}_{(i,j)} = f(a_i, a_j) = f(a_j, a_i) = e(\psi(a_i), \psi(a_j)) = e(H(a_i)^{x_I}, H(a_j)^{x_I})$$

When two tags T_i and T_j come together in the range of a reader R_k, reader R_k reads the current states $S_{T_i}^{k_i}$ and $S_{T_j}^{k_j}$ of tags T_i and T_j's respectively. Reader R_k checks first, whether the keyed MAC stored into tags T_i and T_j are valid or not. If they are, reader R_k computes the bilinear pairing $e(c_{T_i}^{k_i}, c_{T_j}^{k_j})$.

$$\begin{aligned} C_{(i,j)} &= e(c_{T_i}^{k_i}, c_{T_j}^{k_j}) = e(\mathsf{Enc}_{\mathbb{G}}(\psi(a_{T_i})), \mathsf{Enc}_{\mathbb{G}}(\psi(a_{T_j}))) \\ &= \mathsf{Enc}_{\mathbb{G}_T}(e(\psi(a_{T_i}), \psi(a_{T_j}))) \end{aligned}$$

Next, reader R_k and backend server S engage in a secure two party protocol for *plaintext*

equality test (PET) to check whether the underlying plaintext of ciphertext $C_{(i,j)}$ belongs to the set of matching references Ref of backend server S or not. That is, reader R_k and backend server S check whether:

$$\exists\ \mathsf{ref}_p \in \mathsf{Ref},\ C_{(i,j)} = \mathsf{Enc}_{\mathbb{G}_T}(\mathsf{ref}_p)$$

Now, a reader R_k outputs $b = 1$ (i.e., $\mathsf{Check}(T_i, T_j) = 1$), if the plaintext equality test outputs 1; otherwise, it outputs $b = 0$.

Privacy and security overview. To protect the privacy of tags, a tag T_i in T-MATCH stores a BGN encryption of its attribute a_{T_i} and a keyed MAC of the encryption. In each protocol execution, the BGN encryption is re-encrypted by readers R_k and the MAC is computed accordingly. Now, to protect the privacy of tags that participate in the matching protocol against readers R_k and backend server S, we rely on a *modified* privacy preserving plaintext equality test that is run jointly by some reader R_k and backend server S. Moreover, T-MATCH uses shuffling techniques to ensure that the only information leaked at the end of the matching protocol is a bit b that indicates whether the pair of tags participating in the current execution of T-MATCH match or not.

Furthermore, to prevent backend server S from forging new matching references from the set Ref, attributes in T-MATCH are encoded as "signatures" by issuer I, and the matching references are computed as bilinear pairings. Finally, T-MATCH relies on a keyed MAC to prevent adversaries (intruders) from tampering with tags' content without being detected.

6.4.3 Protocol Description

We now describe in more details how T-MATCH performs tag matching.

6.4.3.1 System Setup

A trusted third party (TTP) outputs a matching pair of BGN public key $\mathsf{pk} = (N, \mathbb{G}, \mathbb{G}_T, e, g, \tilde{g}_1)$ and secret key $\mathsf{sk} = q_1$, a cryptographic hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_T$, a secret key $x_I = q_1 x'_I \bmod N$ where x'_I is selected randomly in $\mathbb{Z}_N^*$, and a MAC key K. The TTP selects randomly a secret share $\alpha_1 \in \mathbb{Z}_N$, then it sets the second secret share to $\alpha_2 = \mathsf{sk} - \alpha_1 = q_1 - \alpha_1 \bmod N$.

Next, the TTP computes the set Ref of matching references. On input of attribute $a_i \in \mathbb{A}$, TTP computes $\psi(a_i) = H(a_i)^{x_I} \in \mathbb{G}_2$. If two attributes a_i and a_j match, then the TTP computes the corresponding matching reference $\mathsf{ref}_{(i,j)} = e(\psi(a_i), \psi(a_j)) = e(\psi(a_j), \psi(a_i)) \in \mathbb{G}_{T_2}$.

Finally, the TTP supplies

- each reader R_k with its share $\alpha_{R_k} = \alpha_1$ of secret key sk and with the MAC key K;

- backend server S with its share $\alpha_S = \alpha_2$ of secret key sk and with the set of matching references Ref;

- issuer I with the hash function H, secret key $x_I = q_1 x'_I \bmod N$ and the MAC key K.

6.4.3.2 Tag Initialization

For each new tag T_i, issuer I computes $\psi(a_{T_i}) = H(a_{T_i})^{x_I}$, such that a_{T_i} is the attribute associated with the chemical in the container that T_i will label. Then, using the BGN public key pk, issuer I picks a random number r_i^0 and computes a ciphertext $c_{T_i}^0 = \mathsf{Enc}_{\mathbb{G}}(\psi(a_{T_i})) = \psi(a_{T_i})\tilde{g}_1^{r_i^0}$. Finally, issuer I computes $\sigma_{T_i}^0 = \mathsf{MAC}_K(c_{T_i}^0)$ and stores into tag T_i the state $S_{T_i}^0 = (c_{T_i}^0, \sigma_{T_i}^0)$.

6.4.3.3 Tag Matching

We break down the tag matching protocol into three operations that describe, **first** the interaction between tags T_i, T_j and reader R_k, **second** the interaction between reader R_k and backend server S, and **third** the computation of the output of the Check function by reader R_k.

Tag T_i $\leftrightarrow$ Reader R_k $\leftrightarrow$ Tag T_j. Assume there are two tags T_i and T_j in the range of reader R_k. Tags T_i and T_j store states $S_{T_i}^{k_i} = (c_{T_i}^{k_i}, \sigma_{T_i}^{k_i})$ and $S_{T_j}^{k_j} = (c_{T_j}^{k_j}, \sigma_{T_j}^{k_j})$ respectively.

Reader R_k first reads out the tags T_i and T_j and checks whether $\sigma_{T_i}^{k_i} = \mathsf{MAC}_K(c_{T_i}^{k_i})$ and $\sigma_{T_j}^{k_j} = \mathsf{MAC}_K(c_{T_j}^{k_j})$ or not. If not, reader R_k updates the states of tags T_i and T_j and aborts the protocol. Otherwise, it updates the states of tags T_i and T_j and continues the execution of the protocol.

Now to update the state of tag T_i participating in the protocol, reader R_k proceeds as follows.

- If $\sigma_{T_i}^{k_i} = \mathsf{MAC}_K(c_{T_i}^{k_i})$, then reader R_k picks a random numbers r'_i and re-encrypts the ciphertexts $c_{T_i}^{k_i}$ to obtain new BGN ciphertext $c_{T_i}^{k_i+1} = c_{T_i}^{k_i}\tilde{g}_1^{r'_i}$. Then, it computes $\sigma_{T_i}^{k_i+1} = \mathsf{MAC}_K(c_{T_i}^{k_i+1})$. Finally, reader R_k writes the new state $S_{T_i}^{k_i+1} = (c_{T_i}^{k_i+1}, \sigma_{T_i}^{k_i+1})$ into tag T_i.

- If $\sigma_{T_i}^{k_i} \neq \mathsf{MAC}_K(c_{T_i}^{k_i})$, then reader R_k picks two random strings (st_1, st_2) and stores them into tag T_i.

Reader R_k then computes the BGN ciphertext $C_{(i,j)} = e(c_{T_i}^{k_i}, c_{T_j}^{k_j}) \in \mathbb{G}_T$.

Without loss of generality, we assume that $c_{T_i}^{k_i} = \mathsf{Enc}_{\mathbb{G}}(\psi(a_{T_i})) = \psi(a_{T_i})\tilde{g}_1^{r_i}$ and $c_{T_j}^{k_j} =$

$\mathsf{Enc}_{\mathbb{G}}(\psi(a_{T_j})) = \psi(a_{T_j})\tilde{g}_1^{r_j}$, $r_i, r_j \in \mathbb{Z}_N$. By bilinearity of e:

$$\begin{aligned} C_{(i,j)} &= e(c_{T_i}^{k_i}, c_{T_j}^{k_j}) = e(\psi(a_{T_i})\tilde{g}_1^{r_i}, \psi(a_{T_j})\tilde{g}_1^{r_j}) \\ &= e(\psi(a_{T_i}), \psi(a_{T_j})\tilde{g}_1^{r_j}) e(\tilde{g}_1^{r_i}, \psi(a_{T_j})\tilde{g}_1^{r_j}) \\ &= e(\psi(a_{T_i}), \psi(a_{T_j})) e(\psi(a_{T_i}), \tilde{g}_1^{r_j}) e(\tilde{g}_1^{r_i}, \psi(a_{T_j})) e(\tilde{g}_1^{r_i}, \tilde{g}_1^{r_j}) \end{aligned}$$

We recall that:

- $\tilde{g}_1 = u^{q_2}$ where u is a generator of $\mathbb{G}$, and that there exist $x \in \mathbb{Z}_N$ such that $\tilde{g}_1 = g^x$;
- $\psi(a_{T_i})$ and $\psi(a_{T_j})$ are elements of $\mathbb{G}_2$ and that $\tilde{g}_2 = g^{q_1}$ is a generator of $\mathbb{G}_2$. As a result, there exist x_i and $x_j \in \mathbb{Z}_{q_2}$ such that $\psi(a_{T_i}) = \tilde{g}_2^{x_i} = g^{q_1 x_i}$ and $\psi(a_{T_j}) = \tilde{g}_2^{x_j} = g^{q_1 x_j}$.

$$\begin{aligned} C_{(i,j)} &= e(\psi(a_{T_i}), \psi(a_{T_j})) e(g^{q_1 x_i}, u^{q_2 r_j}) e(u^{q_2 r_i}, g^{q_1 x_j}) e(g^{x r_i}, \tilde{g}_1^{r_j}) \\ &= e(\psi(a_{T_i}), \psi(a_{T_j})) e(g^{x_i}, u^{r_j})^{q_1 q_2} e(u^{r_i}, g^{x_j})^{q_1 q_2} e(g, \tilde{g}_1)^{x r_i r_j} \\ &= e(\psi(a_{T_i}), \psi(a_{T_j})) \underbrace{e(g^{x_i}, u^{r_j})^N}_{1} \underbrace{e(u^{r_i}, g^{x_j})^N}_{1} e(g, \tilde{g}_1)^{x r_i r_j} \\ &= e(\psi(a_{T_i}), \psi(a_{T_j})) e(g, \tilde{g}_1)^R = \mathsf{Enc}_{\mathbb{G}_T}(e(\psi(a_{T_i}), \psi(a_{T_j}))) \end{aligned}$$

where $R = x r_i r_j$.

This directly follows from the homomorphic property of BGN as illustrated in Section 6.4.1.1.

Reader R_k $\leftrightarrow$ Backend server S. Reader R_k then sends ciphertext $C_{(i,j)}$ to backend server S.

Without loss of generality, we assume that $\mathsf{Ref} = \{\mathsf{ref}_1, \mathsf{ref}_2, ..., \mathsf{ref}_\nu\}$, and that for all $\mathsf{ref}_p \in \mathsf{Ref}$, there exist a_i and a_j in $\mathbb{A}$, such that $\mathsf{ref}_p = e(\psi(a_i), \psi(a_j))$.

Upon receiving ciphertext $C_{(i,j)}$ from reader R_k, backend server S proceeds as follows:

- It picks ν random numbers $r_p \in \mathbb{Z}_N^*$, and computes ν ciphertexts $C_p = \left(\frac{C_{(i,j)}}{\mathsf{ref}_p}\right)^{r_p}$, for all p in $\{1, 2, ..., \nu\}$.
- On input of its secret share α_2 and ciphertexts C_p, backend server S computes $M'_p = (M_{(1,p)}, M_{(2,p)}) = (C_p, C_p^{\alpha_2})$. Next, backend server S shuffles M'_p.

  We note that by shuffling messages M'_p, T-Match prevents semi-honest readers R_k from telling whether two pairs of matching tags satisfy the same matching reference or not.
- Finally, backend server S sends M'_p to reader R_k.

The output of the Check function. When receiving M'_p from backend server S, reader R_k uses its secret share α_1 and computes:

$$\begin{aligned} M_p &= {M_{(1,p)}}^{\alpha_1} M_{(2,p)} = C_p^{\alpha_1} C_p^{\alpha_2} = C_p^{\alpha_1+\alpha_2} = C_p^{q_1} = \left(\left(\frac{C_{(i,j)}}{\mathsf{ref}_p}\right)^{r_p}\right)^{q_1} \\ &= \left(\left(\frac{e(\psi(a_{T_i}),\psi(a_{T_j}))e(g,\tilde{g}_1)^R}{\mathsf{ref}_p}\right)^{r_p}\right)^{q_1} \\ &= \left(\frac{e(\psi(a_{T_i}),\psi(a_{T_j}))}{\mathsf{ref}_p}\right)^{q_1 r_p} e(g,\tilde{g}_1)^{q_1 R r_p} \\ &= \left(\frac{e(\psi(a_{T_i}),\psi(a_{T_j}))}{\mathsf{ref}_p}\right)^{q_1 r_p} \end{aligned}$$

Note that if T_i and T_j match then there exists a matching reference $\mathsf{ref}_p \in \mathsf{Ref}$ such that: $e(\psi(a_{T_i}),\psi(a_{T_j})) = \mathsf{ref}_p$. That is:

$$\exists\ p \in \{1,2,...,\nu\} \text{ such that: } M_p = \left(\frac{e(\psi(a_{T_i}),\psi(a_{T_j}))}{\mathsf{ref}_p}\right)^{q_1 r_p} = 1$$

Consequently, if there exists $p \in \{1,2,...,\nu\}$ such that $M_p = 1$, then reader R_k outputs $b = 1$ meaning that T_i and T_j match. Otherwise, R_k outputs $b = 0$, i.e., T_i and T_j do not match.

6.5 Security Analysis

In the following section, we state the security theorems of T-Match.

We recall that backend server S and readers R_k are semi-honest, and that issuer I is honest.

6.5.1 Completeness

Theorem 6.1. T-Match *is complete.*

Proof sketch. If two tags T_i and T_j store attributes a_{T_i} and a_{T_j} that match, then there is $\mathsf{ref} \in \mathsf{Ref}$, such that $\mathsf{ref} = e(\psi(a_{T_i}),\psi(a_{T_j}))$. Therefore, one of the ν messages M_p computed by reader R_k will be equal to 1, and reader R_k will output $\mathsf{Check}(T_i,T_j) = 1$. □

6.5.2 Soundness

To prove the soundness of T-Match, we use the following lemma:

Lemma 6.1. *If* $r_p \in \mathbb{Z}_N^*$ *then:* $M_p = 1 \Leftrightarrow e(\psi(a_{T_i}),\psi(a_{T_j})) = \mathsf{ref}_p$.

Proof. Note that for all $a_i \in \mathbb{A}$, $\psi(a_i) \in \mathbb{G}_2$, and that $\tilde{g}_2 = g^{q_1}$ is a generator of $\mathbb{G}_2$.

As a result, for all $a_i \in \mathbb{A}, \exists\ x_i \in \mathbb{Z}_{q_2}$ such that $\psi(a_i) = \tilde{g}_2^{x_i} = g^{q_1 x_i}$. Consequently, there exist $x_i, x_j, x_p \in \mathbb{Z}_{q_2}$ such that:

$$\begin{aligned} e(\psi(a_{T_i}), \psi(a_{T_j})) &= e(g,g)^{q_1^2 x_i x_j} \\ \mathsf{ref}_p &= e(g,g)^{q_1^2 x_p} \end{aligned}$$

Thus, $M_p = \left(\dfrac{e(\psi(a_{T_i}), \psi(a_{T_j}))}{\mathsf{ref}_p}\right)^{q_1 r_p} = e(g,g)^{q_1^3 x r_p}$, where $x = x_i x_j - x_p \bmod q_2$.

If $M_p = e(g,g)^{q_1^3 x r_p} = 1$, then this implies that $q_1^3 x r_p = 0 \bmod N$. Since $r_p \in \mathbb{Z}_N^*$, it follows that $q_1^3 x = 0 \bmod N$ and $x = x_i x_j - x_p = 0 \bmod q_2$.

We conclude that $x_i x_j = x_p \bmod q_2$ and $q_1^2 x_i x_j = q_1^2 x_p \bmod N$, and $e(\psi(a_{T_i}), \psi(a_{T_j})) = \mathsf{ref}_p$. □

Theorem 6.2. T-MATCH *is sound under the security of MAC and the security of the hash function H.*

Proof sketch. If there is an adversary $\mathcal{A}$ who breaks the soundness property of T-MATCH, then this implies that adversary $\mathcal{A}$ is able to provide reader R_k with a pair of tags T_0 and T_1 such that:

i.) Tag T_0 (respectively T_1) stores a state $S_{\mathsf{T}_0} = (c_{\mathsf{T}_0}, \mathsf{MAC}_K(c_{\mathsf{T}_0}))$ (respectively $S_{\mathsf{T}_1} = (c_{\mathsf{T}_1}, \mathsf{MAC}_K(c_{\mathsf{T}_1}))$);

ii.) $\mathsf{Check}(\mathsf{T}_0, \mathsf{T}_1) = 1$, i.e., there exists a matching reference $\mathsf{ref}_{(p,q)} = e(\psi(a_p), \psi(a_q))$ that matches the pair of tags T_0 and T_1;

iii.) and finally, $\{\mathsf{Dec}_{\mathbb{G}}(c_{\mathsf{T}_0}), \mathsf{Dec}_{\mathbb{G}}(c_{\mathsf{T}_1})\} \neq \{\psi(a_p), \psi(a_q)\}$.

There are two cases to consider, depending on whether T_0 and T_1 encode valid attributes or not.

Case 1. T_0 and T_1 encode valid attributes, i.e., there exist $a_i, a_j \in \mathbb{A}$ such that $\mathsf{Dec}_{\mathbb{G}}(c_{\mathsf{T}_0}) = \psi(a_i)$ and $\mathsf{Dec}_{\mathbb{G}}(c_{\mathsf{T}_1}) = \psi(a_j)$. Breaking the soundness property of T-MATCH implies that there exist $\{a_i, a_j\} \neq \{a_p, a_q\} \subset \mathbb{A}$ such that $\mathsf{ref}_{(p,q)} = e(\psi(a_p), \psi(a_q)) = e(\psi(a_i), \psi(a_j))$ using Lemma 1.

