Security and privacy in online social networks

Leucio Antonio Cutillo

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2012-ENST-020

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

Leucio Antonio CUTILLO

le 5 Avril 2012

Protection des Données Privées dans les Réseaux Sociaux

Directeur de thèse : **Professeur Refik MOLVA**

Jury

M. Claude CASTELLUCCIA, Directeur de Recherche, INRIA, Saint Ismier	Rapporteur
M. Jon CROWCROFT, Professeur, University of Cambridge, Cambridge	Rapporteur
M. Antonio LIOY, Professeur, Politecnico di Torino, Torino	Examinateur
M. David HALES, Docteur, The Open University, Milton Keynes	Examinateur

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

THÈSE

Uno solo è il mio desiderio, quello di vedervi felici nel tempo e nell'eternità.
sac. Giovanni Bosco

Abstract

Social network applications allow people to establish links and exchange information based on various interests such as professional activities, hobbies, et similia. Several commercial social networking platforms that came to light recently suddenly became extremely popular at the international arena. Apart from obvious advantages in terms of fast community building, rapid exchange of information at the professional and private level, social network platforms raise several issues concerning the privacy and security of their users. The goal of this thesis is to identify privacy and security problems raised by the social networks and to come up with the design of radically new architectures for the social network platform. As current social network platforms are based on centralized architectures that inherently threat user privacy due to potential monitoring and interception of private user information, the goal is to design social network platforms based on a distributed architecture in order to assure user privacy. New mechanisms are investigated in order to solve some classical security and trust management problems akin to distributed systems by taking advantage of the information stored in the social network platforms. Such problems range from trust establishment in self-organizing systems to key management without infrastructure to cooperation enforcement in peer-to-peer systems.

This thesis suggests a new approach to tackle these security and privacy problems with a special emphasis on the privacy of users with respect to the application provider in addition to defense against intruders or malicious users. In order to ensure users' privacy in the face of potential privacy violations by the provider, the suggested approach adopts a decentralized architecture relying on cooperation among a number of independent parties that are also the users of the online social network application. The second strong point of the suggested approach is to capitalize on the trust relationships that are part of social networks in real life in order to cope with the problem of building trusted and privacy-preserving mechanisms as part of the online application. The combination of these design principles is Safebook,

a decentralized and privacy-preserving online social network application. Based on the two design principles, decentralization and exploiting real-life trust, various mechanisms for privacy and security are integrated into Safebook in order to provide data storage and data management functions that preserve users'privacy, data integrity, and availability.

Apart from the design of Safebook, a significant part of the thesis is devoted to its analysis and evaluation using various methods such as experimenting with real social network platforms.

Finally, this thesis presents an implementation of Safebook that is written in python and can be executed on multiple operating systems such as Windows, Linux and MacOs. The Safebook implementation is a multithread event-driven application composed by different managers in charge of building and keeping the social network and P2P overlays, performing cryptography operations and providing the main social network facilities such as friendship lookup, wall posting and picture sharing through a user interface implemented under the form of a webpage.

Acknowledgments

This dissertation is the result of three years and a half of research supported by ideas, experiments, prototypes a lot of students, colleagues and friends contributed to.

My intellectual debt to prof. Refik Molva, prof. Thorsten Strufe, Dr. Melek Onen and Dr. Matteo dell'Amico is enormous. With their patient help, I've started taking my first steps into the amazing world of research.

Many thanks to prof. Pietro Michiardi, Dr. Oliver Blass, Carmelo Velardo, Alessandro Duminuco, Marco Paleari, Antonio Barbuzzi, Giuseppe Reina and Mario Pastorelli for their strong influence on my thinking during lots of problem identification and solving steps.

A special acknowledgement goes to seventeen among the best students I have ever met: Dennis Roch, Yao Liu, Jens Trinh, Etienne Peron, Jean Baptiste Barrau, Luca Boasso, Paolo Viotti, Mustafa Zengin, Marco Garieri, Wenting Li, Girolamo Piccinni, Andrea Milazzo, Esko Mattila, Waqas Liaqat Ali, Rajat Rajendra Hubli, Yu Liu, and Yuling Shi. Their help in translating the theory of this work to the practice of a real software prototype was crucial.

Finally, my last and biggest acknowledgment goes to my father Angelo and my girlfriend Veronica, they always supported me in every difficult moment.
This thesis is dedicated to them.

Contents

Abstract i
Acknowledgments iii
Contents v
List of Figures xi
List of Tables xv
Acronyms xix
Notations xxii

1 Introduction **1**
1.1 Research objectives 3
1.2 Main contributions 4
1.3 Thesis organization 6

I Security and Privacy Issues in OSN **9**

2 Online Social Networks **11**
2.1 Social Network Providers and Their Customers 13
2.2 Functional Overview of Online Social Networks 14
2.2.1 Networking functions. 15
2.2.2 Data functions. 15
2.2.3 Access control functions. 16
2.3 Data contained in Online Social Networks 17
2.4 Summary 20

3 Main threats in OSN **23**
3.1 Security and privacy objectives . . . 24
3.1.1 Privacy . . . 24
3.1.2 Integrity . . . 25
3.1.3 Availability . . . 25
3.2 Attack Spectrum and Countermeasures . . . 28
3.3 The "Big Brother" problem . . . 38
3.4 Summary . . . 41

4 Decentralized OSN **43**
4.1 Client-Server based Decentralized OSNs . . . 44
4.2 P2P-based Decentralized OSNs . . . 45
4.3 Main Limitations . . . 48
4.4 Summary . . . 49