- Let E denote the event that for all $\{a_i, a_j\} \neq \{a_p, a_q\} \subset \mathbb{A}$, $e(\psi(a_i), \psi(a_j)) \neq e(\psi(a_p), \psi(a_q))$.
- Let $\bar{E}$ denote the event that there exists $\{a_i, a_j\} \neq \{a_p, a_q\} \subset \mathbb{A}$, such that $e(\psi(a_i), \psi(a_j)) = e(\psi(a_p), \psi(a_q))$.

Assuming that $H : \{0,1\}^* \rightarrow \mathbb{G}$ is a cryptographic hash function implies that for all $a_i \in \mathbb{A}$, $H(a_i)$ is uniformly distributed in $\mathbb{G}$. Therefore, $\psi(a_i) = H(a_i)^{x_I} = H(a_i)^{q_1 x_I'}$ is randomly

distributed in $\mathbb{G}_2$, i.e., for all $a_i \in \mathbb{A}$ there exists x_i uniformly distributed in $\mathbb{Z}_{q_2}$ such that $\psi(a_i) = \tilde{g}_2^{x_i}$, where $\tilde{g}_2$ is a random generator of $\mathbb{G}_2$.

Accordingly, for any pair of attributes $(a_i, a_j) \in \mathbb{A}$, $e(\psi(a_i), \psi(a_j)) = e(\tilde{g}_2, \tilde{g}_2)^{x_i x_j}$ is distributed uniformly in the subgroup $\mathbb{G}_{T_2}$ of order q_2 in $\mathbb{G}_T$.

- Let $\mathsf{P}_{\mathbb{A}}$ denote the set of all possible pairs in $\mathbb{A}$ and L denote the number of these pairs, i.e., $L = |\mathsf{P}_{\mathbb{A}}| = \dfrac{l(l-1)}{2}$, where l is the number of attributes in T-MATCH. Without loss of generality, we denote $\mathsf{P}_{\mathbb{A}} = \{\mathsf{p}_1, \mathsf{p}_2, ..., \mathsf{p}_L\}$.

- Let E_i denote the event that pair p_i in $\mathsf{P}_{\mathbb{A}}$ does not have the same matching reference as pairs $\{\mathsf{p}_1, \mathsf{p}_2, ..., \mathsf{p}_{i-1}\}$.

We recall that q_2 is the order of $\mathbb{G}_{T_2}$, and that $|q_2| = 512$ bits. Now, the probability of event E is:

$$\begin{aligned} Pr(E) &= \prod_{i=1}^{L} \Pr\,(E_i) = 1\left(1 - \frac{1}{q_2}\right)\left(1 - \frac{2}{q_2}\right)...\left(1 - \frac{L-1}{q_2}\right) \\ &\geq \left(1 - \frac{L-1}{q_2}\right)^L \simeq \left(1 - \frac{2^{2|l|}}{2^{|q_2|}}\right)^L \simeq \left(1 - L\frac{2^{2|l|}}{2^{|q_2|}}\right) \simeq \left(1 - \frac{2^{4|l|}}{2^{|q_2|}}\right) \\ Pr(\bar{E}) &= 1 - Pr(E) \simeq 2^{4|l| - |q_2|} \end{aligned}$$

Since typically $|l| \leq 10$, it follows that the probability that event $\bar{E}$ occurs is negligible.

We conclude that given the security of the hash function H, the probability that an adversary $\mathcal{A}$ breaks the soundness property when tags T_0 and T_1 encode valid attributes is negligible.

Case 2. T_0 or T_1 does not encode valid attributes, i.e., for all $a_p \in \mathbb{A}$, $\mathsf{Dec}_{\mathbb{G}}(c_{\mathsf{T}_0}) \neq \psi(a_p)$ or $\mathsf{Dec}_{\mathbb{G}}(c_{\mathsf{T}_1}) \neq \psi(a_p)$.

Without loss of generality, we assume that for all $a_p \in \mathbb{A}$, $\mathsf{Dec}_{\mathbb{G}}(c_{\mathsf{T}_0}) \neq \psi(a_p)$.

Now, if for all $a_p \in \mathbb{A}$ $\mathsf{Dec}_{\mathbb{G}}(c_{\mathsf{T}_0}) \neq \psi(a_p)$, then this implies that tag T_0 was not issued by issuer I. Consequently, T_0's state $S_{\mathsf{T}_0} = (c_{\mathsf{T}_0}, \sigma_{\mathsf{T}_0})$ was necessarily computed by adversary $\mathcal{A}$. As a result, adversary $\mathcal{A}$ is able to compute the MAC of c_{T_0} without the secret key K. This leads to a contradiction under the security of MAC.

We conclude that given the security of MAC, an adversary $\mathcal{A}$ cannot break the soundness of T-MATCH when tag T_0 or tag T_1 does not encode valid attributes. □

6.6 Privacy Analysis

In the this section, we present T-MATCH's privacy theorems.

6.6.1 Privacy against Readers and the Backend Server

Theorem 6.3. T-MATCH *ensures the privacy of tags against readers R_k and backend server S in the semi-honest model under the subgroup decision assumption.*

Proof sketch. We need to show how to transform any admissible pair $(\mathcal{A}_1, \mathcal{A}_2)$ of adversaries against T-MATCH in the real model, into an admissible pair $(\mathcal{B}_1, \mathcal{B}_2)$ of adversaries in the ideal model.

Backend server S is honest. First, we start with the case of an honest backend server S. In this case, we transform the adversary $\mathcal{A}_1$ (semi-honest reader R_k) against backend server S in the real model into an adversary $\mathcal{B}_1$ (semi-honest reader R_k) against S in the ideal model.

Adversary $\mathcal{B}_1$ will execute $\mathcal{A}_1$ locally, obtaining therefore the messages that $\mathcal{A}_1$ would have sent in a real execution of T-MATCH, and providing $\mathcal{A}_1$ with the messages that he expects to receive from backend server S.

- $\mathcal{A}_1$ reads the states $S_{T_i}^{k_i}$ and $S_{T_j}^{k_j}$ stored into tags T_i and T_j respectively, and computes the bilinear pairing $C_{(i,j)}$ of the ciphertexts stored into T_i and T_j. Then, $\mathcal{A}_1$ sends $C_{(i,j)}$ to $\mathcal{B}_1$ who simulates backend server S.
- $\mathcal{B}_1$ sends the ciphertexts stored into tags T_i and T_j and the secret share α_1 of adversary $\mathcal{A}_1$ to the trusted third party.
- $\mathcal{B}_1$ receives a bit b from the TTP which is the output of the Check function.
- To simulate backend server S to adversary $\mathcal{A}_1$, $\mathcal{B}_1$ computes ν messages M'_p such that:
  1. If $b = 1$: $\mathcal{B}_1$ picks $\nu - 1$ pairs of random numbers (x_p, r_p) in $\mathbb{Z}_N^*$, and computes: $M'_p = (M_{(1,p)}, M_{(2,p)}) = (e(g,g)^{r_p}, e(g,g)^{x_p} e(g,g)^{-\alpha_1 r_p})$, where α_1 is the secret share of $\mathcal{A}_1$. Note that $M_p = M_{(1,p)}^{\alpha_1} M_{(2,p)} = e(g,g)^{x_p}$ is randomly distributed in $\mathbb{G}_T$.

     Next, $\mathcal{B}_1$ selects a random number $r_\nu \in \mathbb{Z}_N$ and computes: $M'_\nu = (M_{(1,\nu)}, M_{(2,\nu)}) = (e(g,g)^{r_\nu}, e(g,g)^{-\alpha_1 r_\nu})$.
  2. If $b = 0$: $\mathcal{B}_1$ picks ν pairs of random numbers (x_p, r_p) in $\mathbb{Z}_N^*$, and computes: $M'_p = (M_{(1,p)}, M_{(2,p)}) = (e(g,g)^{r_p}, e(g,g)^{x_p} e(g,g)^{-\alpha_1 r_p})$.
- Finally, $\mathcal{B}_1$ shuffles M'_p and sends them to adversary $\mathcal{A}_1$.

We show that the distribution of messages M'_p sent to $\mathcal{A}_1$ when $\mathcal{B}_1$ is simulating backend server S is computationally indistinguishable from the distribution of messages M'_p that $\mathcal{A}_1$ actually receives from backend server S in a real execution of T-MATCH.

When adversary $\mathcal{A}_1$ runs T-MATCH in the real model, he expects to receive ν messages M'_p distributed as described below:

- Tags T_i and T_j match: there exists a message $M'_q = (M_{(1,q)}, M_{(2,q)})$ such that $M_q = M_{(1,q)}^{\alpha_1} M_{(2,q)} = 1$, and for all $M'_p \neq M'_q$, the product $M_p = M_{(1,p)}^{\alpha_1} M_{(2,p)}$ is randomly distributed in $\mathbb{G}_{T_2}$.

- Tags T_i and T_j do not match: for all $M'_p = (M_{(1,p)}, M_{(2,p)})$, the product $M_p = M_{(1,p)}^{\alpha_1} M_{(2,p)}$ is randomly distributed in $\mathbb{G}_{T_2}$.

Note that the resulting product $M_p = M_{(1,p)}^{\alpha_1} M_{(2,p)}$ from the message $M'_p = (M_{(1,p)}, M_{(2,p)})$ sent by adversary $\mathcal{B}_1$ during his simulation of backend server S is distributed in $\mathbb{G}_T$ and not in $\mathbb{G}_{T_2}$. However, this cannot be detected by $\mathcal{A}_1$. Otherwise, this implies that $\mathcal{A}_1$ is able to tell whether an element of $\mathbb{G}_T$ is an element of the subgroup $\mathbb{G}_{T_2}$ or not, and this leads to a contradiction under the subgroup decision assumption, see Definition 6.5.

Thus, $\mathcal{B}_1$ successfully simulates backend server S to adversary $\mathcal{A}_1$, and the distribution $\mathsf{Real}_{\bar{\mathcal{A}}}$ is computationally indistinguishable from the distribution $\mathsf{Ideal}_{\bar{\mathcal{B}}}$ when backend server S is honest.

Reader R_k is honest. We transform next an adversary $\mathcal{A}_2$ (semi-honest backend server S) against reader R_k in the real model into an adversary $\mathcal{B}_2$ (semi-honest backend server S) against reader R_k in the ideal model as follows.

- $\mathcal{B}_2$ first eavesdrops on reader R_k to get the states of tags T_i and T_j participating in the matching protocol. Notice that such an attack cannot be prevented as the channel between tags and reader R_k is not secure.

- $\mathcal{B}_2$ simulates reader R_k for adversary $\mathcal{A}_2$ in the real model by computing the bilinear pairing $C_{(i,j)}$ of ciphertexts stored into tags T_i and T_j, and by sending $C_{(i,j)}$ to adversary $\mathcal{A}_2$.

- $\mathcal{B}_2$ sends the set of matching references Ref and the secret share α_2 of adversary $\mathcal{A}_2$ to the TTP.

- The TTP returns a bit b to reader R_k in the ideal model.

  Although adversary $\mathcal{B}_2$ does not have access directly to the value of b, he can still infer its value by observing the behavior of reader R_k in the ideal model. In fact, if $b = 1$, then reader R_k raises an alarm, and so does $\mathcal{B}_2$ in the real model when simulating reader R_k. Otherwise, $\mathcal{B}_2$ does nothing.

To conclude, adversary $\mathcal{B}_2$ successfully simulates reader R_k to adversary $\mathcal{A}_2$ in the real model, and the distributions $\mathsf{Real}_{\bar{\mathcal{A}}}$ and $\mathsf{Ideal}_{\bar{\mathcal{B}}}$ are indistinguishable when reader R_k is honest.

Consequently, T-MATCH ensures the privacy of tags against readers R_k and backend server S in the semi-honest model. □

6.6.2 Privacy against Outsiders

To prove that T-MATCH ensures tag unlinkability, we first show that BGN is IND-CPA under re-encryption.

Let $\mathcal{O}_{\mathsf{REnc}}$ be the oracle that when queried with two BGN ciphertexts c_0 and c_1 encrypted using public key pk, flips a random coin $b \in \{0,1\}$, re-encrypts c_b using pk, and returns the resulting ciphertext c_b'.

Let $\mathcal{A}$ be an adversary that selects two BGN ciphertexts c_0 and c_1 and queries the oracle $\mathcal{O}_{\mathsf{REnc}}$ with c_0 and c_1. $\mathcal{O}_{\mathsf{REnc}}$ randomly chooses $b \in \{0,1\}$, re-encrypts c_b to c_b', and returns c_b' to adversary $\mathcal{A}$, who then outputs his guess b' for bit b.

We say that BGN is IND-CPA under re-encryption, if adversary $\mathcal{A}$ has only a negligible advantage in guessing the correct value of b.

Lemma 6.2. *Boneh-Goh-Nissim is IND-CPA under re-encryption under the subgroup decision assumption in* $\mathbb{G}$.

Proof sketch. Let adversary $\mathcal{B}$ be an adversary against the IND-CPA property of BGN, see Section 2.17.

We show now that if there is an adversary $\mathcal{A}$ who breaks the IND-CPA property under re-encryption of BGN with a non-negligible advantage ϵ, then $\mathcal{B}$ can use $\mathcal{A}$ as a subroutine to break the IND-CPA property of BGN with a non-negligible advantage ϵ'.

Let $\mathcal{O}_{\mathsf{Enc}}$ be the oracle that when queried with two messages m_0 and m_1 in $\mathbb{G}$, flips a random coin $b \in \{0,1\}$, encrypts m_b using BGN and public key pk, and returns the resulting ciphertext c_b.

When adversary $\mathcal{A}$ submits the ciphertexts c_0 and c_1 to $\mathcal{B}$, the latter simulates the oracle $\mathcal{O}_{\mathsf{REnc}}$ as follows.

- He first queries the oracle $\mathcal{O}_{\mathsf{Enc}}$ with messages $m_0 = c_0$ and $m_1 = c_1$.
- The oracle $\mathcal{O}_{\mathsf{Enc}}$ flips a random coin $b \in \{0,1\}$, and encrypts m_b to obtain ciphertext $c_b' = m_b \tilde{g}_1^r = c_b \tilde{g}_1^r$. Note that c_b' is a re-encryption of c_b.
- The oracle $\mathcal{O}_{\mathsf{Enc}}$ returns the ciphertext c_b' to adversary $\mathcal{B}$, who gives it to adversary $\mathcal{A}$.

Adversary $\mathcal{A}$ outputs his guess b' for the bit b. To break the IND-CPA property of BGN, adversary $\mathcal{B}$ outputs the same bit b'.

Since ciphertext c_b' is a re-encryption of c_b and $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the IND-CPA of BGN under re-encryption, it follows that $b' = b$ and that $\mathcal{B}$ is able to break the IND-CPA of BGN with a non-negligible advantage $\epsilon' = \epsilon$, leading to a contradiction under the subgroup decision assumption. □

Theorem 6.4. T-MATCH *ensures tag unlinkability against outsiders under the subgroup decision assumption in* $\mathbb{G}$.

Proof. Assume there is an adversary $\mathcal{A}$ who breaks the tag unlinkability of T-MATCH with a non-negligible advantage ϵ. We show that we can build an adversary $\mathcal{B}$ who uses $\mathcal{A}$ as a subroutine and breaks the IND-CPA property of the BGN cryptosystem under re-encryption with a non-negligible advantage ϵ'.

To break the IND-CPA property of BGN, $\mathcal{B}$ proceeds as follows:

- Adversary $\mathcal{B}$ simulates challenger $\mathcal{C}$ and creates a complete T-MATCH system with l attributes $\mathbb{A} = \{a_1, a_2, ..., a_l\}$, an issuer I, η readers R_k and a backend server S.

  $\mathcal{B}$ selects a random MAC key K, a random secret key x_I, random shares α_1 and $-\alpha_1$, and a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$. Next, he computes the matching references Ref that he is going to use to compute the output of the Check function.

  Then, he provides issuer I with secret keys K, x_I and the hash function H, readers R_k with secret key K and secret share α_1, and backend server S with secret share $-\alpha_1$ and the set of matching references Ref.

  Finally, he simulates issuer I and initializes n tags using as input $\mathbb{A}$, public key pk from the re-encryption oracle $\mathcal{O}_{\mathsf{REnc}}$, hash function H, MAC key K and secret key x_I.

  At the end of tag initialization phase, each tag T_i stores a state $S^0_{T_i} = (c^0_{T_i}, \sigma^0_{T_i}) = (\mathsf{Enc}_{\mathbb{G}}(\psi(a_{T_i})), \mathsf{MAC}_K(c^0_{T_i}))$ such that $a_{T_i} \in \mathbb{A}$.

- $\mathcal{B}$ initializes a database DB where he keeps an entry E_{T_i} for each tag T_i such that: $E_{T_i} = (a_{T_i}, c^0_{T_i}, c^1_{T_i}, ..., c^j_{T_i}, ...)$, where $c^0_{T_i}$ is the ciphertext stored into T_i at initialization, and $c^j_{T_i}$ is the ciphertext stored into tag T_i after the j^{th} interaction of tag T_i with readers R_k in the supply chain.

Learning phase. In the following, we show how adversary $\mathcal{B}$ simulates challenger $\mathcal{C}$ in the learning phase.

- $\mathcal{B}$ simulates oracle $\mathcal{O}_{\mathsf{Tag}}$ and provides $\mathcal{A}$ with a set of r tags of his choice.

- $\mathcal{B}$ simulates the output of the Check function to adversary $\mathcal{A}$. Without loss of generality, we assume that adversary $\mathcal{A}$ submits two tags T_i and T_j to some reader R_k in the supply chain.

  - First, adversary $\mathcal{B}$ reads the states $S^{k_i}_{T_i} = (c^{k_i}_{T_i}, \sigma^{k_i}_{T_i})$ and $S^{k_j}_{T_j} = (c^{k_j}_{T_j}, \sigma^{k_j}_{T_j})$ of tags T_i and T_j respectively, verifies the validity of the MACs $\sigma^{k_i}_{T_i}$ and $\sigma^{k_j}_{T_j}$ and writes into tags T_i and T_j the new states $S^{k_i+1}_{T_i} = (c^{k_i+1}_{T_i}, \sigma^{k_i+1}_{T_i})$ and $S^{k_j+1}_{T_j} = (c^{k_j+1}_{T_j}, \sigma^{k_j+1}_{T_j})$ respectively.
  - Next, he looks up the ciphertexts $c^{k_i}_{T_i}$ and $c^{k_j}_{T_i}$ in his database, retrieves their corresponding attributes a_{T_i} and a_{T_j}, and updates the database entries. Finally, he

checks whether a_{T_i} and a_{T_j} match or not. If so, $\mathcal{B}$ simulates reader R_k in the supply chain and outputs 1. Otherwise, he outputs 0.

It is important to note that the simulation presented above of the Check function works because only issuer I and readers R_k can compute a valid state $S_{T_i}^{k_i} = (c_{T_i}^{k_i}, \sigma_{T_i}^{k_i})$.

Challenge phase. $\mathcal{A}$ submits two challenge tags T_0 and T_1.

$\mathcal{B}$ reads and verifies the states stored into T_0 and T_1, and retrieves the corresponding ciphertexts c_0 and c_1 respectively.

- To simulate $\mathcal{O}_{\mathsf{Flip}}$, $\mathcal{B}$ queries the oracle $\mathcal{O}_{\mathsf{REnc}}$ with ciphertexts c_0 and c_1. $\mathcal{O}_{\mathsf{REnc}}$ returns a re-encryption c'_b of ciphertext c_b, $b \in \{0,1\}$.
- Then, $\mathcal{B}$ computes $\sigma'_b = \mathsf{MAC}_K(c'_b)$ and stores the state $S_{\mathsf{T}_b} = (c'_b, \sigma'_b)$ into tag T_b.
- Finally, $\mathcal{B}$ returns tag T_b to $\mathcal{A}$.

$\mathcal{A}$ outputs his guess b' for the bit b.

Now, to break the IND-CPA property of BGN under re-encryption, $\mathcal{B}$ outputs b'.

Notice that if $\mathcal{A}$ outputs $b' = 1$, then tag T_b corresponds to tag T_1, and therewith c'_b is a re-encryption of c_1. Otherwise, tag T_b corresponds to tag T_0 and c'_b is a re-encryption of c_0.

Since adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the tag unlinkability of T-Match, it follows that $\mathcal{B}$ will have a non-negligible advantage $\epsilon' = \epsilon$ in breaking the IND-CPA property of BGN under re-encryption. This leads to a contradiction under the subgroup decision assumption in $\mathbb{G}$. □

6.7 Evaluation

T-Match targets storage only tags that do no feature any computational capabilities. A tag in T-Match is required to store a BGN ciphertext in $\mathbb{G}$ ($|\mathbb{G}| = 1024$ bits) and a MAC of size 160 bits, totaling a storage of 1184 bits.

We believe that T-Match can be deployed using current ISO 18000-3 HF tags, such as UPM RFID HF RaceTrack tags (156) that feature up to 8 Kbits of memory.

In each execution of T-Match, reader R_k reads two tags T_i and T_j and updates their states as follows: it re-encrypts the BGN ciphertexts c_{T_i} and c_{T_j} of tags T_i and T_j respectively, then it computes the MAC of the re-encrypted ciphertexts. This amounts to computing two exponentiations in $\mathbb{G}$ and two keyed hash functions.

To evaluate the Check function, reader R_k computes a bilinear pairing $C_{(i,j)} = e(c_{T_i}, c_{T_j}) \in \mathbb{G}_T$ such that $|\mathbb{G}_T| = 2048$ bits. Then, reader R_k initiates a two round protocol for plaintext equality test with backend server S by sending the ciphertext $C_{(i,j)}$.

Upon receiving ciphertext $C_{(i,j)}$, backend server S performs 2ν exponentiations in $\mathbb{G}_T$, where ν is the number of matching references in Ref, and obtains ν messages M'_p. Next, backend server S shuffles the messages M'_p and sends them to reader R_k.

Finally, when reader R_k receives the messages M'_p, it performs ν exponentiations in $\mathbb{G}_T$ and outputs the outcome of the Check function.

Table 6.1: Evaluation of memory and computation in T-MATCH

	Tag	Reader R_k	Backend server S
Memory	1184 bits	pk, α_1, K	pk, α_2, Ref
Exponentiation in $\mathbb{G}_T$ $\lvert\mathbb{G}_T\rvert = 2048$ bits	–	ν	2ν
Exponentiation in $\mathbb{G}$ $\lvert\mathbb{G}\rvert = 1024$ bits	–	2	–
MAC	–	2	–
Bilinear pairing	–	1	–
Shuffle	–	–	1

6.8 Related Work

T-MATCH shows similarities to secret handshake and secret matching protocols. Nevertheless, traditional solutions for secure and privacy preserving secret matching between two parties as proposed by Ateniese et al. (4), Balfanz et al. (9) cannot be implemented in cheap RFID tags. These solutions require the computation of bilinear pairings which cannot be performed by current RFID tags.

Boneh et al. (26) propose a protocol that allows the public evaluation of 2-DNF formula on boolean variables by relying on the BGN encryption. The protocol proposed in (26) can be slightly modified to implement tag matching. However in this case, tags are required to store $O(l)$ ciphertexts of size 1024 bits where l is the number of attributes in the system – rendering such an approach unrealistic.

Another approach to evaluate the Check function is attribute based encryption see Goyal et al. (74), Pirretti et al. (132), Sahai and Waters (140). The idea is to associate each attribute a_i in the system with some secret component of some private key sk. When two tags T_i and T_j that match come together, the secret key sk can be reconstructed. The reconstruction of a correct secret key sk enables reader R_k to decrypt some ciphertext C for which it knows the underlying plaintext M. The matching is verified by checking whether $\mathsf{Dec}_{\mathsf{sk}}(C) = M$ or not. Though, the use of attribute based encryption can allow reader R_k to evaluate the Check function by itself without a backend server S, it requires either cryptographic operations on tags or their synchronization.

6.9 Summary

RFID tag based matching is required by many real-world supply-chain applications. Matching however, raises new security and privacy concerns. T-MATCH tackles these challenges and provides secure and privacy preserving item matching suited for resource restricted tags that are unable to perform any computation. T-MATCH evaluates, in a privacy preserving manner, a function Check that on the input of two tags T_i and T_j outputs a bit b indicating whether T_i and T_j match or not. T-MATCH is provably secure and privacy preserving under standard assumptions: security of MAC and hash functions, and the subgroup decision assumption. Finally, designed for storage only tags, T-MATCH requires tags to store only 150 bytes.

7

Conclusion and Future Work

7.1 Summary

Although the proliferation of RFID tags is admitted to be financially beneficial, the deployment of RFID technologies still comes with a variety of privacy and security threats that range from denial of service to industrial espionage. While well established cryptographic solutions can always remedy most of these threats in theory, they remain too expensive in practice for the constrained devices that are RFID tags. The dilemma of ensuring tag security and privacy while keeping the computational requirements in tags minimal has given rise to a plethora of work on RFID authentication and the corresponding security and privacy definitions. However, the task of designing secure and privacy preserving authentication protocols that meet the computational constraints of RFID technology was shown to be difficult if not impossible. Actually, existing formalizations of RFID privacy assume a strong adversary against which privacy cannot be achieved without sacrificing RFID scalability and cost effectiveness. Therefore, in this thesis we first focused on bridging this gap between the theoretical formalization of RFID privacy and the practical aspects of RFID technology by assuming an adversary who cannot continuously monitor RFID tags: *there is at least one interaction between tags and readers that is unobserved by the adversary.* Then, we designed four multi-party protocols that provide secure and privacy preserving solutions for RFID-enabled supply chains. More precisely, we targeted the following applications:

7.1.1 Tag Ownership Transfer

In Chapter 4, we presented ROTIV to tackle the privacy and the security issues of tag ownership transfer in the supply chain. The core idea of ROTIV is to store in each tag in the supply chain a symmetric key and an Elgamal encryption of a *short signature* computed by some trusted issuer. The encrypted signature allows owners to identify tags in constant time and to verify the identity of their issuer, whereas the symmetric key is used to mutually authenticate tags and owners. In this manner, ROTIV assures:

- Constant-time mutual authentication between tags and readers, while tags are only required to compute a hash function.

- Issuer verification without trusted third party: Every supply chain partner can verify whether the tags he owns originate from a trusted party or not.

- Provable security: A successful ownership transfer of some tag T implies that T is a legitimate tag that was issued by a trusted party, and that the owner of tag T participated in the protocol execution.

- Provable privacy: A new owner of tag T cannot link T to its past interactions, and a previous owner cannot trace T's future protocol executions.

While most of the privacy and the security properties of ROTIV were proven in the standard model, the security of ROTIV's short signature[8] together with the security of ROTIV's issuer verification were demonstrated in the random oracle model. As discussed in Section 2.2.1.1, security in the random oracle validates the overall design of the protocol, yet the security of the scheme in the real world depends heavily on the implementation choices of the underlying hash function. We recall that in Chapter 4, we have suggested the use of the hashing algorithm proposed by Brier et al. (28) which hashes into ordinary curves and was shown to be *indifferentiable* from a random oracle.

7.1.2 Product Tracking

In Chapter 5, we introduced Tracker and Checker that aim at verifying the genuineness of products by checking the validity of the paths they took in the supply chain. Both protocols build upon an original combination of polynomial-based path encoding and signatures to enable each reader in the supply chain to update the states of tags, while supply chain verifiers check the genuineness of tags by reading the tags' states.

Both Tracker and Checker check the genuineness of products by verifying their paths in the supply chain, still, they target different application scenarios: Tracker focuses on the problem of product traceability by a trusted party, whereas Checker aims at solving the issue of on-site checking by allowing each partner in the supply chain to verify the genuineness of products that are present in his site.

We summarize the contributions of Chapter 5 as follows:

- Efficient and compact path encoding that does not depend on the length of the path; each path is encoded as an element of the finite field $\mathbb{F}_q$, $|q| = 160$ bits.

[8] To the best of our knowledge, there is no short signature that is secure in the standard model and relies on standard assumptions.

- Both protocols can be implemented in storage only tags that do not perform any computation. A tag T is only required to store an encrypted state that is updated and re-encrypted at each protocol execution by readers in the supply chain.
- Comprehensive privacy and security definitions that capture the requirements of product tracking applications.
- Provable security: Readers in the supply chain cannot forge valid path signatures.
- Provable privacy: A supply chain partner cannot trace or link tags that are not present in his site.

Since TRACKER and CHECKER rely on storage only tags, it follows that both protocols are vulnerable to DoS attacks: an adversary can always invalidate the state of a tag by writing "garbage" into the tag. Such an attack cannot be countered unless tags are able to execute some sort of reader authentication, increasing thus the computational requirements on the tags and therewith their prices. This shows that there is a tradeoff between DoS resistance and the financial cost of tracking applications.

Also, the security of TRACKER and CHECKER was proven in the random oracle model. Similar to ROTIV, we suggested the use of the hashing algorithm presented in (28), which we believe to be sufficient in the context of this thesis.

7.1.3 Item Matching

In Chapter 6, we presented T-MATCH which is a protocol for RFID-based item matching in the supply chain that aims at the automation of safety inspection when transporting or storing hazardous goods such as chemicals. The goal of T-MATCH is to allow each reader in the supply chain to detect the presence of two dangerously reactive chemicals in its vicinity. T-MATCH relies on techniques of secure two-party computation to enable a reader and a backend server to compute jointly on the input of two tags T_i and T_j the outcome of a boolean function $\mathsf{Check}(T_i, T_j)$. $\mathsf{Check}(T_i, T_j) = 1$ if T_i and T_j match (i.e., they are attached to barrels that contain dangerously reactive chemicals), and 0 otherwise.

The contributions of T-MATCH are:

- T-MATCH targets storage only tags that do not perform any computation. A tag T_i in T-MATCH is only required to store a state that is updated at every protocol execution by readers in the supply chain.
- Original definitions capturing the security and the privacy requirements of RFID-based item matching in the supply chain.
- T-MATCH is provably privacy preserving: T-MATCH relies on techniques of secure two-party computation to ensure that neither readers nor backend server can disclose the private content of a tag.

- T-MATCH is provably secure: Readers raise an alarm only when they interact with a pair of matching tags.

Even though tags in T-MATCH do not perform any computation, the readers and the backend server have to execute $O(\nu)$ computations, where ν is the number of matching references (i.e., the number of pairs of chemicals that are dangerously reactive). The design of a privacy preserving RFID-based item matching protocol whose running time do not depend on the number of matching references is far from being straightforward. We believe however, that in the real world, the number of matching references will not be large enough to be computationally prohibitive for the readers and the backend server.

Another limitation of T-MATCH is that the item matching can only be performed pairwise. That is, in the presence of n barrels of chemicals, a reader has to call the protocol $\frac{n(n-1)}{2}$ times to decide whether there are dangerously reactive chemicals in its vicinity or not, which could be time consuming.

Finally, the cost effectiveness of storage only tags comes at the expense of resistance to DoS attacks. As explained in Sections 6.3.1.1 and 7.1.2, such attacks cannot be thwarted unless tags are able to authenticate the readers updating their states.

7.2 Future Work

Here, we give an overview of possible research directions that could be investigated as a natural continuation of the results presented in this manuscript.

- The **formal definitions of tag privacy** throughout this thesis were indistinguishability-based. A direction of future work could consist of redefining privacy using simulator-based definitions where information leakage is quantified by the ability of an adversary to distinguish between a real execution of the protocol and a simulated one, in accordance with the work of Vaudenay (159) and Paise and Vaudenay (129).

- **Privacy preserving RFID-based grouping proofs:** A grouping proof is a protocol that convinces a verifier (usually a reader) that a group of tags were read (almost) simultaneously. Such protocols are useful in industries such as automotive and aeronautics, as they can be employed to prove that the different components of some product were assembled (almost) at the same time. Most existing protocols (32, 88, 141) rely on hash functions and timestamps to assure that tags were read/updated simultaneously. We argue however that such protocols can be implemented using storage only tags and without timestamps. The idea would be to replace timestamps by random numbers and secret sharing techniques, in a way that enables the readers in the supply chain to verify that **1.)** tags belong to the same group and that **2.)** they were read simultaneously.

- **Physical tag identification :** While physical tag identification could be used to detect cloned products and to verify the genuineness of identification documents, it could also be viewed as an efficient technique to violate privacy. Fortunately, physical fingerprinting (identification) of RFID tags is still prone to errors in dynamic environment with high tag mobility (165), thus limiting the scope of tracking attacks using physical approaches. Still, it is very important to investigate the feasibility and the cost of accurate physical-layer identification, and to propose appropriate counter-measures to reduce its impact on tag privacy.

- **Efficient distance bounding protocols in RFID tags:** Recently a new class of attacks were brought to the attention of researchers, whereby malicious parties have the protocol between the tag and the reader run over distances much larger than the nominal range of the tag using some communication relay. Such attacks could be prevented if tags were equipped with efficient mechanisms to estimate the distance separating them from readers. The design of such "distance bounding" mechanisms in the context of resource-constrained tags raises very challenging research questions.

Appendix A

Resistance to Forgery of Matching References

In the following, we demonstrate that it is computationally infeasible for a backend server S to forge a new matching reference from its set Ref of matching references.

Theorem A.1. T-MATCH *is resistant to forgery of matching references under the BCDH assumption and the subgroup decision assumption in the random oracle model.*

Proof sketch. Assume that there is an adversary $\mathcal{A}$ (backend server S) who breaks the resistance to forgery of matching references with a non-negligible advantage ϵ. We show that there is an adversary $\mathcal{B}$ who uses backend server S to break the BCDH assumption in $\mathbb{G}$ with a non-negligible advantage ϵ'.

Let $\mathcal{O}_{\mathsf{BCDH}}$ be an oracle that selects randomly $x, y, z \in \mathbb{Z}_N$, and returns $g, g^x, g^y, g^z \in \mathbb{G}$.

- Adversary $\mathcal{B}$ first queries the oracle $\mathcal{O}_{\mathsf{BCDH}}$ that returns $g, g^x, g^y, g^z \in \mathbb{G}$.
- Then, adversary $\mathcal{B}$ simulates a complete T-MATCH system with l attributes $\mathbb{A} = \{a_1, a_2, ..., a_l\}$, an issuer I and η readers R_k.
  1. He provides issuer I with g^z from the BCDH challenge instead of the secret key x_I;
  2. he supplies readers R_k with a secret MAC key K, a BGN public key and the secret share α_1;
  3. he provides backend server S with the secret share α_2 and the set of matching references Ref that are computed as described below.
- We note that the goal of adversary $\mathcal{B}$ is to convince adversary $\mathcal{A}$ that there are two attributes a_1 and a_2 such that:

$$H(a_1) = g^{xr_1} \text{ and } H(a_2) = g^{yr_2}$$

where r_1 and r_2 are random elements of $\mathbb{Z}_N^*$.

To this end, adversary $\mathcal{B}$ simulates a random oracle $\mathcal{H}$ to compute the hash function H.

Simulation of the random oracle $\mathcal{H}$. To respond to the queries of the random oracle $\mathcal{H}$, adversary $\mathcal{B}$ keeps a table T_H of tuples $(a_j, r_j, H(a_j))$:

On a query a_i, $3 \leq i \leq l$, $\mathcal{B}$ replies as follows:

1.) If there is a tuple $(a_i, r_i, H(a_i))$ that corresponds to a_i, then $\mathcal{B}$ returns $H(a_i)$.

2.) If a_i has never been queried before, then adversary $\mathcal{B}$ picks a random number $r_i \in \mathbb{Z}_N^*$, and answers with $H(a_i) = g^{r_i}$. Finally, adversary $\mathcal{B}$ stores the tuple $(a_i, r_i, H(a_i))$ in table T_H.

On a query $H(a_1)$ ($H(a_2)$ resp.), $\mathcal{B}$ picks a random number $r_1 \in \mathbb{Z}_N^*$ ($r_2 \in \mathbb{Z}_N^*$ resp.) and replies with $H(a_1) = g^{xr_1}$ ($H(a_2) = g^{yr_2}$ resp.).

- Now adversary $\mathcal{A}$ computes the set of matching references Ref by first selecting ν pairs of attributes $\{a_i, a_j\} \subset \mathbb{A}$ such that $\{i, j\} \neq \{1, 2\}$ and computing their corresponding matching reference $\mathsf{ref}_{(i,j)}$.