II A privacy preserving distributed OSN leveraging real life trust **51**

5 Safebook **53**
5.1 Rationale . . . 54
5.1.1 Design principles . . . 55
5.1.2 Idea of the solution . . . 56
5.2 Main components . . . 59
5.2.1 Matryoshka . . . 59
5.2.2 Peer-to-peer substrate . . . 61
5.2.3 Trusted Identification Service . . . 62
5.3 Functionalities . . . 64
5.3.1 Data Management . . . 64
5.3.2 Key Management . . . 65
5.3.3 Communication Management . . . 66
5.4 Core protocols . . . 69
5.4.1 Profile Creation . . . 69
5.4.2 Social Network Service setup and maintenance . . . 70
5.4.3 Social Network Communication and Relationship management . . . 74

5.5 Summary . . . 78

6 Performance of the Approach 79

6.1 Mirror reachability - building one chain . . . 80
6.2 Data availability - Matryoshka feasibility . . . 81
6.3 Data storage and availability . . . 85
6.3.1 Maximum fragment size evaluation . . . 86
6.3.2 Retrieved data evaluation . . . 87
6.3.3 Profile size evaluation . . . 88
6.3.4 Example . . . 88
6.4 Summary . . . 89

7 Impact of social graphs on performance and privacy 93

7.1 Privacy from the graph theory perspective . . . 93
7.1.1 Node degree . . . 94
7.1.2 Clustering Coefficient . . . 95
7.1.3 Mixing time . . . 97
7.1.4 Results . . . 98
7.2 Impact of social graphs on Safebook . . . 99
7.2.1 Impact on privacy . . . 99
7.2.2 Impact on performance . . . 100
7.2.3 Performance and privacy trade-off . . . 101
7.3 Summary . . . 103

8 Implementation 105

8.1 Overall Architecture . . . 106
8.2 Account creation . . . 109
8.3 User interface and OSN facilities . . . 110
8.4 S2S: the P2P overlay of Safebook . . . 114
8.5 Additional challanges . . . 117
8.6 Summary . . . 117

9 Conclusion and future work 119

9.1 Directions for future research . . . 122

Appendices **125**

A Résumé étendu **127**
A.1 Objectifs de recherche . . . 129
A.2 Contributions principales . . . 130

B Further Matryoshka security features **147**
B.1 Matryoshka Verification Protocol . . . 148
B.2 Specific vulnerabilities . . . 149
B.2.1 Denial of Service and Traffic Analysis . . . 149

C Privacy Preserving Picture Sharing in Distributed OSNs **153**
C.1 Introduction . . . 154
C.2 Problem Statement . . . 156
C.2.1 Usage control for picture sharing in online social networks . . . 156
C.2.2 Decentralized online social networks . . . 156
C.3 The proposed usage control mechanism . . . 157
C.3.1 Safebook: a P2P DOSN leveraging real life social trust . . . 157
C.3.2 Overview of the solution . . . 159
C.3.3 Solution description . . . 160
C.4 Evaluation . . . 164
C.5 Conclusion and Future Work . . . 169

D PRICE: PRivacy preserving Incentives for Cooperation Enforcement **171**
D.1 Introduction . . . 172
D.2 Problem Statement . . . 173
D.2.1 Cooperation enforcement in P2P networks . . . 173
D.2.2 Credit-based incentive mechanisms . . . 173
D.2.3 Security and Privacy Challenges . . . 174
D.3 Solution Overview . . . 174
D.3.1 Environment . . . 175
D.3.2 Scenario . . . 175
D.4 Description . . . 176
D.4.1 Preliminaries . . . 176
D.4.2 Account creation . . . 177

D.4.3 Payment order . . . 179
D.4.4 Payment notification . . . 180
D.5 Evaluation . . . 181
D.5.1 Security . . . 181
D.5.2 Privacy . . . 182
D.5.3 Performance . . . 183
D.6 Related Work . . . 186
D.7 Conclusion . . . 189

Bibliography **191**

List of Figures

2.1 OSN customers and their relationships to PII and SNS 14
2.2 Main functionality of a typical OSN platform 17
2.3 Types of data commonly stored in OSN profiles. 18

3.1 Impersonation attacks: victim $\mathcal{U}$ doesn't have any OSN account, victim $\mathcal{V}$ has an account on OSN1 and victim $\mathcal{Z}$ on OSN2. The attacker $\mathcal{A}$ generates $\mathcal{U}$'s account on OSN2, a copy of $\mathcal{V}$'s account on OSN1 and OSN2, and logs on OSN2 with the credentials of $\mathcal{Z}$. 32
3.2 Main PII related threats in current OSNs. 33

5.1 Cyclic relation showing how real life trust between users can build the OSN itself. 57
5.2 Safebook overlays (left), main components (center) and Matryoshka (right). . 60
5.3 An example of communication between users with different ACPs. 67
5.4 Account creation for user $\mathcal{V}$. 71
5.5 Matryoshka setup for user $\mathcal{V}$. 72
5.6 Entrypoint registration for user $\mathcal{V}$'s Matryoshka. 73
5.7 A $\mathcal{V}$'s prism is leaving $\Theta_{\mathcal{V}}$. 74
5.8 $\mathcal{V}$'s data lookup. 75
5.9 Friendship advertisement in Safebook. 76
5.10 Profile data storage for $\mathcal{V}$. 77

6.1 online, offline and the corresponding residual life distributions derived from the Skype dataset 82
6.2 Chain residual lifetime with respect to h 83

6.3 Data availability where $\overline{f} = 130$, $p = 0.53$ and $\overline{f_l} = \overline{f} - 1$ (no overlapping between friend lists 84
6.4 Data availability where $\overline{f} = 130$, $p = 0.53$ and $\overline{f_l} = \overline{f} - ml$ (full overlapping between friend lists 85
6.5 Fragment size evaluation for different upload bandwidth c with varying request rate λ. 90
6.6 Maximum size of data retrieved at every request for different upload bandwidth c with varying request rate λ. 91