Computation of the matching references. On input of a pair of attributes $\{a_1, a_i\}$, adversary $\mathcal{B}$ first retrieves the tuples $(a_1, r_1, H(a_1) = g^{xr_1})$ and $(a_i, r_i, H(a_i) = g^{r_i})$, then computes:

$$\begin{aligned} \mathsf{ref}_{(1,i)} &= e(g^x, g^z)^{r_1 r_i} = e(g^{xr_1}, g^{r_i})^z \\ &= e(H(a_1), H(a_i))^z \in \mathbb{G}_T \end{aligned}$$

Similarly, adversary $\mathcal{B}$ computes the matching reference of a pair of attributes $\{a_2, a_i\}$.

$$\begin{aligned} \mathsf{ref}_{(2,i)} &= e(g^y, g^z)^{r_2 r_i} = e(g^{yr_2}, g^{r_i})^z \\ &= e(H(a_2), H(a_i))^z \in \mathbb{G}_T \end{aligned}$$

On input of a pair of attributes $\{a_i, a_j\}$ such that $i \neq 1, 2$ and $j \neq 1, 2$, adversary $\mathcal{B}$ first retrieves the tuples $(a_i, r_i, H(a_i) = g^{r_i})$ and $(a_j, r_j, H(a_j) = g^{r_j})$, then computes the corresponding matching reference:

$$\begin{aligned} \mathsf{ref}_{(i,j)} &= e(g, g^z)^{r_i r_j} = e(g^{r_i}, g^{r_j})^z \\ &= e(H(a_i), H(a_j))^z \in \mathbb{G}_T \end{aligned}$$

We recall that according to T-MATCH, the matching reference of two attributes a_i and a_j is computed as:

$$\begin{aligned} \mathsf{ref}_{(i,j)} &= e(\psi(a_i), \psi(a_j)) = e(H(a_i)^{x_I}, H(a_j)^{x_I}) \\ &= e(H(a_i), H(a_j))^{x_I^2} \in \mathbb{G}_{T_2} \end{aligned}$$

It follows that the secret key x_I of issuer I looks as if it fulfills the equation: $x_I^2 = z$[10].

- Finally, adversary $\mathcal{B}$ supplies adversary $\mathcal{A}$ with the set of matching references Ref.

  Note that $\mathsf{ref}_{(i,j)} \in \mathbb{G}_T$ instead of $\mathbb{G}_{T_2}$, nonetheless, $\mathcal{A}$ cannot detect this thanks to the subgroup decision assumption in $\mathbb{G}_T$.

- Now, adversary $\mathcal{A}$ outputs a new matching reference $\mathsf{ref} \notin \mathsf{Ref}$, such that there is $(a_i, a_j) \in \mathbb{A} \times \mathbb{A}$ for which the following equation holds:

  $$\mathsf{ref} = e(H(a_i), H(a_j))^z$$

  Note that if $(a_i, a_j) = (a_1, a_2)$, then $\mathsf{ref} = e(H(a_i), H(a_j))^z = e(g^{xr_1}, g^{yr_2})^z$ and $e(g,g)^{xyz} = \mathsf{ref}^{\frac{1}{r_1 r_2}}$.

  Let p denote the probability that adversary $\mathcal{A}$ computes the matching reference of attributes $\{a_1, a_2\}$.

  Accordingly, if adversary $\mathcal{A}$ has a non-negligible advantage ϵ in breaking the resistance to the forgery of matching references, then adversary $\mathcal{A}$ can break the BCDH problem with a non-negligible advantage $\epsilon' = p\epsilon$, leading to a contradiction.

□

[10] Adversary $\mathcal{A}$ cannot tell whether z is a quadratic residue or not.

Appendix B

Résumé

L'identification par radiofréquence communément connue sous le nom de RFID est une technologie d'auto-identification comme les codes à barre, la biométrie, les cartes à puce ... etc. Un tag RFID est un dispositif sans fil équipé d'un identifiant unique de 96 bits, qui contrairement aux codes à barre permet l'identification des objets sans intervention humaine.

Dans un premier temps, la technologie RFID était envisagée pour remplacer les codes à barres, dans le but d'automatiser les processus de collection de données et de traitement d'information concernant les produits dans la chaîne d'approvisionnement. Or aujourd'hui, la technologie RFID est déjà intégrée dans les passeports biométriques et dans les applications de contrôle d'accès.

Ce qui rend la technologie RFID intéressante est son coût qui est relativement faible. Un tag RFID peut être vendu pour 0.15 U.S.$ sans remise sur le volume. Bien que le prix actuel des tags RFID soit encore prohibitif pour les applications de chaînes d'approvisionnement, on s'attend que le coût d'un tag baisse pour atteindre des niveaux commercialement viables permettant une adoption de la technologie RFID à grande échelle, non seulement dans les chaînes d'approvisionnement mais aussi dans d'autres applications.

Néanmoins, la rentabilité des tags RFID a un prix qui est la vie privée des parties ayant des tags RFID et aussi la vie privée des partenaires dans la chaîne d'approvisionnement. C'est très important de noter que la technologie RFID n'est pas conçue pour protéger la vie privée de ses utilisateurs, en effet, le but original de cette technologie est de permettre l'identification et le suivi des objets dans la chaîne d'approvisionnement. De ce fait, les tags RFID envoient ses identifiants chaque fois interrogés par un lecteur RFID sans le consentement de ses propriétaires. Cela implique que les attaques sur la vie privée telles que le suivi des personnes et l'espionnage industriel peuvent être montées facilement par la simple interrogation des tags RFID.

Pour répondre à ces problèmes liés à la protection de la vie privée, deux approches ont vu le jour. La première repose sur des mesures physiques pour limiter la portée de ces attaques, par exemple: des cages de Faraday sont utilisées pour empêcher la lecture non-autorisée des

passeports RFID. La deuxième approche qui nous intéresse dans ce manuscrit vise à protéger la vie privée des tags RFID en se basant sur des solutions cryptographiques.

La conception de protocoles cryptographiques pour RFID s'est avérée une tâche difficile pour deux raisons principales: D'abord, maintenir le coût faible de RFID est d'une importance primordiale pour favoriser un déploiement à grande échelle des tags RFID. Par conséquent, toute solution cryptographique pour RFID doit correspondre aux ressources strictes des tags. Deuxièmement, il est crucial de concevoir des protocoles efficaces pour les chaînes d'approvisionnement pour ne pas ralentir les performances de ces dernières.

Ces défis soulevés par les approches cryptographiques utilisées pour protéger la vie privée des systèmes RFID ont stimulé un travail de recherche très actif qui portait principalement sur la conception de protocoles d'authentification qui préservent la vie privée et qui conviennent aux capacités de calcul des tags RFID. L'objectif de ces protocoles est de permettre aux lecteurs RFID *légitimes* d'authentifier et d'identifier les tags RFID, alors que les lecteurs non-légitimes ne doivent pas être en mesure d'identifier un tag en écoutant ses communications ou en l'interrogeant. Les protocoles d'authentification pour les systèmes RFID proposés dans la littérature peuvent être classés en trois catégories, comme suit:

- **Authentification légère:** Elle se repose sur des opérations binaires comme "et", "xor" (18, 66, 91). Pourtant efficace, ce type de protocoles est sujet aux attaques de récupération de clés (14, 64, 128).

- **Authentification symétrique:** Les protocoles dans cette catégorie utilisent les primitives cryptographiques symétriques (48, 50, 58, 122, 153) qui peuvent être mises en oeuvre dans les tags RFID. Cependant, Damgård and Pedersen (42) ont montré qu'il y a un compromis entre la sécurité de protocoles d'authentification symétriques et leur évolutivité. En effet, pour assurer la sécurité et la protection de la vie privée des tags RFID, un protocole symétrique doit s'exécuter dans un temps linéaire dans le nombre de tags.

- **Authentification asymétrique:** Contrairement à l'authentification symétrique, les solutions basées sur les techniques asymétriques (103, 113, 126) offrent la possibilité de concevoir de protocoles d'authentification qui s'exécutent dans un temps constant et qui protègent en même temps la vie privée des tags.

La diversité et l'hétérogénéité des protocoles d'authentification pour les systèmes RFID ont suscité un intérêt dans la formalisation des définitions de protection de la vie privée (5, 92, 129, 159) adaptées au contexte RFID. Ces définitions visent **premièrement** à capturer les capacités d'un adversaire contre les tags RFID, **deuxièmement** à mesurer l'information qu'un adversaire peut apprendre en écoutant le canal sans fil entre les tags et les lecteurs RFID. Ces définitions formelles ont ouvert la voie à une analyse plus approfondie des protocoles existants. Cette analyse nous a permis d'identifier ce qui peut être réellement atteint

en termes de sécurité et protection de la vie privée des tags RFID.

En effet, il était démontré que la plupart des protocoles actuels n'arrivent pas à protéger la vie privée des tags contre un adversaire actif qui *écoute toutes les communications des tags*: Vaudenay (159) a montré le résultat intuitif qui stipule que *la protection de la vie privée ne peut pas être assurée contre un tel adversaire*. Aussi, un résultat plus positif indique que pour assurer la protection de la vie privée contre un adversaire plus faible, les tags doivent implémenter les protocoles d'accord des clés, ce qui impose l'utilisation de la cryptographie à clé publique dans les tags. Or, la cryptographie à clé publique est impraticable dans des puces aussi contraintes que les tags RFID. De ce fait, on a conclu que les protocoles cryptographiques pour les systèmes RFID **1.)** peuvent au mieux se baser sur des fonctions symétriques, et que **2.)** les formalisations de la protection de la vie privée doivent être relâchées afin de combler le fossé entre ce qui est souhaitable et ce qui est effectivement réalisable dans un environnement RFID.

Pour ces raisons là, dans ce manuscrit on a visé à:

- Formaliser des définitions appropriées de protection de la vie privée et de sécurité qui tiennent en compte les limitations des puces aussi contraintes que les tags RFID et les actions potentielles qu'un adversaire peut effectuer pour compromettre la sécurité et la vie privée des tags. D'ailleurs, on rappelle que les capacités de calcul restreintes des tags RFID ne permettent pas l'implémentation de fonctions asymétriques dans les tags.

- Proposer des solutions cryptographiques pour les applications de chaîne d'approvisionnement qui prennent en considération les limitations des tags RFID en termes de capacité de calcul et qui comptent améliorer la collaboration entre les partenaires de la chaîne d'approvisionnement. En particulier, on s'est intéressé à trois applications: le transfert de propriété des tags, la vérification d'authenticité, et l'appariement des objets dans la chaîne d'approvisionnement.

  Il est important de noter que les solutions cryptographiques pour les applications de chaînes d'approvisionnement doivent être financièrement rentables et efficaces pour assurer leurs déploiements à grande échelle.

Dans cette optique, on a considéré dans cette thèse un modèle de protection de la vie privée où un adversaire peut modifier l'état interne des tags, par contre il ne peut pas écouter toutes leurs communications.

Par ailleurs, on estime que l'hypothèse ci-dessus est *réaliste* dans le contexte des chaînes d'approvisionnement pour deux raisons: **1.)** Les tags RFID n'implémentent pas des mécanismes de protection physique. Cela signifie que n'importe quel adversaire ayant accès aux tags peut facilement lire et parfois réécrire leurs contenus. **2.)** Les tags RFID dans la chaîne d'approvisionnement changent régulièrement d'endroit, il est donc difficile pour un adversaire d'observer toutes leurs interactions.

Sous cette hypothèse, on a d'abord formalisé des définitions de sécurité et de protection de la vie privée qui correspondent bien aux exigences des chaînes d'approvisionnement. Ensuite, on a proposé des protocoles cryptographiques *multipartites* pour les applications de chaînes d'approvisionnement, dont certains peuvent être implémentés dans les tags sans capacité de calcul, voir II.

B.1 Sécurité et la Vie Privée des Systèmes RFID

La RFID est une technologie qui permet l'identification et la traçabilité des objets sans ligne de vue directe ou intervention humaine. Un tag RFID est un dispositif sans fil à faible coût qui étiquette l'objet auquel il est attaché en ayant un identifiant unique et non-réutilisable. L'identifiant unique du tag agit comme un pointeur vers une entrée de base de données contenant toute l'historique de l'objet étiqueté. En conséquence, la technologie RFID a été envisagée pour remplacer les codes à barres dans la chaîne d'approvisionnement, car il favorise une identification rapide et automatisée du produit, ainsi que la possibilité d'enregistrer et de tracer l'historique des produits étiquetés dans la chaîne d'approvisionnement.

Or, la prolifération des tags RFID vient avec de nouvelles menaces pour la sécurité et la vie privée des entreprises/individus qui possèdent les tags. Ces menaces potentielles ont donné naissance à un domaine de recherche très actif qui à la fois vise la formalisation des modèles de sécurité et de la vie privée, et la conception des protocoles d'authentification qui protègent la vie privée des tags RFID. Les principaux défis à relever dans ce domaine de recherche sont la définition de modèles formels qui décrivent globalement les capacités d'un adversaire contre les systèmes RFID dans le monde réel, et la conception des protocoles d'authentification **1.)** qui assurent la sécurité et la confidentialité des données des tags, et **2.)** qui peuvent être implémentés dans les tags RFID.

B.1.1 Systèmes RFID

Un système RFID comprend plus de composantes que les tags RFID déjà mentionnés, en effet, il se compose de:

- Tags;
- lecteurs;
- système backend.

Les tags et les lecteurs communiquent sur un canal sans fil non-sécurisé, alors que le canal entre les lecteurs et le système back-end est généralement sécurisé.

B.1.1.1 Tags RFID

Un tag RFID comprend une micro puce qui abrite une mémoire, des fonctionnalités logiques limitées, et une antenne. Les tags peuvent être classés en fonction de leurs fréquences. Tags de haute fréquence HF fonctionnent à $13,56$MHz et leur portée de lecture maximale est de 1m. Tags de ultra-haute fréquence UHF opèrent entre 858 et 930 MHz et la portée moyenne de lecture est de 3m. Les tags UHF représentent la technologie dominante pour les applications de chaînes d'approvisionnement, tandis que les tags HF sont plus appropriés pour les applications à proximité comme par exemple la billetterie électronique. Outre la fréquence, les tags peuvent être classés en fonctions de leurs modes d'alimentation (135). Un tag *passif* est un tag qui n'est équipé d'aucune batterie, et s'appuie donc sur des mécanismes de backscattering pour répondre aux requêtes envoyées par les lecteurs à proximité. Un tag *actif* par contre a sa propre batterie et peut initier la communication avec les lecteurs. Un tag *semi-actif* est un tag hybride qui possède sa propre alimentation mais n'initie jamais la communication avec les lecteurs.

Il suit que les tags passifs sont beaucoup moins chers que les tags actifs, et par conséquent, ils sont plus appropriés à remplacer les codes à barres dans les chaînes d'approvisionnement. Les avantages des tags passifs sont bien évidemment leur faible coût, leur petite taille et leur durée de vie qui n'est pas limitée par la durée de vie de la batterie. Cependant, les tags passifs ont peu de ressources et peu de capacités de calcul, ce qui transforme la conception des applications basées sur les tags RFID à un véritable défi.

Donc, dans cette thèse on s'est concentré uniquement sur les tags passifs.

B.1.1.2 Lecteurs RFID et Systèmes Backend

Les lecteurs RFID sont des émetteurs-récepteurs qui sont capables de communiquer avec les tags RFID sur un canal radiofréquence. Un lecteur peut être en mesure de lire ou d'écrire le contenu des tags. Il est généralement doté d'une antenne, d'un microprocesseur, d'une source d'alimentation électrique, et éventuellement d'une interface qui lui permet de transmettre les données reçues des tags au système backend.

Le système backend comprend habituellement une base de données qui recueille les informations transmises par les lecteurs pour des fins diverses qui dépendent de l'application pour laquelle la technologie RFID est utilisée.

Il existe deux catégories de lecteurs (59):

- Lecteurs fixes: Les lecteurs sont placés dans un endroit fixe et ils sont toujours connectés à un réseau qui les lie avec le système backend. Par exemple: dans les applications de contrôle d'accès où les lecteurs sont situés à l'entrée d'une zone sécurisée.
- Lecteurs mobiles: Les lecteurs peuvent être portables et ils ne sont pas obligés à communiquer en permanence avec le système backend. Ils sont principalement employés pour interroger les prix des produits dans un supermarché ou pour l' inventaire.

B.1.2 Applications RFID

La technologie RFID peut être intégrée dans plusieurs applications qui varient selon le but d'identification. Parmi les applications de la technologie RFID, on trouve le paiement automatisé, le contrôle d'accès, et la gestion des chaînes d'approvisionnement.

L' une des applications éminentes des tags RFID est la gestion des chaînes d'approvisionnement, dans laquelle les tags stockent en plus de leurs identifiants uniques, des informations supplémentaires qui sont utilisées pour automatiser et réguler les processus de production et de distribution dans la chaîne d'approvisionnement, tout en minimisant les erreurs dues à l'intervention humaine. En attachant des tags RFID aux produits circulant dans la chaîne d'approvisionnement, le manager de la chaîne peut automatiquement identifier les contrefaçons, les goulots d'étranglement de production, la pénurie de stocks et l'origine des produits défectueux. Ce type d'applications est d'une grande valeur ajoutée, car elle réduit le temps et les erreurs lors de la gestion des produits, tout en diminuant le nombre de personnes impliquées dans la chaîne d'approvisionnement.

Parfois, un tag doit non seulement s'identifier, mais aussi prouver qu'il est légitime en s'authentifiant. Une telle fonctionnalité est nécessaire dans certaines applications telles que le paiement automatique, la détection des contrefaçons, le contrôle d'accès... etc.

Un autre domaine d'applications des tags RFID est la traçabilité des objets étiquetés. Étant donné que les lecteurs sont placés à des endroits fixes et connus, un objet étiqueté peut être facilement localisé avec une certaine précision. Une telle application peut être utilisée afin de détecter la présence de certains produits dans une usine ou un entrepôt, ou de localiser des personnes à l'intérieur d'un bâtiment.

Par ailleurs, les partisans de la technologie RFID croient que la prolifération potentielle des tags RFID débouchera sur des applications qui peuvent assister les gens dans leurs tâches quotidiennes. Une de ces applications est les "maisons intelligentes" avec des appareils intelligents tels que les machines à laver qui sélectionnent automatiquement les cycles de lavage appropriées pour ne pas endommager les habits délicats, ou des réfrigérateurs qui détectent l'expiration de produits alimentaires (89). Dans le même esprit, la technologie RFID pourrait être utilisée pour faciliter la navigation des personnes âgées dans la maison ou pour vérifier si un patient se conforme à ses prise de médicament ou non (89).

B.1.3 Menaces de Sécurité et de la Vie Privée

Dans cette section, on discute quelques menaces contre la sécurité et la vie privée liées au déploiement de la technologie RFID.

B.1.3.1 Menaces de Sécurité

La technologie RFID fait face à des menaces de sécurité diverses telles que le déni de service, les attaques de relais, et le clonage.

- *Déni de service:* Une telle attaque est exécutée en envoyant des signaux dans la même bande de fréquence que les lecteurs légitimes pour causer un brouillage électromagnétique qui empêche les tags légitimes de communiquer avec les lecteurs légitimes.

- *Attaques de relai:* Ces attaques sont mises en oeuvre en plaçant un dispositif entre un tag RFID et un lecteur. Ce dispositif relaie les informations échangées entre les deux parties légitimes qui croient qu'ils sont physiquement proches l'un de l'autre.

- *Clonage:* Cette attaque est réalisée en écoutant les communications des tags pour récupérer leurs identifiants uniques, puis en écrivant ces identifiants dans de nouveaux tags reprogrammables. Le clonage pourrait être utilisé pour remplacer le contenu des tags attachés à des produits chers avec le contenu des tags attachés aux produits moins chers dans un super marché par exemple.

En vue de protéger les systèmes RFID contre les attaques décrites ci-dessus, Karygiannis et al. (95) suggèrent certaines contre-mesures de sécurité qui peuvent être implémentées. Par exemple, le clonage peut être réduit par les protocoles d'authentification entre le tag et le lecteur. Toutefois, la rareté des ressources de calcul dans les tags RFID rend la conception de protocoles d'authentification résistant aux différentes attaques très difficile.

D'ailleurs, les protocoles de distance bounding (8, 27, 77, 99) ont été proposées pour protéger contre les attaques de relai. L'idée derrière ces protocoles est tout simplement d'estimer la distance physique séparant les lecteurs et les tags lors d'une communication.

Finalement, les attaques de brouillage électromagnétique peuvent être empêchées en augmentant la sécurité physique près des lecteurs RFID à travers des gardes, des barrières, des caméras, et des murs blindés pour bloquer les signaux électromagnétiques externes, dans le but de limiter les interférences radio qu'elles soient accidentelles ou malveillantes (95).

B.1.3.2 Menaces de la Vie Privée

Vu que les tags RFID répondent à toutes les requêtes sans le consentement de leurs propriétaires ou détenteurs, la prolifération de la technologie RFID fait apparaître de nouveaux risques qui peuvent entrainer des violations de la vie privée des tags et de leurs détenteurs, telles que l'espionnage industriel, le profilage des consommateurs et la traçabilité des individus.

- *L'espionnage industriel:* En écoutant les communication des objets étiquetés présents sur la chaîne d'approvisionnement, une entreprise peut recueillir des informations confidentielles et sensibles sur les processus opérationnels internes d'un concurrent industriel. Ces informations pourraient servir à déterminer les programmes de distribution, le taux quotidien de production, la disponibilité ou la rupture de stock, ainsi que l'identité des fournisseurs et des partenaires.

- *Le profilage des consommateurs:* Toute personne portant un objet étiquetés par un tag RFID est sujette à l'inventaire clandestin. Un caissier dans un magasin peut facilement apprendre les produits qui intéressent un certain client en lisant les tags attachés aux produits achetés par ce client.

- *Traçabilité:* Les interactions d'un tag peut être facilement tracées, vu que les tags RFID envoient leurs identifiants uniques en clair chaque fois interrogés.

Dans ce qui suit, on décrit brièvement certaines approches qui ont été proposées pour limiter la portée de ces menaces contre la vie privée des tags RFID.

- Désactivation des tags: Les tags RFID peuvent être désactivés en utilisant une commande "KILL" envoyée par les lecteurs. Quand un tag reçoit la commande KILL, il devient hors service. Maintenant, pour éviter le déni de service via la désactivation des tags, la commande KILL est protégée par un code PIN connu seulement par les lecteurs autorisés. Même si la désactivation définitive des tags est une mesure très efficace pour protéger la vie privée des individus, cette technique empêche l'implémentation de tout service après-vente qui se base sur la technologie RFID.

- Proxying: Cette approche vise à protéger la vie privée des tags en utilisant des dispositifs qui agissent comme des pare-feux RFID (94, 136). Ces dispositifs transmettent aux tags RFID uniquement les requêtes qui répondent aux politiques de sécurité définies au préalable par les détenteurs des tags.

- Blocage: Cette approche protège la vie privée des tags en s'appuyant sur des mesures physiques. Par exemple, une cage de Faraday peut être utilisée pour protéger contre la lecture non autorisée. Il est également possible d'empêcher une lecture non autorisée via les tags bloqueurs (93). Un tag bloqueur exploite les propriétés de protocoles d'anticollision pour interrompre toute interaction entre les tags et les lecteurs non-autorisés.

- Pseudonymes: Au lieu d'avoir un identifiant unique permanent, les tags utilisent des pseudonymes qui changent au fil du temps pour éviter les attaques de traçabilité (voir (82)). Un lecteur est donc tenu de régulièrement réécrire les pseudonymes (i.e., les identifiants) des tags qu'il est en train de lire, en gardant tout de même un historique des anciens pseudonymes.

- Rechiffrement: Alors que le chiffrement de l'identifiant d'un tag protège la confidentialité de ce tag, il n'empêche pas la traçabilité de ce dernier. En effet, à chaque requête, le tag envoie le chiffrement de son identifiant. Or, ce chiffrement peut servir de nouvel identifiant permettant la traçabilité du tag. Pour remédier à cette limitation, Ateniese et al. (3), Golle et al. (73), Juels and Pappu (90) suggèrent l'utilisation des techniques

de rechiffrement. Un tag dans ce cas là stocke un chiffrement IND-CPA (cf. définition 2.17) de son identifiant. Maintenant, quand un lecteur lit le chiffrement c stocké dans un tag T, il rechiffre c pour obtenir un nouveau chiffrement c' que le lecteur stocke dans tag T. Ainsi, l'adversaire ne peut pas tracer un tag sur une longue période de temps. Il faut noter que grâce à la propriété IND-CPA, c et c' chiffre le même identifiant.

- Authentification protégeant la vie privée: Ce type d'authentification permet aux tags de s'identifier auprès des lecteurs légitimes dans un système RFID d'une façon qui préserve la vie privée. C'est à dire qu'après une authentification d'un tag T auprès d'un lecteur légitime R, un adversaire peut seulement apprendre si l'authentification a réussi ou non, alors que R peut bien authentifier et identifier T.

La plupart des travaux antérieurs sur la sécurité et la vie privée des systèmes RFID a été axée sur:

- Les protocoles d'authentification qui protègent la vie privée des tags et qui conviennent aux capacités limitées de calcul des tags RFID. Ces protocoles vont de protocoles d'authentification légères qui reposent uniquement sur des opérations binaires (18, 66, 91), aux protocoles d'authentification symétriques (48, 50, 58, 122, 153), en arrivant aux protocoles d'authentification à clé publique (103, 113, 126).

- Les modèles formels de sécurité et de la vie privée qui décrivent d'une manière complète et compréhensive les attaques possibles contre les systèmes RFID (5, 92, 129, 159).

B.1.4 Limitations de la Sécurité et de la Vie Privée des Systèmes RFID

La plupart des protocoles proposés dans la littérature pour les systèmes RFID visent à protéger la vie privée des tags au niveau applicatif, cependant Avoine and Oechslin (7) ont souligné que la non-traçabilité des tags ne peut être jamais assurée en s'appuyant uniquement sur les protocoles cryptographiques. À savoir, un protocole cryptographique protégeant la vie privée des tags au niveau applicatif n'empêche pas un adversaire de tracer les tags via leurs caractéristiques et propriétés physiques. Par exemple, Danev et al. (43) et Zanetti et al. (164) ont exploité les caractéristiques spectrales des réponses émises par les tags pour extraire des empreintes physiques qui permettent l'identification correcte des tags individuels du même fabricant et du même modèle. De ce fait, les auteurs ont proposé d'utiliser ces empreintes physiques pour détecter les produits clonés dans la chaîne d'approvisionnement, et pour vérifier l'authenticité des documents d'identité qui intègrent les tags RFID. Il est manifeste qu'une identification précise des tags RFID par les empreintes physiques compromet la vie privée des tags indépendamment des contre-mesures *"cryptographiques"* proposées pour protéger la vie privée au niveau applicatif. Néanmoins, une identification précise des tags RFID qui s'appuie sur les empreintes physiques nécessite un environnement contrôlé où

les tags sont à proximité et à une position fixe par rapport au lecteur (43), ce qui n'est pas toujours le cas surtout dans un contexte où l'adversaire vise à tracer un tag/individu. Ainsi, la conception des protocoles cryptographiques pour les systèmes RFID demeure une solution viable qui peut protéger relativement la vie privée des tags.

Pourtant, protéger la vie privée des tags RFID au niveau applicatif s'est avéré être une tâche très difficile. Le problème réside dans le fait que les formalisations existantes de la vie privée des tags supposent généralement *un adversaire fort contre lequel la vie privée ne peut jamais être assurée en respectant les limitations en termes de calcul et de puissance des tags RFID*. Ce qui nous amène à conclure que la conception de protocoles RFID préservant la vie privée appelle à la formalisation d'un modèle plus faible mais réaliste qui capte les capacités d'un adversaire du monde réel et qui répond aux capacités limitées de la technologie RFID.

Dans ce manuscrit donc, on a considéré d'abord un adversaire qui peut interagir et modifier le contenu des tags, mais qui ne peut pas surveiller la totalité de leurs interactions. Cette hypothèse peut également être formulée comme suit: *il y a au moins une exécution du protocole entre les tags RFID dans le système et les lecteurs légitimes qui n'est pas observée par l'adversaire.* Ceci est en fait compatible avec le travail de Ateniese et al. (3), Dimitriou (51), Lim and Kwon (111) et Sadeghi et al. (139). On croit que cette hypothèse est réaliste vu la nature mobile des tags RFID.

Ensuite, on s'est adressé aux protocoles multipartites qui impliquent plusieurs lecteurs, étendant ainsi notre recherche au-delà des simples protocoles d'authentification entre tag et lecteur pour mettre en oeuvre des applications pour la chaîne d'approvisionnement qui assurent la protection de la vie privée des tags, cf. II.

B.2 Protocoles Cryptographiques pour les Chaînes d'Approvisionnement Équipées de Tags RFID

Une chaîne d'approvisionnement se définit comme un réseau de *partenaires*, qui peuvent comprendre des distributeurs, des transporteurs, des fournisseurs qui tous participent à la production, la vente, et finalement la livraison d'un produit donné (1). Tandis que la gestion de la chaîne d'approvisionnement est définie comme étant *la gestion et le contrôle de tous les matériaux et de toute information pendant tous les processus de production et de distribution (i.e., de l'acquisition des matières premières jusqu'à la livraison du produit aux utilisateurs finaux)* (115). Ainsi, la gestion de la chaîne d'approvisionnement vise principalement à retracer les mouvements de produits pour éviter les bottlenecks de production, réduire les pertes de produits et améliorer la réactivité de la chaîne d'approvisionnement aux rappels de produits.

Toutefois, lorsque les produits ne sont équipés que de codes à barres optiques, la tâche la plus simple comme l'inventaire demandera beaucoup de main d'oeuvre et deviendra par la suite sujette aux erreurs humaines. Par conséquent, les distributeurs comme Wal-Mart et le

US DoD (115, 152) ont approuvé l'adoption de la technologie RFID au niveau des palettes pour améliorer les performances de la chaîne d'approvisionnement. L'avantage principal de la technologie RFID est la possibilité d'identifier les produits individuels sans ligne de vue directe. Cette propriété permet aux partenaires de la chaîne d'approvisionnement de suivre les différents produits et de tracer leurs historiques de façon opportune sans intervention humaine. En conséquence, il est admis que l'utilisation de tags RFID dans la chaîne d'approvisionnement est d'une valeur commerciale importante car elle augmente la visibilité de la chaîne, ce qui favorise la régulation du taux de production, la détection de la contrefaçon, la mise en application des règles de sécurité... etc.

Pourtant, l'omniprésence de la technologie RFID facilite le déni de service et l'espionnage industriel, comme expliqué dans la Section 3.1.4. Bien que le déni de service puisse être adressé par une augmentation de la sécurité physique près de tags RFID, les problèmes liés à la vie privée des tags sont plus difficiles à traiter. En effet, la vie privée des tags ne doit pas seulement être assurée contre les intrus, mais aussi contre les partenaires de la chaîne d'approvisionnement. Autrement dit, un partenaire de la chaîne d'approvisionnement ne doit pas être capable de suivre les tags RFID qui ne sont pas sur son site. Cette contrainte là appelle à des solutions innovantes qui s'appuient sur la cryptographie, tout en tenant en compte les ressources limitées des tags. À savoir, une solution cryptographique pour les applications de chaîne d'approvisionnement doit être **1.)** *efficace*, afin de ne pas influencer les performances globales de la chaîne d'approvisionnement, et **2.)** *réalisable* dans les tags passifs (idéalement, les tags sans capacité de calcul), afin de ne pas imposer un coût supplémentaire à la chaîne d'approvisionnement.

Maintenant, pour concevoir des applications de chaînes d'approvisionnement qui sont à la fois *pas chers*, *efficaces* et protègent *la vie privée*, on a relâché les modèles formels de la vie privée des systèmes RFID, en supposant qu' *un adversaire ne peut pas surveiller en permanence les tags dans la chaîne d'approvisionnement*, comme discuté ci-dessus et dans la Section 3.4.

En supposant un tel adversaire, on était en mesure de concevoir **1.)** un protocole de transfert de propriété qui s'exécute dans un temps constant alors que les tags ne calculent que des fonctions de hachage (cf. Chapitre 4), **2.)** deux protocoles qui se basent sur des tags sans capacité de calcul et qui s'attaquent au problème de la vérification d'authenticité des produits circulant dans la chaîne d'approvisionnement (voir Chapitre 5), et enfin **3.)** un protocole d'appariement d'objets qui met en application les règles de sécurité dans la chaîne d'approvisionnement en utilisant uniquement des tags sans capacité de calcul (cf, Chapitre 6).

B.2.1 Transfert de Propriété avec Vérification d'Authenticité

Tant qu' un produit/tag circule dans la chaîne d'approvisionnement entre de différents partenaires, sa propriété éventuellement sera transférée. Dans ce contexte, la propriété d'un tag

est la capacité qui permet à un partenaire dans la chaîne d'approvisionnement d'authentifier le tag et de transférer la propriété de ce dernier. D'autre part, le transfert de propriété correspond à l'action de transmettre les informations nécessaires pour authentifier et identifier un tag d'un partenaire à l'autre.

En vue de protéger la sécurité et la vie privée des tags et des partenaires dans la chaîne d'approvisionnement, un protocole de transfert de propriété des tags RFID doit s'assurer des points suivants:

- *Authentification mutuelle sûre* entre les tags RFID et leurs propriétaires (i.e., les partenaires de la chaîne d'approvisionnement).
- *Propriété exclusive:* Les parties non-autorisées ne doivent pas être en mesure de transférer la propriété d'un tag donné sans le *consentement* de son propriétaire.
- *Backward unlinkability:* Un ancien propriétaire d'un tag RFID ne doit pas être capable de tracer un tag une fois la propriété de ce dernier est transférée.
- *Forward unlinkability:* Le nouveau propriétaire d'un tag RFID ne doit pas être capable de relier un tag à ses *interactions antérieures*.

En outre, les protocoles de transfert de propriété des tags RFID sont tenus d'être efficaces pour ne pas ralentir les performances globales de la chaîne d'approvisionnement. Ainsi, un protocole de transfert de propriété des tags RFID doit reposer sur un protocole d'authentification efficace qui prend en compte les ressources de calcul limitées des tags RFID: Comme indiqué précédemment, on présume que les tags RFID peuvent au mieux mettre en oeuvre des primitives symétriques telles que les fonctions de hachage. On rappelle cependant que les protocoles d'authentification symétriques déjà proposés dans la littérature sont conçus pour protéger la vie privée contre un *adversaire* fort qui peut *en permanence* intercepter les communications des tags, et par la suite, ils nécessitent une recherche linéaire dans le nombre de tags dans la chaîne d'approvisionnement pour authentifier un tag. De ce fait, il faut relâcher les modèles de la vie privée pour concevoir des protocoles d'authentification efficaces, en supposant qu'il y ait au moins *une interaction entre un tag donné et son propriétaire qui n'est pas interceptée par l'adversaire*.

Pour répondre aux exigences de la vie privée et de sécurité décrites ci-dessus, on a introduit ROTIV (voir Chapitre 4), qui en sus de fonctions de base liées au transfert de propriété offre une nouvelle fonctionnalité qui est *la vérification d'authenticité*. Autrement dit, un partenaire dans la chaîne d'approvisionnement peut vérifier l'origine des tags RFID dont il est propriétaire. Cette fonctionnalité empêche les partenaires malveillants d'injecter des produits de contrefaçon dans la chaîne d'approvisionnement.

L'idée principale de ROTIV est de stocker dans chaque tag dans la chaîne d'approvisionnement une clé symétrique et un chiffrement Elgamal de son identifiant qui est signé par un

émetteur de confiance. Le chiffrement à clé publique permet au propriétaire d'identifier les tags en temps constant, tandis que la clé symétrique permet l'authentification mutuelle des tags et de leurs propriétaires. En plus, chaque tag dans ROTIV est associé à un ensemble de références de propriété qui permettent à un propriétaire d'un tag T d'authentifier et de transférer la propriéte du tag T. Après chaque authentification mutuelle réussie du tag T, son état et ses références de propriété sont mis à jour afin d'assurer à la fois sa sécurité et sa vie privée. Finalement, le contrôle d'authenticité d'un tag T est exécuté en vérifiant si la signature chiffrée stockée dans T est une signature valide par l'émetteur de confiance ou non.

B.2.1.1 Aperçu de ROTIV

Dans ROTIV, un tag T stocke l'état $S_T^j = (K_T^j, c_T^j)$, où K_T^j est une clé partagée entre le tag T et son propriétaire, et c_T^j est un chiffrement Elgamal de l'identifiant du tag T signé par l'émetteur de confiance I.

Quand un propriétaire $O_{(T,k)}$ démarre une authentification mutuelle avec un tag T, le tag répond avec le chiffrement c_T^j et un MAC calculé en utilisant la clé secrète K_T^j. A la réception de la réponse du tag T, le propriétaire $O_{(T,k)}$ utilise sa clé secrète pour déchiffrer c_T^j, et vérifie par la suite si le texte en clair résultant du déchiffrement de c_T^j correspond à une entrée dans sa base de données DB_k. Si c'est le cas, $O_{(T,k)}$ vérifie le MAC envoyé par T en utilisant la clé secrète K_T^j stockée dans sa base de données. Ainsi, ROTIV permet l'authentification mutuelle en temps constant, alors que les tags ne calculent que des primitives symétriques (i.e., MAC).