7.1 Log-log plot of the degree complementary cumulative distribution of real-life social networks. 95
7.2 Average clustering coefficient of real-life social networks with respect to node degree. 97
7.3 Mixing time of real-life social networks. 99
7.4 Ratio of common friends between two nodes $\mathcal{V}$ and θ^h at social distance h in the social network 103

8.1 Overall architecture of Safebook. 106
8.2 Internal (left) and external (right) message exchange in Safebook 107
8.3 Account Creation: out of band step on the left, in band step on the right. . . 111
8.4 The Safebook logo: two persons shaking hands represent the process at the basis of Matryoshka and, more generally, of Safebook. 112
8.5 Graphical interface of Safebook: on the top-left the Safebook join; on the top-right the profile page in the podium section; on the middle-left picture sharing in the gallery page; on the middle-right wall posting in the square; on the bottom-left friendship advertisement; on the bottom-right friend browsing in the contacts page. 113

A.1 Clients des services de réseaux sociaux et leur relations avec les informations personnellement identifiables. 132
A.2 Fonctionnalité principale d'un typique réseau social en ligne. 133
A.3 Types de données généralement enregistrées dans les profils des réseaux sociaux en ligne. 134

A.4 Les attaques d'usurpation d'identité: la victime $\mathcal{U}$ ne posséde aucun compte de reseau social, la victime $\mathcal{V}$ a un compte sur OSN1 et la victime $\mathcal{Z}$ sur OSN2. L'agresseur $\mathcal{A}$ génère un compte $\mathcal{V}$ sur l'OSN2, une copie du compte de $\mathcal{V}$ sur OSN1 et OSN2, et il s'enregistre sur OSN2 avec les informations d'identification de $\mathcal{Z}$. 135

A.5 Principales menaces liées à l'access aux informations personnellement identifiables dans les OSNs actuelles. 136

A.6 la relation cyclique montrant comment la confiance entre les utilisateurs dans la vie réelle peuve construire l'OSN elle-même. 136

A.7 Les recouvrements de Safebook (à gauche), les composants principaux (au centre) et la Matryoshka (à droite). 137

A.8 Un exemple de communication entre les utilisateurs avec des différents politiques d'access aux donnés partagés. 138

A.9 Les distributions des temps en ligne, hors ligne et correspondantes à la vie résiduelle provenant de l'ensemble de données Skype. 139

A.10 Durée de vie résiduelle de la chaîne par rapport à sa longueur h. 140

A.11 Taille maximale des données récupérées à chaque demande avec bande passante c en upload et différents taux de demandes λ. 141

A.12 Log-log plot de la distribution cumulative complementaire du degré dans des reseaux sociaux réelles. 142

A.13 Coefficient de clustering moyen dans des reseaux sociaux reelles par rapport au degré des connections. 142

A.14 Temps de mélange (en pas) dans des réseaux sociaux réels. 143

A.15 Ratio des amis communs entre les deux noeuds $\mathcal{V}$ et θ^h à une distance sociale h dans le reseau social. 144

A.16 Architecture globale de Safebook. 145

A.17 L'échange interne (à gauche) et externe (à droite) de messages en Safebook. . 145

A.18 l'interface graphique de Safebook: sur le coin supérieur gauche comment rejoindre Safebook; en haut à droite de la page le profil dans la section podium; au milieu à gauche le partage de photos dans le page de la galerie; au milieu à droite l'affichage sur le mur dans la page square; en bas à gauche l'annonce de l'amitié et en bas à droite la navigation dans la page des contacts. 146

B.1 A colluding intruder $\mathcal{M}$ in $\Theta_{\mathcal{V}}$. 150
B.2 A black hole $\mathcal{B}$ in $\Theta_{\mathcal{V}}$ (right). 151

C.1 The Matryoshka graph of a user $\mathcal{V}$, from [57] 159
C.2 Data lookup in Safebook, from [57] 159
C.3 Picture publication steps for $\mathcal{V}$, with $\mathcal{V}$'s face $f_{\mathcal{V}}$ made publicly available: 1- picture input; 2- face detection; 3- face tagging; 4- face extraction; 5- face obfuscation; 6- picture and publisher face publication. 161
C.4 Public picture advertisement: 1- N is informed about P; 2- face detection; 3- face tagging; 4- publisher face extraction; 5- face obfuscation; 6- picture and N's face publication according to N's access control policy on her face. 163
C.5 Average outdegree, average number of 4-hops chains, average number of served Matryoshka for different social network graphs when 30% of nodes are on-line. 165
C.6 Unauthorized picture broadcast by friendship relations establishment between a malicious $\mathcal{V}$ and any users $\mathcal{F}_i$, or by recursive collusion with nodes $\mathcal{C}_i$ not necessarily belonging to $\mathcal{V}$'s contact list. 166
C.7 Average number of compromised chains when 10% of nodes misbehave for different social network graphs. 167
C.8 Average number of compromised chains when 25% of nodes misbehave for different social network graphs. 168
C.9 Spread of information vs usage control: in case of unprotected picture publication, automatic face obfuscation is guaranteed by peer collaboration even when there is software manipulation; in case of encrypted publication, the software manipulation may violate user's privacy, but with limited impact within the DOSN. 169

D.1 Payment scheme. 176
D.2 Total transaction time evaluation: T_{RTT} and T_L from [110], T from Monte Carlo techniques (10000 samples). 184
D.3 Evaluation of the number of notaries t to be contacted for every transaction for different online- p and misbehaving- m probabilities. 186
D.4 Evaluation of the bandwidth consumption for different transaction rates λ (per hour) and notaries to be contacted t. 187

List of Tables

3.1 Attacks vs. Security Objectives in Online Social Networks 28
3.2 Current OSN and their characteristics. 41

4.1 Current DOSN proposals as an answer to the "Big Brother" problem in centralized OSNs 48

5.1 An example of ACP based on set operations between contacts granted with user-defined badges. 67

7.1 Main characteristics of five social graphs from Facebook ($\overline{p_\nu}$ computed assuming p_{mal}=0.01). 100
7.2 Characteristics summary of examined SN graphs. 101

D.1 Notation 178
D.2 Time statistics in seconds for the three main distributions in figure D.2 184
D.3 Coin registry size in MB for different values of k 185

Acronyms

These are the main acronyms used in this document. The meaning of an acronym is usually indicated once, when it first occurs in the text.