Pour assurer la backward et la forward unlinkability, les tags sont tenus à mettre à jour leurs états après chaque authentification mutuelle réussie, en utilisant des mécanismes de mise à jour de clés symétriques et des techniques de rechiffrement.

Maintenant, pour transférer la propriété d'un tag T, le propriétaire $O_{(T,k)}$ du tag T fournit les références de propriété correspondant au tag T au futur propriétaire $O_{(T,k+1)}$. Ces références permettent au propriétaire $O_{(T,k+1)}$ de contrôler d'abord l'authenticité (l'origine) du tag T en vérifiant que c_T^j chiffre une signature valide de l'identifiant du tag T par l'émetteur I, puis de s'authentifier à T et mettre à jour le chiffrement c_T^j.

B.2.1.2 Contributions

En résumé, les contributions de ROTIV sont les suivantes:

- Authentification mutuelle en temps constant alors que les tags ne calculent que des fonctions de hachage.

- Vérification d'authenticité qui permet aux propriétaires potentiels d'un tag T de vérifier l'identité de son émetteur.

- Contrairement aux travaux antérieurs (60, 101, 117, 142), le transfert de propriété dans ROTIV ne nécessite pas une tierce partie de confiance.
- Formalisations des exigences de sécurité et de la vie privée liées au transfert de propriété dans les chaînes d'approvisionnement.
- Preuves formelles de la sécurité et de la protection de la vie privée de ROTIV.

B.2.2 Vérification d'Authenticité de Produits dans la Chaîne d'Approvisionnement

La traçabilité des produits est l'une des applications majeures des chaînes d'approvisionnement équipées de tags RFID, car elle permet de vérifier l'authenticité de produits en temps réel et sans intervention humaine (56, 83, 118, 151, 160). L'idée est de tracer le chemin que les produits ont pris dans la chaîne d'approvisionnement en lisant leurs tags RFID: Si un tag a pris un chemin valide dans la chaîne d'approvisionnement, on peut en déduire que ce tag est un tag légitime. Toutefois, l'utilisation de tags RFID pour la vérification d'authenticité vient avec de nouvelles menaces pour la sécurité et la vie privée des tags et des partenaires dans la chaîne d'approvisionnement.

En ce qui concerne la sécurité, il doit être vérifiable si un produit est authentique en lisant le tag attaché au produit. À cette fin, la chaîne d'approvisionnement possède un ensemble de vérificateurs qui vérifient le chemin que les tags prennent dans la chaîne d'approvisionnement. Entre temps, les lecteurs au long de la chaîne d'approvisionnement mettent à jour les états internes des tags qui sont à leur proximité. Maintenant, le défi principal est de permettre aux lecteurs de mettre à jour les états des tags tout en les empêchant d'injecter des produits contrefaits.

Le deuxième défi concerne la protection de la vie privée des tags. En règle générale, les partenaires de la chaîne d'approvisionnement ne veulent divulguer aucune information sur leurs processus internes ou sur leurs relations stratégiques ni à leurs concurrents ou à leurs clients. Alors, un adversaire dans la chaîne d'approvisionnement ne doit pas être en mesure de retrouver ou reconnaître les tags qu'il a lus auparavant.

Il est aussi important de noter que les solutions répondant à ces exigences de sécurité et de la vie privée doivent être légères en termes de calcul côté tags pour permettre un déploiement à grande échelle. Idéalement, elles devraient être réalisables dans les tags RFID les moins chers, à savoir, les tags sans capacité de calcul. Par conséquent, tout calcul cryptographique requis par le protocole doit être effectué par les lecteurs. En plus, la vérification d'authenticité (i.e., vérification des chemins empruntés par les tags) par les lecteurs ne doit pas être chère en termes de calcul pour éviter toute surcharge des lecteurs, et donc toute entrave des performances de la chaîne d'approvisionnement.

Dans cette optique, on a présenté deux protocoles appelés Tracker et Checker (cf. Chapitre 5) qui permettent la vérification d'authenticité de produits dans la chaîne d'appro-

visionnement d'une façon sûre et respectant la vie privée des tags. L'idée principale derrière ces deux protocoles est d'encoder les chemins dans la chaîne d'approvisionnement par des polynômes, puis employer l'encodage obtenu pour signer les identifiants des tags. TRACKER cible le scénario de la traçabilité des produits où la vérification d'authenticité est effectuée par une *partie de confiance* appelée *manager*, alors que CHECKER aborde le problème du contrôle-sur-site qui permet à chaque lecteur dans la chaîne d'approvisionnement d'agir comme *vérificateur* qui peut parfois être "malveillant".

B.2.2.1 Aperçu de Tracker

TRACKER repose sur une partie de confiance appelée manager M pour vérifier l'authenticité des produits/tags dans la chaîne d'approvisionnement. En utilisant les notations de la section 5.2, cela signifie que $\mathcal{V} = \{M\}$. On rappelle que le contrôle d'authenticité des tags est effectuée en vérifiant la séquence des étapes dans la chaîne d'approvisionnement que les tags ont visitées. D'où un tag T dans TRACKER stocke un état interne S_T^j encodant le chemin que le tag T a pris dans la chaîne d'approvisionnement. L'idée qui sous-tend TRACKER est d'encoder les différents chemins dans la chaîne d'approvisionnement par des *polynômes*. Plus précisément, un chemin $\mathcal{P}$ dans la chaîne d'approvisionnement est représenté par l'évaluation d'un polynôme $Q_{\mathcal{P}} \in \mathbb{F}_q[X]$ en un point fixe x_0, offrant ainsi un codage compact et efficace des chemins.

Maintenant, TRACKER s'appuie sur la propriété que pour deux chemins différents $\mathcal{P} \neq \mathcal{P}'$, valide ou non, l'équation $Q_{\mathcal{P}}(x_0) = Q_{\mathcal{P}'}(x_0)$ ne tient qu'avec une probabilité négligeable quand q est assez grand et x_0 est un générateur de $\mathbb{F}_q^*$. Donc, deux chemins différents produiront deux valeurs différentes, et ainsi deux encodages différents. Par conséquent, l'état d'un tag T va être uniquement associé à un seul chemin (valide) dans la chaîne d'approvisionnement.

Toutefois, la représentation du chemin telle que présentée ci-dessus n'empêche pas le clonage des chemins: En effet, un adversaire peut copier le chemin d'un tag valide dans un tag contrefait et injecter ce dernier dans la chaîne d'approvisionnement. Pour résoudre ce problème, les tags dans TRACKER stockent une signature $\sigma_{\mathcal{P}}(\mathsf{ID})$ de leurs chemins définie comme $\sigma_{\mathcal{P}}(\mathsf{ID}) = H(\mathsf{ID})^{Q_{\mathcal{P}}(x_0)}$ au lieu de $Q_{\mathcal{P}}(x_0)$, où H est une fonction de hachage cryptographique. La signature de chemin correspond donc à l'identifiant du tag signé par l'encodage polynomiale du chemin pris par le tag. Par construction, les signatures de chemin valides prouvent que les tags sont émis par une autorité légitime, et qu'ils ont emprunté des chemins valides dans la chaîne d'approvisionnement.

TRACKER peut être structuré en trois parties: **1.)** L'émetteur I écrit un état initial S_T^0 dans un nouveau tag T. **2.)** Les lecteurs R_k au long de la chaîne d'approvisionnement mettent à jour la signature de chemin stockée dans T en appliquant des opérations arithmétiques simples représentées par une fonction de mise à jour notée f_{R_k} de l'état actuel S_T^j du tag T (cf. équation 5.2). Cela se traduit à la fin par l'évaluation de $\sigma_{\mathcal{P}_{\text{valid}_i}} = H(\mathsf{ID})^{Q_{\mathcal{P}_{\text{valid}_i}}(x_0)}$. **3.)** Enfin, le manager M vérifie si l'état S_T^j stocké dans T correspond à l'un des chemins valides

dans la chaîne d'approvisionnement. Si c'est le cas, le manager M accepte le tag T comme tag légitime et identifie le chemin valide que T a pris dans la chaîne d'approvisionnement.

Sécurité et vie privée de TRACKER***.*** D'une part, pour protéger la vie privée des tags dans TRACKER, chaque tag stocke un chiffrement IND-CPA (plus précisément, un chiffrement Elgamal sur les courbes elliptiques) de son état $S_T = (\mathsf{ID}, H(\mathsf{ID}), \sigma_{\mathcal{P}}(\mathsf{ID}))$, alors que les lecteurs utilisent les propriétés homomorphiques d'Elgamal pour mettre à jour les signatures de chemin stockées dans les tags sans déchiffrement. À la fin de la chaîne d'approvisionnement, le manager M déchiffre et vérifie la validité des chemins et par la suite la validité des tags.

D'autre part, la sécurité de TRACKER repose sur la difficulté du problème CDH (cf. définition 2.26). En fait, on montre que s'il existe un adversaire qui est capable de calculer un chiffrement Elgamal d'un état valide $S_T = (\mathsf{ID}, H(\mathsf{ID}), \sigma_{\mathcal{P}_{\mathrm{valid}_i}}(\mathsf{ID}))$, alors ce même adversaire sera en mesure de résoudre le problème CDH.

B.2.2.2 Aperçu de Checker

Bien que TRACKER permette une vérification efficace, sûre et protégeant la vie privée des tags dans la chaîne d'approvisionnement, cette solution souffre de deux inconvénients majeurs. **1.)** Elle nécessite une *partie de confiance* qui est le manager pour vérifier les chemins des tags. **2.)** La vérification ne peut être effectuée qu'une fois les tags arrivent au manager, mais pas avant. Cela limite le déploiement à grande échelle d'une telle solution, surtout dans un contexte où les partenaires demandent d'être en mesure de vérifier l'authenticité des produits en temps réel et sur leurs "sites".

Par conséquent, dans cette thèse, on a proposé une deuxième solution pour la traçabilité de produits dans la chaîne d'approvisionnement appelée CHECKER. En effet, CHECKER s'adresse au problème de vérification d'authenticité sur site en permettant à chaque lecteur R_k dans la chaîne d'approvisionnement de vérifier la validité des chemins empruntés par les tags, au lieu d'une vérification effectuée par une partie de confiance et qui n'a lieu qu'à la fin de la chaîne d'approvisionnement. En utilisant les notations de la section 5.2, ceci correspond à un système de vérification d'authenticité, où chaque étape de la chaîne d'approvisionnement est un check point, et chaque lecteur dans la chaîne d'approvisionnement est un vérificateur.

De ce fait, un tag T dans CHECKER passant par un chemin valide $\mathcal{P}_{\mathrm{valid}_i}$ stocke un état chiffré $S_T^j = (\mathsf{Enc}(\mathsf{ID}), \mathsf{Enc}(\sigma_{\mathcal{P}_{\mathrm{valid}_i}}(\mathsf{ID})))$, où ID est l'identifiant de T et $\sigma_{\mathcal{P}_{\mathrm{valid}_i}}(\mathsf{ID})$ est la signature de chemin définie comme suit: $\sigma_{\mathcal{P}_{\mathrm{valid}_i}}(\mathsf{ID}) = H(\mathsf{ID})^{\phi(\mathcal{P}_{\mathrm{valid}_i})}$.

Lors de l'initialisation, l'émetteur I écrit dans un tag T un état initial chiffré $S_T^0 = (\mathsf{Enc}_{\mathsf{pk}_1}(\mathsf{ID}), \mathsf{Enc}_{\mathsf{pk}_1}(\sigma_{\mathsf{v}_0}(\mathsf{ID})))$, où pk_1 est la clé publique de la prochaine étape du tag T dans la chaîne d'approvisionnement.

Sans perte de généralité, on suppose que chaque fois le tag T visite un lecteur R_k, celui-ci lit l'état chiffré S_T^j stocké dans T et le déchiffre en utilisant sa propre clé secrète sk_k pour

obtenir le pair $(\mathsf{ID}, \sigma_{\mathcal{P}}(\mathsf{ID}))$. Ensuite, lecteur R_k vérifie si T est passé par un chemin valide dans la chaîne d'approvisionnement menant à R_k ou non. Après la vérification du chemin, lecteur R_k calcule la fonction f_{R_k} pour mettre à jour l'état stocké dans T comme décrit dans l'équation 5.2. Enfin, il chiffre le nouvel état du tag T en utilisant la clé publique de l'étape suivante que le tag T va visiter.

***Sécurité et vie privée de* Checker.** La sécurité de Checker est assurée par l'utilisation d'une signature qui utilise le codage polynomial des chemins dans la chaîne d'approvisionnement comme clé secrète. La différence entre Checker et Tracker réside dans le fait que Checker emploie les groupes bilinéaires, qui permettent de calculer la clé de vérification comme une clé publique. Cette propriété offre aux lecteurs dans Checker la possibilité de vérifier l'authenticité des produits en utilisant des signatures qui sont relativement courtes sans mettre en péril la sécurité de la chaîne d'approvisionnement. En fait, on montre qu'un adversaire ne peut pas forger un état valide, sinon il sera capable de casser BCDH (cf. définition 2.32).

Pour protéger la vie privée des tags contre les lecteurs dans la chaîne d'approvisionnement, les tags stockent un chiffrement IND-CCA de leurs états. Vu que Checker utilise les sous-groupes de courbes elliptiques qui acceptent la construction des couplages bilinéaires, on note que tout chiffrement IND-CCA qui prend place dans les groupes où le problème de DDH est difficile peut être utilisé pour chiffrer les états internes des tags. Pour faciliter la présentation, on utilise le shéma de Cramer-Shoup (41). En plus, chaque lecteur R_k dans la chaîne d'approvisionnement est doté d'une paire de clés publique et secrète $(\mathsf{sk}_k, \mathsf{pk}_k)$.

B.2.2.3 Contributions

Les contributions majeures de Tracker et Checker sont les suivantes:

- Ils permettent de déterminer le chemin exact que chaque tag a emprunté dans la chaîne d'approvisionnement.
- Ils garantissent la vie privée et la sécurité des tags dans la chaîne d'approvisionnement.
- Contrairement aux travaux précédents sur la vérification d'authenticité des produits dans la chaîne d'approvisionnement tels que Ouafi and Vaudenay (127) ou Li and Ding (110), nos protocoles peuvent être implémentés dans des tags sans capacité de calcul.

B.2.3 Appariement de Produits dans la Chaîne d'Approvisionnement

L'une des applications importantes de la technologie RFID est l'automatisation des contrôles de sécurité lors du transport de marchandises dangereuses – telles que les produits chimiques hautement réactifs – dans les chaînes d'approvisionnement. En effet, il est dangereux de placer certains produits chimiques proches les uns des autres, parce que même les petites

fuites de ces produits peuvent entraîner une vraie menace pour la vie des travailleurs dans la chaîne d'approvisionnement.

Certaines solutions récemment proposées visant à assurer les règles de sécurité lors du transport de produits chimiques dans les chaînes d'approvisionnement équipent chaque produit chimique d'un tag RFID qui stocke des informations qui l'identifient (voir Cobis (40)). Avant que deux produits chimiques soient placés côte à côte, leurs tags sont scannés par un lecteur RFID, et chaque tag envoie son contenu en clair au lecteur. Le lecteur de son côté envoie les données lues à un serveur qui effectue l'appariement des produits en se basant sur un ensemble des références d'appariement noté Ref qui identifient les pairs de produits chimiques réactifs. Maintenant, lorsque deux produits chimiques réactifs sont détectés, le serveur déclenche une alarme.

Cependant, la solution présentée ci-dessus souffre de plusieurs incovénients qui entraînent des menaces à la sécurité et la vie privée. Le fait que les tags transmettent leurs contenus en clair permet à toute partie malveillante écoutant le canal entre tags et lecteur de déduire des informations sur les produits chimiques réactifs circulant dans la chaîne d'approvisionnement et de déterminer leurs emplacements. Il suit donc que les protocoles d'appariement à base de tag RFID nécessitent une conception minutieuse qui assure la sécurité et la vie privée des tags RFID dans la chaîne d'approvisionnement.

Un protocole d'appariement des tags RFID protégeant la vie privée doit s'assurer que l'appariement est effectué sans divulguer le contenu des tags. Autrement dit, la seule information révélée après l'exécution du protocole aux lecteurs de la chaîne d'approvisionnement est un bit b, tel que $b = 1$ si les tags sont attachés à des produits réactifs, sinon $b = 0$. Idéalement aussi, un adversaire ne doit pas être en mesure de faire la distinction entre les tags en écoutant le canal sans fil entre les tags et les lecteurs.

En ce qui concerne la sécurité, il est obligatoire de s'assurer qu'un protocole d'appariement est correct (presque) tout le temps. Plus précisément, il est nécessaire de détecter tous les éléments incompatibles (produits chimiques réactifs). Cela correspond à la complétude du protocole: Le protocole doit toujours déclencher une alarme lorsque deux produits chimiques réactifs sont mis l'un à côté de l'autre. En outre, le protocole doit être efficace: Une alarme se déclenche uniquement quand c'est nécessaire, lorsqu'un appariement est détecté par le protocole, on peut conclure que les tags impliqués dans le protocole sont des produits chimiques réactifs. Cette deuxième contrainte correspond à la validité du protocole.

Notez que les solutions adressant les problèmes de sécurité et de la vie privée décrits ci-dessus sont fortement contraintes par les capacités de calcul restreintes des tags RFID. Alors que la vie privée des tags contre les intrus peut être assurée en utilisant des techniques de rechiffrement, la protection de la vie privée des tags contre les lecteurs dans la chaîne d'approvisionnement est beaucoup plus difficile à assurer, en particulier, lors de l'utilisation des tags RFID sans capacité de calcul. En effet, les solutions traditionnelles qui garantissent la sécurité et la vie privée dans les protocoles d'appariement (cf. Ateniese et al. (4), Balfanz

et al. (9)) sont en général basées sur les protocoles de poignées de mains secrètes entre deux parties, qui peuvent pas être mis en oeuvre dans un environnement RFID.

Ainsi, on conçoit T-MATCH (voir Chapitre 6), un nouveau protocole pour l'appariement de tags impliquant deux tags T_i et T_j attachés à deux produits chimiques circulant dans la chaîne d'approvisionnement, plusieurs lecteurs R_k et un serveur backend S. T-MATCH cible les tags sans capacité de calcul afin de permettre son déploiement à un coût raisonnable.