API	Application Programming Interface
AS	Application Service
CT	Communication and Transport
DHT	Distributed Hash Table
DEK	Data Encryption Key
DOSN	Distributed Online Social Network
KDC	Key Distribution Center
KEK	Key Encryption Key
OSN	Online Social Network
P2P	Peer-to-peer
PII	Personally Identifiable Information
SN	Social Network
SNA	Social Network Application
SNP	Social Network Providers

SNS Social Network Services

Notations

Generally, boldface upper-case letters denote matrices and boldface lower-case letters denote vectors (unless stated otherwise). Calligraphic upper-case letters denote sets. The superscript ^{T} stands for transpose.

$\mathcal{V}$	a user in the social network
$\mathcal{K}_{\mathcal{V}}^{-}$	a private key generated by user $\mathcal{V}$
$\mathcal{K}_{\mathcal{V}}^{+}$	a public key generated by user $\mathcal{V}$
$K_{\mathcal{V}}$	public-private keypair generated by user $\mathcal{V}$
M	a message
MK	a master key
b	a badge
$\mathcal{A}_{\mathcal{V}}$	the ACP of user $\mathcal{V}$
$\mathcal{B}_{\mathcal{V}}$	the set of all the badges defined by $\mathcal{V}$
$\mathcal{R}_{\mathcal{V}}$	the set of all the rules defined by $\mathcal{V}$
$\mathcal{S}_{\mathcal{V}}$	the distributed data storage space of user $\mathcal{V}$
r	a rule in the ACP $\mathcal{A}$
s_r	a seed associated to a rule r

$\mathcal{D}_r^n$	a set of n DEKs associated to rule r
$\mathcal{F}_\mathcal{V}$	the set of all the contacts of $\mathcal{V}$
$\mathcal{E}_\mathcal{V}^\mathcal{U}$	the set of all the DEKs sent by $\mathcal{V}$ to $\mathcal{U}$
$\{M\}_{S_{\mathcal{K}_\mathcal{V}}}$	a message M signed by the user $\mathcal{V}$'s private key $\mathcal{K}_\mathcal{V}^-$
$E_{\mathcal{K}_\mathcal{U}}\{M\}$	a message M encrypted with the user $\mathcal{U}$'s public key $\mathcal{K}_\mathcal{U}^+$
$Name_\mathcal{V}$	a tuple of properties identifying user $\mathcal{X}$
$NId_\mathcal{V}$	node identifier of user $\mathcal{V}$
$UId_\mathcal{V}$	user identifier of user $\mathcal{V}$
$Cert\left(NId_\mathcal{V}, \mathcal{N}_\mathcal{V}^+\right)$	node identifier certificate for user $\mathcal{V}$
$Cert\left(UId_\mathcal{V}, \mathcal{U}_\mathcal{V}^+\right)$	user identifier certificate for user $\mathcal{V}$
min	minimum of following expression
max	maximum of following expression
t	continuous time
$Pr[\cdot]$	probability of argument
$p(\cdot)$	probability density function
X	a random variable
$E[X]$	expected value of a random variable X
f_i	the ith element of vector f, if the latter is defined
$\overline{f}$	arithmetic mean of vector f
$\bigcup$	set union
$\setminus$	set element exclusion
$v \in V$	an element v in a set V

$\|\|V\|\|$	cardinality of a set V
$G(V,E)$	a graph composed by a set V of vertexes and and a set E of edges
$deg(\cdot)$	degree function
$c(\cdot)$	local clustering function
$C(\cdot)$	global clustering function
$B(n,p)$	binomial distribution with number of trials n and success probability in each trial p
R^h	random walk distribution after h hops
ssd	steady state distribution
$\Delta_x(h)$	variation distance
$\tau_x(\epsilon)$	mixing time

Chapter 1

Introduction

Social Networking Services (SNS), like *Facebook*, *LinkedIn*, or *Google+*, are a predominant factor of Internet. Catering for a very large user population with a vast difference in social, educational and national background, they allow even users with limited technical skills to publish personal information and to communicate with ease.

In general, the ***Online Social Networks*** (OSN) resulting from these SNSs are digital representations of a subset of the relations that their participants, the registered persons or institutions, entertain in the physical world. Spanning participating parties through their relationships, they model the social network as a graph. However, the popularity and broad acceptance of social networking services as platforms for messaging and socialising attracts not only faithful users who are trying to add value to the community, but parties with rather adverse interests, be they commercial or plain malicious, as well.

The main motivation for members to join an OSN, to create a profile, and to use the different applications offered by the service, is the possibility to easily share information with selected contacts or with the public for either *professional* or *personal* purposes. In the first case, the OSN is used as a facility geared toward career management or business goals, hence SNS with a more serious image, like *XING* or *LinkedIn*, are chosen. As members in this case are aware of the professional impact of the OSN, they usually pay attention to the content of the data they publish about themselves and others. In the case of a more

private use, they share more personal information like contact data, personal pictures, or videos. Other members in the shared pictures can be marked ("*tagged*"), and links to their respective profiles are created automatically.

The core application used by OSN members is the creation and maintenance of their contact lists, which describe the members' milieux and maps them into the digital OSN graph. By informing members automatically on profile changes of their contacts, the SNS thus helps users to stay up to date with news of their contacts and very often the popularity of users is measured in the number of contacts their profile links to.

Analyzing the OSN with respect to their security properties and the privacy of their users, some obvious threats become apparent. Generally, a wealth of personal data on the participants is stored at the providers, especially in the case of OSN targeting non-professional purposes. This data is either visible to the public, or, if the user is aware of privacy issues and able to use the settings of the respective SNS, to a somewhat selected group of other members. As profiles are attributed to presumably known persons from the real world, they are implicitly valued with the same trust as the presumed owner of the profile. Furthermore, any actions and interactions coupled to a profile are again attributed to the presumed owner of this profile, as well.

Different studies have shown that participants clearly represent the weak link for security in OSN and that they are vulnerable to several types of social engineering attacks. This is partially caused by a lack of awareness to the consequences of simple and presumably private actions, like accepting contact requests, or tagging pictures, as well as communication operations like commenting on profiles or posting on walls. The low degree of usability of privacy controls offered by the SNS, and finally and most importantly inherent assumptions about other participants and trust in other profiles, which are actually a desired characteristic, certainly add to the problem.

By analyzing the privacy problems in current OSN, it becomes apparent that even if all participants were aware of exposures and competent in the use of SNS, and even if a concise set of privacy measures were deployed, the OSN would still be exposed to potential privacy violations by the omniscient service provider: the data, directly or indirectly supplied by all participants, is collected and stored permanently at the databases of the service provider, which potentially becomes a ***"Big Brother"*** capable of exploiting this data in many ways that can violate the privacy of individual users or user groups. The importance of this privacy exposure is underlined by the market capitalization of these providers, which reaches

50 billion U.S. Dollars (Facebook Inc, according to the investment of Goldman Sachs and Digital Sky Technologies in 2011)[12], and by the OSN worldwide advertisements revenue, which reached 5 billion U.S. Dollars in 2011 and is estimated to double by 2013 (according to eMarketer [25]).

This thesis claims that the user's privacy can be easily jeopardized due to the centralized architecture of OSNs, and current providers are not likely to address this problem due to their business model. This work considers instead the protection of private data in OSN a pressing topic and proposes a new architecture for OSN with the purpose of privacy by design.

1.1 Research objectives

This thesis assumes the protection of the user's ***privacy against the omniscient SNS provider*** to be the main objective for OSN and aims at identifying the main characteristics an OSN should meet to achieve such an objective and at providing a new architecture for privacy preserving OSN. As an additional objective, the protection of the user's ***privacy against malicious users*** is also addressed.

We define the objective of privacy as the possibility to hide any information about any user at any time, even to the extent of hiding users' participation and activities within the OSN. Therefore, privacy not only encompasses the protection of personal information which users publish at their profiles, but also takes into account the communication between users, that is, it requires that no parties other than directly addressed or explicitly trusted ones should have the possibility to trace communication patterns. The details of messages have to be unobservable, so only the requesting and responding parties should know one another's identity and the content of the request. Access to the content of a user profile may only be granted by the user directly, and this access control has to be as fine-grained as the profile itself.

Together with the objective of privacy, this thesis addresses further security objectives of ***integrity*** and ***availability***, which in OSN come in slightly different flavors than in traditional systems.
In the context of OSNs, integrity has to be extended beyond the basic goal of protecting users' data and identity against unauthorized modification to cover a variety of attacks such as the creation of personae, bogus profiles, cloned profiles, or other types of impersonation.

Each profile should then be unambiguously associated to an individual in the real world. Availability should prevent denial-of-service attacks that in the context of OSN may aim at seizuring a victim's profile or disrupting the possibility to communicate with the user. Moreover, availability should not only achieve the basic goal of guaranteeing SNS even in face of attacks and faults, but also target robustness against censorship.

1.2 Main contributions

The centralized nature of OSNs allows SNS providers to monitor and intercept user's sensitive data. This problem recently attracted quite some interest in the research community and the outcome of the research can be summarized in a family of solutions known as ***Decentralized Online Social Networks*** (DOSN). Such DOSNs aim at distributing the user's data with the adoption of a client-server (or cloud) approach, where users do not participate in the storage service and the stored data is always available, or through a peer-to-peer (P2P) approach, where users participate in the storage service and the stored data may not be always available.

Even though the user's shared data is protected by encryption in all the current DOSNs, such solutions are not suitable to achieve our research objectives. Client-server (or cloud) approaches do not always evade the potential control of a single party, as e.g. a company or an organization, on the hosted user's data. Such control evasion might have been achieved if users had set up and maintained their own servers to host their data and that one of other users, thus leading to a P2P-like approach.
However, current P2P DOSNs suffer from exposures to communication tracing by malicious peers. In the context of OSN, such communication traces are likely to correspond to friendship relationships in the social network, and therefore they can even disclose details on the structure of the social network graph. Among the current P2P-based DOSN approaches, none of them addresses this problem. In addition, current P2P DOSNs often leverage on existing P2P architectures suffering from the well known problems of lack of cooperation due to selfishness of nodes and denial-of-service attacks due to the creation of multiple peer identities under the control of a malicious party. For these reasons, current P2P DOSNs no not seem suitable for the goal of privacy preserving OSN.

With this work, we hope to provide a basis for new research focusing on privacy in OSN. Business statistics, newspapers and current research let us strongly believe the relevance of

this topic is nowadays very high and will become even more important in the next years. Reliable solutions are therefore needed to accomplish the task of providing privacy to OSN users through the users' education and the proposal and implementation of appropriate OSN architectures.

The first contribution of this thesis consists in an analysis of Online Social Networks that includes the main OSN actors, the OSN functionalities, the nature of the sensitive data shared by users, and the main threats resulting from potential misuse of such data.