B.2.3.1 Aperçu de T-MATCH

Dans T-MATCH, un lecteur R_k dans la chaîne d'approvisionnement lit le contenu d'un pair de tags T_i et T_j, coopère avec le serveur backend S pour effectuer l'appariement des deux tags, et délivre finalement le résultat d'appariement tout en assurant la vie privée des tags T_i et T_j face à des lecteurs R_k curieux et un serveur backend S curieux.

Chaque lecteur R_k dans la chaîne d'approvisionnement est tenu d'évaluer une fonction booléenne Check pour tout pair de tags T_i et T_j en coopérant avec le serveur backend, telle que Check envoie $b = 1$, si T_i et T_j sont attachés à deux produits chimiques réactifs. A cet effet, chaque tag T_i dans T-MATCH stocke un chiffrement IND-CPA homomorphique Enc de son attribut a_{T_i} (i.e., type de produit chimique). Lorsque deux tags T_i et T_j sont lus par le lecteur R_k, ce dernier récupère les chiffrements $\mathsf{Enc}(a_{T_i})$ et $\mathsf{Enc}(a_{T_j})$ des attributs de tags T_i et T_j respectivement. Afin de protéger la vie privée des tags RFID, le lecteur R_k rechiffre les chiffrements stockés dans les tags T_i et T_j. Maintenant, pour évaluer la fonction Check, R_k utilise la propriété homomorphique de Enc pour calculer un chiffrement $\mathsf{Enc}(f(a_{T_i}, a_{T_j}))$ d'une fonction f des attributs a_{T_i} et a_{T_j}. Ensuite, le lecteur R_k et le serveur backend S s'engagent dans un protocole de test d'égalité (84) pour vérifier si $f(a_{T_i}, a_{T_j}) \in \mathsf{Ref}$, où Ref est l'ensemble des références d'appariement stockées dans le serveur S. Si c'est le cas, la fonction Check renvoie $b = 1$; sinon, Check renvoie $b = 0$.

Sécurité et vie privée de T-MATCH***.*** Pour protéger la vie privée et la sécurité des tags RFID, chaque tag T_i dans T-MATCH stocke un chiffrement BGN (26) de son attribut a_{T_i} et un code d'authentification de message (MAC) de ce chiffrement. À chaque exécution du protocole, le chiffrement BGN est rechiffré par un lecteur R_k dans la chaîne d'approvisionnement et le MAC est recalculé. Maintenant, en vue de protéger la vie privée des tags contre le lecteur R_k la chaîne d'approvisionnement et le serveur backend S, T-MATCH s'appuie sur un protocole de test d'égalité de texte préservant la vie privée et qui est exécuté conjointement par le lecteur R_k et le serveur backend S. Aussi, T-MATCH utilise les permutations pour s'assurer que la seule information divulguée à la fin d'une exécution du protocole est le bit b indiquant si le pair de tags participant à l'exécution du protocole sont attachés à des produits chimiques réactifs ou non.

De plus, pour empêcher le serveur backend S de forger de nouvelles références d'appariement, les attributs des tags dans T-MATCH sont encodés comme des "signatures" par l'émetteur

I qui est de confiance, et les références d'appariement sont calculées comme des couplages bilinéaires. Finalement, T-MATCH repose sur les MACs pour assurer l'intégrité des données stockées dans les tags.

B.2.3.2 Contributions

Pour résumer, les contributions majeures de T-MATCH sont les suivantes:

- T-MATCH propose une nouvelle solution pour l'appariement d'objets dans la chaîne d'approvisionnement qui cible uniquement les tags sans capacité de calcul. Les tags dans T-MATCH n'effectuent aucun calcul, par contre, ils doivent stocker seulement un état qui est mis à jour à chaque exécution du protocole par les lecteurs R_k de la chaîne d'approvisionnement.

- T-MATCH protège la vie privée des tags: T-MATCH repose sur des techniques du calcul multipartites sûres pour garantir que ni les lecteurs R_k ni le serveur backend S peuvent divulguer le contenu d'un tag.

- T-MATCH est sûr: Les lecteurs R_k ne déclenchent une alarme sauf quand ils interagissent avec un pair de tags attachés à des produits chimiques réactifs.

B.3 Conclusion

Alors que la prolifération des tags RFID est admise à être financièrement avantageuse, le déploiement de cette technologie vient toujours avec une variété de menaces de la vie privée et de sécurité qui vont du déni de service à l'espionnage industriel. Bien que la cryptographie offre déjà des solutions qui peuvent remédier à la plupart de ces menaces en théorie, elle reste trop coûteuse dans la pratique, pour des dispositifs aussi contraints que les tags RFID. Le dilemme d'assurer la sécurité et la vie privée des systèmes RFID tout en gardant les exigences de calcul dans les tags minimales, a donné lieu à une multitude de travaux sur l'authentification RFID et sur les formalisations de sécurité et de la vie privée. Cependant, la tâche de concevoir des protocoles d'authentification qui protègent la vie privée et qui répondent aux limitations de calcul de la technologie RFID s'était avérée difficile, voire impossible. En réalité, les formalisations existantes de la vie privée de systèmes RFID supposent un adversaire fort contre lequel la vie privée ne peut être assurée sans pour autant sacrifier la scalabilité et la rentabilité de la technologie RFID.

En conséquence, dans cette thèse, on s'est concentré d'abord à combler cet écart entre la formalisation théorique de la vie privée et les aspects pratiques de la technologie RFID, en supposant un adversaire qui ne peut pas surveiller en permanence les tags: *il y a au moins une interaction entre les tags et les lecteurs qui n'est pas observée par l'adversaire.* Ensuite, on a conçu quatre protocoles multipartites qui se basent sur la technologie RFID et qui fournissent

des solutions efficaces et sûres pour les applications de la chaîne d'approvisionnement. Plus précisément, on a introduit:

- Un protocole de transfert de propriété avec vérification d'origine;
- deux protocoles de vérification d'authenticité de produits qui peuvent être implémentés dans les tags sans capacité de calcul;
- un protocole d'appariement d'objets qui facilite la mise en application des règles de sécurité dans la chaîne d'approvisionnement en utilisant des tags sans capacité de calcul.

Bibliography

[1] Dictionary.com's 21st Century Lexicon, Jun 2012. http://dictionary.reference.com/browse/supplychain. 61, 178

[2] Alien Technology. RFID Tags, 2009. http://www.alientechnology.com/tags/index.php. 116

[3] G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable RFID tags via insubvertible encryption. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 92–101, New York, NY, USA, 2005. ACM. ISBN 1-59593-226-7. 32, 54, 58, 103, 176, 178

[4] G. Ateniese, J. Kirsch, and M. Blanton. Secret Handshakes with Dynamic and Fuzzy Matching. In *Proceedings of the Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2007. 132, 156, 186

[5] G. Avoine. Adversarial Model for Radio Frequency Identification. Cryptology ePrint Archive, Report 2005/049, 2005. http://eprint.iacr.org/2005/049.pdf. 2, 32, 33, 37, 38, 63, 70, 170, 177

[6] G. Avoine and P. Oechslin. A scalable and provably secure hash-based RFID protocol. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pages 110–114, March 2005. 33, 49

[7] G. Avoine and P. Oechslin. RFID Traceability: A Multilayer Problem. In *Financial Cryptography and Data Security*, volume 3570 of *Lecture Notes in Computer Science*, pages 577–577. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-26656-3. 57, 177

[8] G. Avoine and A. Tchamkerten. An Efficient Distance Bounding RFIDÂăAuthenticationÂăProtocol: Balancing False-Acceptance Rate and Memory Requirement. In *Information Security*, volume 5735 of *Lecture Notes in Computer Science*, pages 250–261. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-04473-1. 31, 175

[9] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. C. Wong. Secret Handshakes from Pairing-Based Key Agreements. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, page 180, Los Alamitos, CA, USA, 2003. IEEE Computer Society. ISBN 0-7695-1940-7. 23, 132, 156, 187

[10] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM. ISBN 0-89791-629-8. 10

[11] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 26–45, London, UK, 1998. Springer-Verlag. ISBN 3-540-64892-5. xv, 14, 15, 16

[12] C. Berbain, O. Billet, J. Etrog, and H. Gilbert. An efficient forward private RFID protocol. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 43–53, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-894-0. 49, 50

[13] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *Information Theory, IEEE Transactions on*, 24(3):384–386, May 1978. 44

[14] O. Billet and K. Elkhiyaoui. Two Attacks against the F_f RFID Protocol. In *Progress in Cryptology - INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 308–320. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-10627-9. 2, 5, 43, 47, 170

[15] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography*. Cambridge University Press, New York, NY, USA, 1999. ISBN 0-521-65374-6. 18

[16] I. F. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels. *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series), Chapter IX, pages 183-213*. Cambridge University Press, New York, NY, USA, 2005. ISBN 052160415X. 18

[17] E.O. Blass and M. Zitterbart. Towards Acceptable Public-Key Encryption in Sensor Networks. In *Proceedings of ACM 2nd International Workshop on Ubiquitous Computing*, pages 88–93, Miami, USA, 2005. ISBN 972-8865-24-4. 116

[18] E.O. Blass, A. Kurmus, R. Molva, G. Noubir, and A. Shikfa. The F_f-Family of Protocols for RFID-Privacy and Authentication. *IEEE Transactions on Dependable and Secure Computing*, 2010. ISSN 1545-5971. 2, 32, 43, 45, 46, 170, 177

[19] E.O. Blass, K. Elkhiyaoui, and R. Molva. Tracker: security and privacy for RFID-based supply chains. In *NDSS'11, 18th Annual Network and Distributed System Security Symposium, 6-9 February 2011, San Diego, California, USA, ISBN 1-891562-32-0*, 02 2011. 5

[20] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, July 2003. ISSN 0004-5411. 44

[21] L. Bolotnyy and G. Robins. Physically Unclonable Function-Based Security and Privacy in RFID Systems. In *Pervasive Computing and Communications, 2007. PerCom '07. Fifth Annual IEEE International Conference on*, pages 211–220, march 2007. 56, 57

[22] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32:586–615, March 2003. ISSN 0097-5397. 23, 77

[23] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 514–532, London, UK, 2001. Springer-Verlag. ISBN 3-540-42987-5. 23, 77

[24] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques*, EUROCRYPT'03,

pages 416–432, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-14039-5. 129

[25] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. *Journal of Cryptology*, 17:297–319, 2004. ISSN 0933-2790. 23

[26] D. Boneh, E-J. Goh, and K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *TCC*, pages 325–341, Cambridge, MA, USA, 2005. 23, 142, 143, 156, 187

[27] S. Brands and D. Chaum. Distance-bounding protocols. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 344–359, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc. ISBN 3-540-57600-2. 31, 175

[28] E. Brier, J.S. Coron, T. Icart, D. Madore, H. Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 237–254. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-14622-0. 79, 108, 160, 161

[29] J. Bringer, H. Chabanne, and E. Dottax. HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In *Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU 2006), 29 June 2006, Lyon, France*, pages 28–33. IEEE Computer Society, 2006. ISBN 0-7695-2549-0. 43, 44, 45

[30] J. Bringer, H. Chabanne, and T. Icart. Cryptanalysis of EC-RAC, a RFID Identification Protocol. In *Cryptology and Network Security*, volume 5339 of *Lecture Notes in Computer Science*, pages 149–161. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-89640-1. 53

[31] T. Burbridge and A. Soppera. Supply chain control using a RFID proxy re-signature scheme. In *RFID, 2010 IEEE International Conference on*, pages 29–36, april 2010. 129

[32] M. Burmester, B. Medeiros, and R. Motta. Provably Secure Grouping-Proofs for RFID Tags. In *Proceedings of the 8th IFIP WG 8.8/11.2 international conference on Smart Card Research and Advanced Applications*, CARDIS '08, pages 176–190, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-85892-8. 162

[33] S. Canard and I. Coisel. Data Synchronization in Privacy-Preserving RFID Authentication Schemes. In *Workshop on RFID Security – RFIDSec'08*, 7 2008. 50

[34] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51:557–594, July 2004. ISSN 0004-5411. 10

[35] J.Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18 (2):143–154, 1979. ISSN 0022-0000. 50

[36] C. Castelluccia and G. Avoine. Noisy Tags: A Pretty Good Key Exchange Protocol for RFID Tags. In *Smart Card Research and Advanced Applications*, volume 3928 of *Lecture Notes in Computer Science*, pages 289–299. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-33311-1. 55, 56

[37] H. Chabanne and G. Fumaroli. Noisy Cryptographic Protocols for Low-Cost RFID Tags. *Information Theory, IEEE Transactions on*, 52(8):3562–3566, August 2006. ISSN 0018-9448. 55, 56

[38] Sanjit Chatterjee and Alfred Menezes. On Cryptographic Protocols Employing

Asymmetric Pairings – The Role of Ψ Revisited. Cryptology ePrint Archive, Report 2009/480, 2009. http://eprint.iacr.org/. 23

[39] K. Chawla, G. Robins, and W. Weimer. On Mitigating Covert Channels in RFID-Enabled Supply Chains. In *RFIDSec Asia*, Singapore, 2010. http://rfidsec2010.i2r.a-star.edu.sg. 129

[40] Cobis Consortium. Collaborative Business Items: Chemical drums use-case, 2007. http://www.cobis-online.de/files/live.stream.wvx. 131, 186

[41] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98*, pages 13–25. Springer-Verlag, 1998. 54, 117, 118, 128, 185

[42] I. Damgård and M. Pedersen. RFID Security: Tradeoffs between Security and Efficiency. In *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 318–332. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-79262-8. 2, 50, 51, 170

[43] B. Danev, T. S. Heydt-Benjamin, and S. Čapkun. Physical-layer identification of RFID devices. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 199–214, Berkeley, CA, USA, 2009. USENIX Association. 57, 177, 178

[44] E. De Cristofaro and G. Tsudik. Practical Private Set Intersection Protocols with Linear Complexity. In Radu Sion, editor, *Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 143–159. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-14576-6. 103

[45] E. De Cristofaro, J. Kim, and G. Tsudik. Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model. In *Advances in Cryptology - ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 213–231. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-17372-1. 103

[46] R. H. Deng, Y. Li, M. Yung, and Y. Zhao. A new framework for RFID privacy. In *Proceedings of the 15th European conference on Research in computer security*, ESORICS'10, pages 1–18, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15496-4, 978-3-642-15496-6. 34, 35, 39, 43

[47] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal. Design and Implementation of PUF-Based "Unclonable" RFID ICs for Anti-Counterfeiting and Security Applications. In *RFID, 2008 IEEE International Conference on*, pages 58–64, april 2008. 56

[48] R. Di Pietro and R. Molva. Information confinement, privacy, and security in RFID systems. In *ESORICS 2007, 12th European Symposium On Research In Computer Security, September 24-26, 2007, Dresden, Germany / Also published in LNCS, Volume 4734/2008, ISBN: 978-3-540-74834-2*, Dresden, Germany, 09 2007. 2, 32, 33, 45, 50, 51, 170, 177

[49] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, nov 1976. ISSN 0018-9448. 21

[50] T. Dimitriou. A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 59–66, September 2005. 2, 32, 33, 50, 170, 177

[51] T. Dimitriou. rfidDOT: RFID delegation and ownership transfer made simple. In *Proceedings of International Conference on Security and privacy in Communication Networks*, Istanbul, Turkey, 2008. ISBN 978-1-60558-241-2. 50, 58, 70, 71, 93, 178

[52] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, USA, 1985. Springer New York, Inc. 54

[53] K. Elkhiyaoui, E.O. Blass, and R. Molva. Rotiv: Rfid ownership transfer with issuer verification. In *Proceedings of the 7th international conference on RFID Security and Privacy*, RFIDSec'11, pages 163–182, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-25285-3. 5

[54] K. Elkhiyaoui, E.O. Blass, and R. Molva. CHECKER: On-site Checking in RFID-based Supply Chains. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 173–184, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1265-3. doi: 10.1145/2185448.2185471. 5

[55] K. Elkhiyaoui, E.O. Blass, and R. Molva. T-MATCH: Privacy-Preserving Item Matching for Storage-Only RFID Tags. In *Workshop on RFID Security – RFIDSec'12*, Nijmegen, Netherlands, June 2012. 5

[56] EU project SToP. Stop Tampering of Products, 2010. http://www.stop-project.eu/. 95, 182

[57] J. Fan, J. Hermans, and F. Vercauteren. On the claimed privacy of EC-RAC III. In *Proceedings of the 6th international conference on Radio frequency identification: security and privacy issues*, RFIDSec'10, pages 66–74, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-16821-3, 978-3-642-16821-5. 53

[58] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 85–140. Springer Berlin / Heidelberg, 2004. ISBN 978-3-540-22666-6. 2, 32, 48, 170, 177

[59] K. Finkenzeller. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*. John Wiley & Sons, Inc., New York, NY, USA, 2 edition, 2003. ISBN 0470844027. 28, 29, 173

[60] S. Fouladgar and H. Afifi. An Efficient Delegation and Transfer of Ownership Protocol for RFID Tags. In *First International EURASIP Workshop on RFID Technology*, Vienna, Austria, September 2007. 66, 68, 93, 182

[61] D. Freeman, M. Scott, and E. Teske. A Taxonomy of Pairing-Friendly Elliptic Curves. *J. Cryptology*, 23(2):224–280, April 2010. ISSN 0933-2790. 23

[62] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, PKC '99, pages 53–68, London, UK, 1999. Springer-Verlag. ISBN 3-540-65644-8. 128

[63] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156:3113–3121, September 2008. ISSN 0166-218X. 23

[64] H. Gilbert, M. Robshaw, and H. Sibert. Active attack against HB+: a provably secure lightweight authentication protocol. *Electronics Letters*, 41(21):1169–1170, October 2005. ISSN 0013-5194. 2, 43, 44, 170

[65] H. Gilbert, M. Robshaw, and Y. Seurin. Good Variants of HB$^+$ Are Hard to Find. In *Financial Cryptography and Data Security*, volume 5143 of *Lecture Notes in Computer Science*, pages 156–170. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-85229-2. 45

[66] H. Gilbert, M. J. B. Robshaw, and Y. Seurin. HB#: increasing the security and efficiency of HB+. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 361–378, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78966-9, 978-3-540-78966-6. 2, 32, 33, 43, 44, 45, 170, 177

[67] M. Girault. Self-certified public keys. In *Advances in Cryptology âĂŤ EUROCRYPT âĂŹ91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer Berlin / Heidelberg, 1991. ISBN 978-3-540-54620-7. 52

[68] M. Girault, G. Poupard, and J. Stern. On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. *Journal of Cryptology*, 19:463–487, 2006. ISSN 0933-2790. 52