The second contribution of this thesis consists in filling the lack of privacy preserving OSN by proposing a new decentralized architecture for OSN targeting user's privacy as the main goal. Decentralization is based on a new P2P system that *leverages the real life trust between OSN members* resulting from the OSN application as a natural cooperation enforcement mechanism *to build the social network application itself.* In the proposed solution, called ***Safebook***, neighbor peers are arranged according to their maintainers' real life trust that is, according to the social network graph. Nodes maintained by one user's friends store such user's data and serve it even when the user is off-line. As with anonymous routing, data requests and replies are recursively delegated to different peers to hide the actual requester's identifier and prevent the disclosure of the trust relationships between OSN members. Data confidentiality is assured by adoption of encryption techniques and integrity of profiles is assured by off-line trusted identification service(s) whose jurisdiction is limited to the purpose of identification only.

The Safebook architecture has been designed with the main goal of preserving user's privacy by the very beginning: profile integrity through adoption of certified identifiers that are signed by a trusted identification service, together with data confidentiality and integrity through adoption of classical encryption techniques, protect the social network graph vertices as represented in the OSN, i.e. the users'profiles; multi-hop routing of messages and further encryption techniques provide communication untraceability and confidentiality, and protects the social network graph edges as represented in the OSN, i.e. the user's contact list. Therefore, Safebook achieves ***privacy by design***.

The third contribution of this thesis consists in the evaluation of the feasibility and performance of Safebook. Starting from the online probability of peers, the number of user's friends, and the length of the hop-by-hop trusted paths providing communication untraceability, analytical models estimate the probability of retrieving a target user's data and the maximum size of each message containing such data.

The fourth contribution of this thesis consists in the investigation of the strong relationship between the topological properties of the social network graph and the achievable users' privacy in Online Social Networks. We observe three metrics, namely clustering coefficient, degree distribution and mixing time, and show that they give fundamental insights on the privacy degree of both centralized and distributed OSNs.

Further investigation is conducted on the impact of the social network graph topology on both the performance and privacy of Safebook. In Safebook there is a strong trade-off between performance and privacy because delay and reachability are inversely proportional to privacy. In fact, the lower the length of the hop-by-hop trusted paths, the higher the probability of deriving the friendship relationships between Safebook users. Nevertheless, the higher such length, the lower the probability of retrieving data, and the higher the retrieval delay. We observe that the optimal choice for this length depends on the social graph itself.

The fifth and last contribution of this thesis consists in the implementation and deployment of Safebook. The Safebook prototype is written in python and can be executed on multiple operating systems such as Windows, Linux and MacOs. A web based user interface helps the user to benefit from the available privacy tools such as those allowing her to share data with limitations.

1.3 Thesis organization

The first part of this thesis discusses the security and privacy issues in Online Social Networks.

Chapter 2 introduces Online Social Networks, provides details on the OSN actors such as the user and the provider, and illustrates the main functionalities of an OSN. Then, the core information stored in OSNs is identified and classified into several main areas.

Chapter 3 presents privacy, integrity and availability objectives for OSN. A detailed spectrum of attacks that may be perpetrated in OSNs is discussed against these objectives, and countermeasures are proposed to contrast such attacks. However, most of the countermeasures reveal to be ineffective against the Social Network Service provider itself, that plays the role of an omniscient centralized entity, a *"Big Brother"*. An overview of the main centralized OSNs is also provided.

Chapter 4 gives an overview of the solutions that researchers presented to contrast the "Big Brother" problem together with their limitations. Characterized by a decentralized approach through client-server, cloud or peer-to-peer architectures, these solutions mostly focus on the protection of the user's profile data rather than that one of the trust relationships between users.

The second part of this thesis introduces a new approach for privacy preserving Online Social Networks.

Chapter 5 motivates the need for a new privacy preserving OSN addressing the objectives of security and privacy presented in Chapter 3 and proposes a new decentralized approach for OSN achieving privacy by design. Such an approach, namely *Safebook*, targets decentralization and cooperation enforcement with the help of an ad-hoc peer-to-peer network mapping the real life social network graph. The trust relationships established in such an OSN are leveraged to build the OSN itself and provide data storage and communication obfuscation services.
Privacy against centralized omniscient entities is achieved thanks to the adoption of a decentralized P2P approach. Privacy against malicious users is achieved thanks to communication obfuscation through anonymous routing techniques, data confidentiality and integrity through the use of encryption, integrity of profiles'identity through certified identifiers.

Chapter 6 analyzes the feasibility of Safebook with the help of real network measurements. Online session times of peers and analytical models are taken as a basis to evaluate the probability of building trusted paths in Safebook and their residual lifetime. Therefore, data availability and the performance of data management operations are evaluated.

Chapter 7 introduces with an analysis of privacy from the graph theory perspective. The basic finding shows that three metrics, namely the degree, the clustering coefficient and the mixing time, give fundamental insights on the privacy degree of the OSN regardless of its particular centralized or distributed nature. The chapter further investigates the impact of

the social graph topology on the specific OSN architecture proposed in Safebook. Based on the findings presented in the first part, on analytical models and on some real social network dumps, the chapter shows a strong trade-off between performance and privacy such that delay and reachability are inversely proportional to privacy.

Chapter 8 presents the prototype of Safebook, an event-driven application composed by different managers in charge of building the overlays and running Safebook protocols. All managers communicate through a main dispatcher.Similarly to all current social network services, Safebook is accessible via internet browsers through a user interface implemented under the form of a web page.

Finally, Chapter 9 concludes this thesis and presents the future work.

Part I

Security and Privacy Issues in OSN

Chapter 2

Online Social Networks

This chapter introduces Online Social Networks. At the beginning, the chapter provides details on the OSN actors such as the user and the provider, and illustrates their relations. Then, the main functionalities of an OSN are discussed. Finally, the core information stored in OSNs is identified and classified into several main areas.