[69] O. Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17 of *Algorithmics and Combinatorics*. Springer, 1999. 8, 9

[70] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521830842. 9, 135, 138

[71] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33:792–807, August 1986. ISSN 0004-5411. 11, 12

[72] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17:281–308, April 1988. ISSN 0097-5397. 17

[73] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal Re-encryption for Mixnets. In *In Proceedings of the 2004 RSA Conference, Cryptographer's track*, pages 163–178. Springer-Verlag, 2002. 32, 176

[74] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5. 156

[75] J. Ha, S. Moon, J. Zhou, and J. Ha. A New Formal Proof Model for RFID Location Privacy. In *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 267–281. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-88312-8. 38, 39, 40

[76] G. Hammouri and B. Sunar. PUF-HB: a tamper-resilient HB based authentication protocol. In *Proceedings of the 6th international conference on Applied cryptography and network security*, ACNS'08, pages 346–365, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-68913-3, 978-3-540-68913-3. 56

[77] G. P. Hancke and M. G. Kuhn. An RFID Distance Bounding Protocol. In

First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm 2005, Athens, Greece, 5-9 September, 2005, pages 67–73. IEEE, 2005. ISBN 0-7695-2369-2. 31, 175

[78] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003. ISBN 038795273X. 18

[79] D.E. Holcomb, W.P. Burleson, and K. Fu. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *Computers, IEEE Transactions on*, 58(9):1198–1210, sept. 2009. ISSN 0018-9340. 56

[80] N. J. Hopper and M. Blum. Secure Human Identification Protocols. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 52–66, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42987-5. 44

[81] T. Icart. How to Hash into Elliptic Curves. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 303–316. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-03355-1. 144

[82] S. Inoue and H. Yasuura. RFID Privacy Using User-controllable Uniqueness. In *RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003. 32, 176

[83] International Medical Products Anti-Counterfeiting Taskforce. International Medical Products Anti-Counterfeiting Taskforce – IMPACT, 2010. http://www.who.int/impact/. 95, 182

[84] M. Jakobsson and A. Juels. Mix and Match: Secure Function Evaluation via Ciphertexts. In *Advances in Cryptology at ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177. Springer Berlin / Heidelberg, 2000. ISBN 978-3-540-41404-9. 132, 142, 187

[85] RFID Journal, Jun 2012. http://www.rfidjournal.com/faq/20. 1

[86] A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In *Proceedings of the 4th International Symposium on Algorithmic Number Theory*, ANTS-IV, pages 385–394, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67695-3. 23

[87] A. Joux and Kim Nguyen. Separating Decision Diffie–Hellman from Computational Diffie–Hellman in Cryptographic Groups. *Journal of Cryptology*, 16:239–247, 2003. ISSN 0933-2790. 22, 23

[88] A. Juels. Yoking-Proofs for RFID Tags. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, PERCOMW '04, pages 138–, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2106-1. 162

[89] A. Juels. RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, 2006. 30, 174

[90] A. Juels and R. Pappu. Squealing Euros: Privacy Protection in RFID-Enabled Banknotes. In *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 103–121. Springer Berlin / Heidelberg, 2003. ISBN 978-3-540-40663-1. 32, 176

[91] A. Juels and S. Weis. Authenticating Pervasive Devices with Human Protocols. In

Advances in Cryptology – CRYPTO 2005, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-28114-6. 2, 32, 33, 43, 44, 170, 177

[92] A. Juels and S.A. Weis. Defining Strong Privacy for RFID. In *PerCom Workshops*, pages 342–347, White Plains, USA, 2007. ISBN 978-0-7695-2788-8. 2, 32, 33, 37, 38, 51, 63, 68, 69, 70, 170, 177

[93] A. Juels, R. L. Rivest, and M. Szydlo. The blocker tag: selective blocking of RFID tags for consumer privacy. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 103–111, New York, NY, USA, 2003. ACM. ISBN 1-58113-738-9. 32, 55, 176

[94] A. Juels, P. F. Syverson, and D. V. Bailey. High-Power Proxies for Enhancing RFID Privacy and Utility. In *Privacy Enhancing Technologies*, volume 3856 of *Lecture Notes in Computer Science*, pages 210–226. Springer, 2005. ISBN 3-540-34745-3. 31, 176

[95] T. Karygiannis, B. Eydt, G. Barber, L. Bunn, and T. Phillips. Guidelines for Securing Radio Frequency Identification (RFID) Systems. *NIST Special Publication 800–98*, page 154, April 2007. 30, 31, 175

[96] J. Katz and A. Y. Lindell. Aggregate Message Authentication Codes. In *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 155–169. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-79262-8. 129

[97] J. Katz, A. Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 146–162, Berlin, Heidelberg, 2008. Springer-Verlag. 142

[98] J. Katz, J. Shin, and A. Smith. Parallel and Concurrent Security of the HB and HB$^+$ Protocols. *Journal of Cryptology*, 23: 402–421, 2010. ISSN 0933-2790. 44

[99] C. Kim, G. Avoine, F. Koeune, F.X. Standaert, and O. Pereira. The Swiss-Knife RFID Distance Bounding Protocol. In *Information Security and Cryptology – ICISC 2008*, volume 5461 of *Lecture Notes in Computer Science*, pages 98–115. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-00729-3. 31, 175

[100] H. Krawczyk. LFSR-based Hashing and Authentication. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 129–139, London, UK, UK, 1994. Springer-Verlag. ISBN 3-540-58333-5. 50

[101] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan. Lightweight Mutual Authentication and Ownership Transfer for RFID Systems. In *INFOCOM*, pages 251–255, 2010. 66, 93, 182

[102] S. S. Kumar and C. Paar. Are Standards Compliant Elliptic Curve Cryptosystems feasible on RFID? In *Proceedings of Workshop on RFID Security*, 2006. 43, 52

[103] Y. K. Lee, L. Batina, and I. Verbauwhede. EC-RAC (ECDLP Based Randomized Access Control): Provably Secure RFID authentication protocol. In *RFID, 2008 IEEE International Conference on*, pages 97–104, april 2008. 2, 32, 33, 53, 54, 170, 177

[104] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede. Elliptic-Curve-Based Security Processor for RFID. *Computers, IEEE Transactions on*, 57(11):1514–1527, November 2008. ISSN 0018-9340. 43, 52, 53

[105] Y. K. Lee, L. Batina, and I. Verbauwhede. Untraceable RFID authentication protocols: Revision of EC-RAC. In *RFID, 2009 IEEE International Conference on*, pages 178–185, april 2009. ISBN 978-1-4244-3337-7. 53

[106] Y. K. Lee, L. Batina, D. Singelée, and I. Verbauwhede. Low-Cost Untraceable Authentication Protocols for RFID. In Susanne Wetzel, Cristina Nita-Rotaru, and Frank Stajano, editors, *Proceedings of the 3rd ACM Conference on Wireless Network Security – WiSec'10*, pages 55–64, Hoboken, New Jersey, USA, March 2010. ACM, ACM Press. 53

[107] E. Levieil and P. Fouque. An Improved LPN Algorithm. In *Security and Cryptography for Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-38080-1. 44

[108] T. Li and R. Deng. Vulnerability Analysis of EMAP - An Efficient RFID Mutual Authentication Protocol. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 238–245, april 2007. 43

[109] T. Li and G. Wang. Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols. In *New Approaches for Security, Privacy and Trust in Complex Environments*, volume 232 of *IFIP International Federation for Information Processing*, pages 109–120. Springer Boston, 2007. 43

[110] Y. Li and X. Ding. Protecting RFID communications in supply chains. In *Proceedings of ACM Symposium on Information, Computer and Communications Security*, pages 234–241, Singapore, 2007. ISBN 1-59593-574-6. 96, 129, 185

[111] C. H. Lim and T. Kwon. Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *International Conference on Information and Communications Security – ICICS'06*, volume 4307 of *Lecture Notes in Computer Science*, pages 1–20, Raleigh, North Carolina, USA, December 2006. Springer. 58, 68, 70, 71, 74, 93, 178

[112] C. Ma, Y. Li, R. H. Deng, and T. Li. RFID privacy: relation between two notions, minimal condition, and efficient construction. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 54–65, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-894-0. 39

[113] M. McLoone and M. Robshaw. Public Key Cryptography and RFID Tags. In *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 372–384. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-69327-7. 2, 32, 52, 170, 177

[114] A. Menezes, S. Vanstone, and T. Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC '91, pages 80–89, New York, NY, USA, 1991. ACM. ISBN 0-89791-397-3. 23

[115] K. Michael and L. McCathie. The Pros and Cons of RFID in Supply Chain Management. In *Proceedings of the International*

Conference on Mobile Business, ICMB '05, pages 623–629, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2367-6. 61, 178, 179

[116] A. Miyaji, M. Nakabayashi, and S. Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 2001. 23

[117] D. Molnar, A. Soppera, and D. Wagner. A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags. In *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 276–290. Springer Berlin / Heidelberg, 2006. 50, 51, 66, 93, 182

[118] Motorola. Saudi Arabia's luxury retailer Jade Jewellery implements Motorola's RFID technology to improve inventory management and security, 2010. http://tinyurl.com/yg6wzjv. 95, 182

[119] J. Munilla and A. Peinado. HB-MP: A further step in the HB-family of lightweight authentication protocols. *Computer Networks*, 51(9):2262–2267, 2007. ISSN 1389-1286. 45

[120] C. Ng, W. Susilo, Y. Mu, and R. Safavi-Naini. RFID Privacy Models Revisited. In *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 251–266. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-88312-8. 42

[121] G. Noubir, K. Vijayan, and H. J. Nussbaumer. Signature-based method for runtime fault detection in communication protocols. *Computer Communications Journal*, 21(5):405–421, 1998. ISSN 0140-3664. 106

[122] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. In *RFID Privacy Workshop*, MIT, Massachusetts, USA, November 2003. 2, 32, 33, 37, 49, 170, 177

[123] T. Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *Advances in Cryptology at CRYPTO' 92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer Berlin / Heidelberg, 1993. ISBN 978-3-540-57340-1. 53

[124] T. Okamoto. Cryptography Based on Bilinear Maps. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 3857 of *Lecture Notes in Computer Science*, pages 35–50. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-31423-3. 18

[125] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Eurocrypt '98, LNCS 1403*, pages 308–318. Springer-Verlag, 1998. 143

[126] Y. Oren and M. Feldhofer. A low-resource public-key identification scheme for RFID tags and sensor nodes. In *Proceedings of the second ACM conference on Wireless network security*, WiSec '09, pages 59–68, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-460-7. 2, 32, 33, 53, 170, 177

[127] K. Ouafi and S. Vaudenay. Pathchecker: an RFID Application for Tracing Products in Suply-Chains. In *Workshop on RFID Security – RFIDSec'09*, pages 1–14, Leuven, Belgium, 2009. http://www.cosic.esat.kuleuven.be/rfidsec09/Papers/pathchecker.pdf. 96, 128, 185

[128] K. Ouafi, R. Overbeck, and S. Vaudenay. On the Security of HB$^{\#}$ against a Man-in-the-Middle Attack. In *Advances in Cryptology - ASIACRYPT 2008*, volume

5350 of *Lecture Notes in Computer Science*, pages 108–124. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-89254-0. 2, 43, 45, 170

[129] R. Paise and S. Vaudenay. Mutual authentication in RFID: security and privacy. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, ASIACCS '08, pages 292–299, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-979-1. 2, 32, 34, 37, 42, 54, 69, 74, 162, 170, 177

[130] P. Peris-Lopez, J. C. Hern, J. M. Estevez Tapiador, and A. Ribagorda. LMAP: A real lightweight mutual authentication protocol for low-cost RFID tags. In *In: Proc. of 2nd Workshop on RFID Security*, page 06. Ecrypt, 2006. 43

[131] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, and A. Ribagorda. EMAP: An Efficient Mutual-Authentication Protocol for Low-Cost RFID Tags. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4277 of *Lecture Notes in Computer Science*, pages 352–361. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-48269-7. 43

[132] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 99–112, New York, NY, USA, 2006. ACM. ISBN 1-59593-518-5. 156

[133] M. O. Rabin. Digitalized Signatures and Public-key Functions as Intractable as Factorization. Technical report, MIT, Cambridge, MA, USA, 1979. 53

[134] M.O. Rabin. Fingerprinting by random polynomials. Technical Report TR-15-81, Center for Research in Computing Technology. Harvard University, Cambridge, Massachusetts, USA, 1981. 129

[135] Damith C. Ranasinghe, Daniel W. Engels, and Peter H. Cole. Low-Cost RFID Systems: Confronting Security and Privacy. In *In: Auto-ID Labs Research Workshop*. Portal, 2005. 28, 173

[136] M. Rieback, B. Crispo, and A. Tanenbaum. RFID Guardian: A Battery-Powered Mobile Device for RFID Privacy Management. In *Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 259–273. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-26547-4. 31, 176

[137] P. Rogaway and T. Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In *FSE*, volume 3017 of *Lecture Notes in Computer Science*. Springer, 2004. ISBN 3-540-22171-9. 10

[138] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0245-6. 57

[139] A.R. Sadeghi, I. Visconti, and C. Wachsmann. Anonymizer-Enabled Security and Privacy for RFID. In *8th International Conference on Cryptology And Network Security – CANS'09*, Kanazawa, Ishikawa, Japan, December 2009. Springer. ISBN 978-3-642-10432-9. 58, 103, 178

[140] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *Advances in Cryp-*

tology – EUROCRYPT 2005, volume 3494 of *Lecture Notes in Computer Science*, pages 557–557. Springer Berlin / Heidelberg, 2005. 156

[141] J. Saito and K. Sakurai. Grouping proof for RFID tags. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 2, pages 621 – 624 vol.2, march 2005. 162

[142] J. Saito, K. Imamoto, and K. Sakurai. Reassignment Scheme of an RFID Tag's Key for Owner Transfer. In *Embedded and Ubiquitous Computing*, volume 3823 of *Lecture Notes in Computer Science*, pages 1303–1312. Springer Berlin / Heidelberg, 2005. 66, 182

[143] C. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Advances in Cryptology âĂŤ EUROCRYPT âĂŹ89*, volume 434 of *Lecture Notes in Computer Science*, pages 688–689. Springer Berlin / Heidelberg, 1990. ISBN 978-3-540-53433-4. 53

[144] M. Scott. Authenticated ID-based Key Exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive, Report 2002/164, 2002. http://eprint.iacr.org/. 24

[145] A. Shamir. How to share a secret. *Commun. ACM*, 22:612–613, November 1979. ISSN 0001-0782. 134

[146] A. Shamir. Memory efficient variants of public-key schemes for smart card applications. In *Advances in Cryptology at EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 445–449. Springer Berlin / Heidelberg, 1995. ISBN 978-3-540-60176-0. 53

[147] A. Shamir. SQUASH – A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags. In *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 144–157. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-71038-7. 48

[148] H. Shuihua and C.-H. Chu. Tamper Detection in RFID-Enabled Supply Chains Using Fragile Watermarking. In *Proceedings of IEEE RFID*, pages 111–117, Las Vegas, USA, 2008. 129

[149] B. Song. RFID Tag Ownership Transfer. In *Workshop on RFID Security – RFIDSec'08*, Budapest, Hungary, July 2008. 68, 93

[150] M. Soos. Analysing the Molva and Di Pietro Private RFID Authentication Scheme. In *Workshop on RFID Security – RFIDSec'08*, Budapest, Hungary, July 2008. 51

[151] TAGSYS RFID. RFID Luxury Goods Solutions, 2010. http://www.tagsysrfid.com/Markets/Industries/Luxury-Goods. 95, 129, 182

[152] M. Tajima. Strategic value of RFID in supply chain management. *Journal of Purchasing and Supply Management*, 13(4): 261 – 273, 2007. ISSN 1478-4092. 61, 179

[153] G. Tsudik. YA-TRAP: yet another trivial RFID authentication protocol. In *International Conference on Pervasive Computing and Communications Workshops*, Pisa, Italy, 2006. ISBN 0-7695-2520-2. 2, 32, 33, 170, 177

[154] P. Tuyls and L. Batina. RFID-Tags for Anti-counterfeiting. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 115–131. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-31033-4. 56, 57

[155] UPM RFID Technology. UPM Raflatac MiniTrak datasheet, 2011. http://www.upmrfid.com/rfid/images/MiniTrack_SLI_datasheet.pdf/$FILE/MiniTrack_SLI_datasheet.pdf. 128

[156] UPM RFID Technology. UPM RFID HF RaceTrack RFID Tag, 2011. http://www.rfidtags.com/upm-rfid-racetrack-rfid-tag. 155

[157] I. Vajda and L. Buttyán. Lightweight Authentication Protocols for Low-Cost RFID Tags. In *In Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003*, 2003. 43

[158] Ton van Deursen and Sasa Radomirovic. Untraceable RFID protocols are not trivially composable: Attacks on the revision of EC-RACs. Cryptology ePrint Archive, Report 2009/332, 2009. http://eprint.iacr.org/. 53

[159] S. Vaudenay. On Privacy Models for RFID. In *Proceedings of ASIACRYPT*, pages 68–87, Kuching, Malaysia, 2007. ISBN 978-3-540-76899-9. 2, 32, 33, 34, 35, 36, 37, 40, 41, 42, 43, 53, 54, 69, 71, 74, 102, 104, 141, 162, 170, 171, 177

[160] Verayo. Verayo Anti-Counterfeiting Solution, 2010. http://www.verayo.com/solution/anti-counterfeiting.html. 95, 129, 182

[161] E. Verheul. Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems. In *Advances in Cryptology âĂŤ EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer Berlin / Heidelberg, 2001. ISBN 978-3-540-42070-5. 22

[162] L. C. Washington. *Elliptic Curves: Number Theory and Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 2003. ISBN 1584883650. 18, 20

[163] S. Weis, S. Sarma, R. Rivest, and D. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *Security in Pervasive Computing*, volume 2802 of *Lecture Notes in Computer Science*, pages 50–59. Springer Berlin / Heidelberg, 2004. ISBN 978-3-540-20887-7. 48

[164] D. Zanetti, B. Danev, and S. Čapkun. Physical-layer identification of UHF RFID tags. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, MobiCom '10, pages 353–364, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0181-7. 57, 177

[165] D. Zanetti, P. Sachs, and S. Capkun. On the practicality of UHF RFID fingerprinting: how real is the RFID tracking problem? In *Proceedings of the 11th international conference on Privacy enhancing technologies*, PETS'11, pages 97–116, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-22262-7. 163