Social Network Services (SNS) are drastically revolutionizing the way people interact, thus becoming *de facto* a predominant service on the web, today. The impact of this paradigm shift on socioeconomic and technical aspects of collaboration and interaction is comparable to those caused by the deployment of the World Wide Web in the 1990's.

Catering for a broad range of users of all ages, and a vast difference in social, educational, and national background, SNS allow even users with limited technical skills to publish ***Personally Identifiable Information*** (PII) and to communicate with an extreme ease, sharing interests and activities.

An ***Online Social Network*** (OSN) offering, usually centralized, online accessible SNS contain digital representations of a subset of the relations that their users, both registered persons and institutions, entertain in the physical world. geared towards career management or business contacts; such networks typically provide SNS with a more serious image. In contrast, OSNs with a more private and leisure-oriented background are typically used for

sharing and exchanging more personal information, like, e.g., contact data, photographs, and videos; OSNs provided by such networks have usually a more youthful interface. The core OSN application is the creation and maintenance of *contact lists*. Through informing users automatically on profile changes of their contacts, the SNS help users to remain up to date with news of their contacts.

These properties of the SNS have led to the definition of boyd and Ellison [58], according to which *Social Network Sites* or *Online Social Network Services* are:

> *" ... web-based services that allow individuals to (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system".*

This definition, however, leaves aside some additional services that become apparent when observing the use of SNS. In particular, the communication of members through direct, sometimes instant message exchange, the annotation of profiles (e.g., via comments and recommendations), or the creation of links pointing to other profiles (picture tagging). The publication and browsing of images has grown to become a core function of these services [106]. Additionally, SNS typically provide support for a variety of third-party applications featuring advanced interactions between members ranging from simple "poking" of another member or the support for interest groups for a common topic to "likeness" testing with other members.

Maintenance and access to the OSN and their services are offered by commercial ***Social Network Providers*** (SNP), like Facebook[1], LinkedIn [2], Google[3], XING[4], and the likes. In general, a large amount of PII provided by the users is stored at the databases being under control of these providers, especially in the case of OSN targeting non-professional purposes. This data is either visible to the public, or, if the user is aware of privacy issues and able to use the settings of the respective SNS, it is accessible by selected group of other users. As profiles are attributed to presumably known persons from the real world, they are implicitly valued with the same trust as the assumed owner of the profile. Furthermore, any

[1] http://www.facebook.com
[2] http://www.linkedin.com
[3] https://plus.google.com
[4] http://www.xing.com

actions and interactions coupled to a profile are again attributed to the assumed owner of this profile, as well.
A SNP can, together with its SNS, also offer an application programming interface (API), allowing interested users to program a ***Social Network Application*** (SNA), thus extending and enhancing the functional range of the service.

2.1 Social Network Providers and Their Customers

Social network providers offer social networking services to the users and may further provide additional interfaces and services to other customers. These customers may come from different domains and pursue various goals.

In particular, ***sponsors*** belong to customers who advertise their services to the users through the OSN platform. Their advertisements may be of different kinds: plain commercial sponsors buy banner space or other marketing services from the SNP to advertise their products; SNS frequently contain "market pages" at which users can publish classified advertisements (ads), job offers, and the likes, for which they may be billed. Also sponsors may create commercial interest groups or profiles inside the OSN.

Another type of OSN customers are ***third party service providers***, who extend the content and functionality of SNS with their own applications. These applications such as quizzes and games are typically executed on the servers under control of these third parties connected to the SNS via appropriate APIs. Often these applications have extensive access to the personal data of OSN users.

Finally, all sorts of ***data analysts*** may act as customers of SNP. These customers typically have data mining interests and may also get access to the personal information of users and their activities within the OSN. The analysis carried out by data analysts may serve different purposes, including scientific research (such as statistics, social behavior, or network-relevant aspects) and non-scientific data mining, typically for commercial purpose such as marketing.

Figure 2.1 illustrates and summarizes the diversity of OSN customers and reflects their relationship to the SNS functionality and possible access to the personal information of the OSN users.

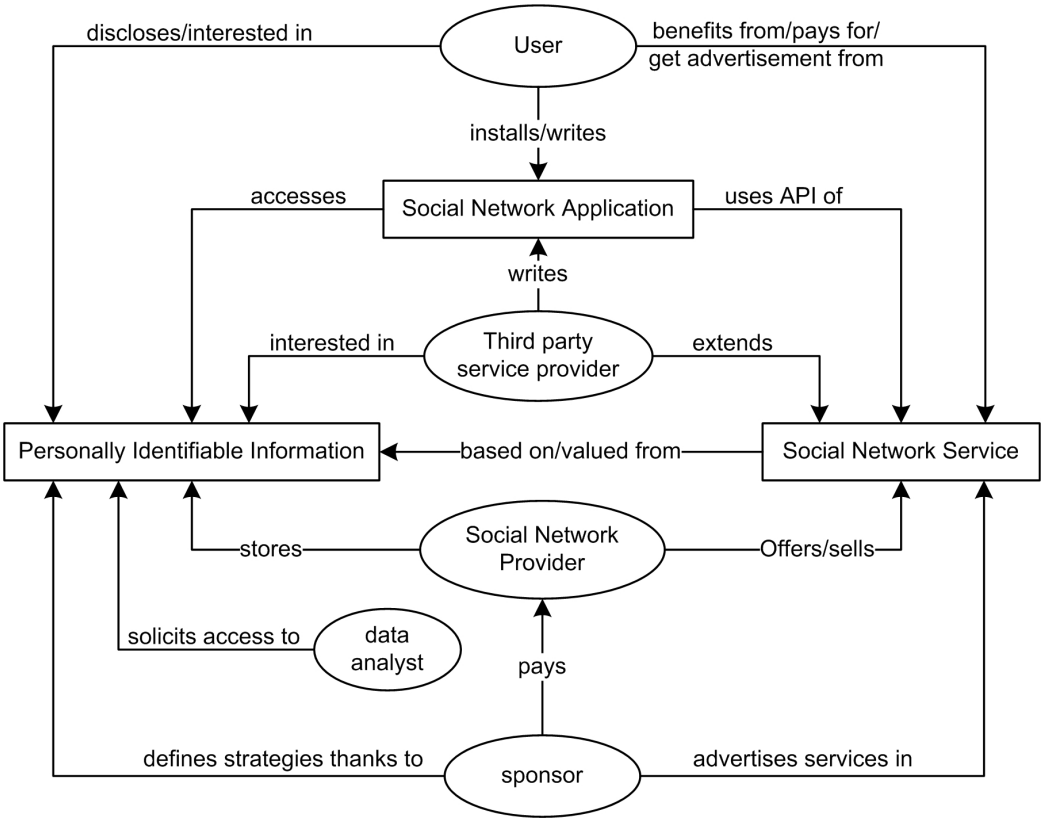

Figure 2.1: OSN customers and their relationships to PII and SNS

2.2 Functional Overview of Online Social Networks

Even though each OSN is usually tailored to some specific use, the functional range of these platforms is essentially quite similar. Generally speaking, OSN functionality can be classified into three main types: The ***networking functions*** serve the actual purpose of OSN to foster social relationships amongst users within the virtual platform. In particular, they provide functionality for building and maintaining the social network graph. The ***data functions*** are responsible for the management of user-provided content and communications amongst the users. Their variety contributes to the enhancement of users' interactions and makes the platform more attractive. Finally, the ***access control functions*** aim to implement the user-defined privacy measures and to restrict unauthorized access to the user-provided data and information.

2.2.1 Networking functions.

An OSN can be represented as a social graph whose vertexes are constituted by users and whose edges are constituted by social ties such as friendship, kinship and the like (see figure 2.2). OSN users can typically build their profiles and establish relationships with each other. The set of networking functions includes all functions that update the vertices and the edges of the social network graph. In particular, the OSN user invokes the *profile creation* function upon his or her registration to the OSN platform. This function adds a new vertex representing that user in the social network graph. Thereafter, with *profile lookup* the user can find other users who are also represented via vertices. Through the call to the *relationship link establishment* function the user can set up a new relationship with some other user. This function typically sends notification to that user, who in turn can accept or ignore the request. If the user accepts the request then users are added to the contact lists of each other and a new edge representing their relationship is added to the social network graph. The OSN users can also encounter profiles for possible relationships thanks to the *contact list browsing* function, which is realized through the traversal along the edges of the graph. Additional networking functions can be used to remove vertices and edges from the graph, for example upon the deletion of the user's profile.

2.2.2 Data functions.

OSN users can typically advertise themselves via their own profiles and communicate with each other using various applications like blogs, forums, polls, chats, and on-line galleries.The *profile update* function allows the OSN users to maintain details on their own profiles and provide fresh information to other users, who may call the *profile retrieval* function to visit the profile. Communication amongst users via blogs and forums is typically implemented through the *post* function, which inserts the message in the main profile page which sometimes is called the 'wall' or 'stream'. This block of information is not limited to plain text and can also contain videos, pictures, or hyperlinks. An OSN user willing to setup multimedia galleries typically calls the *upload* function, which transfers digital data from the user's device to the OSN database. In case of content depicting other users, the *tag* function can create a link pointing to their profile. OSN users can typically evaluate content published by other users through the *like* or *dislike* functions. These functions can also be considered as a feedback to the publisher from other users. In consequence, the user may either be

encouraged, or discouraged to provide similar uploads and posts. Using the *comment* function OSN users can articulate their point of view in a more explicit way. OSN users can also exchange personal messages. Here, in particular, the *write to* function simulates the asynchronous offline communication (e.g., e-mail), whereas the *chat to* function allows for the synchronous real-time communication. An OSN user can send messages to individuals and also to subgroups of users from his or her contact list. The latter subgroup can be defined via the *regroup* function. Additionally, users may *create* interest groups, *advertise* own interest groups to other users, and *join* interest groups created by other users. The user who creates an interest group obtains administrator rights for this group by default; however, these rights can be changed thereafter, and distributed to other group members.

2.2.3 Access control functions.

OSN users are usually allowed to define their own privacy settings through some access control functions. In particular, an OSN user may have control over the

- visibility of her on-line presence within the OSN;
- visibility of contacts from her contact list;
- visibility and access to her own profile information;
- access to her own uploaded content and posted communications.

All these functions usually take as input the information to be protected and the list of profiles having the rights to access it. The eligible profiles can be clustered into generic groups such as 'friends', 'friends of friends', 'everybody', or user-defined groups, such as 'family members', 'colleagues' or the like.

For example, the profile lookup function takes as an input a target's profile identifier, such as the name of the profile owner, and returns a list of possible candidates. An OSN user can apply output restrictions on this function to partially hide her own presence in the OSN. However, the protected profile would remain reachable due to the profile browsing functionality of the OSN. Nevertheless, sensitive relationships can be hidden from unauthorized users by imposing restrictions on the output of the contact list browsing function. Thus, combined with the restrictions on profile lookup, this constraint can completely hide some profile in the OSN, since this profile will become unreachable from other users outside of the profile's contact list. Note that new contacts could still be added to the profile owner's

contact list on the initiative of the latter. Another example is the control on the output of the profile retrieval function, which allows the profile owner to control the disclosure of the profile to other users. This allows some OSN user to hide parts of the private profile information from selected partners. Finally, the data related to online or offline indicators, one-to-one or one-to-many communications, such as posts, walls, comments, positive or negative marks, tags and the like can be protected by the means of restrictions on the huge set of the networking and data functions.

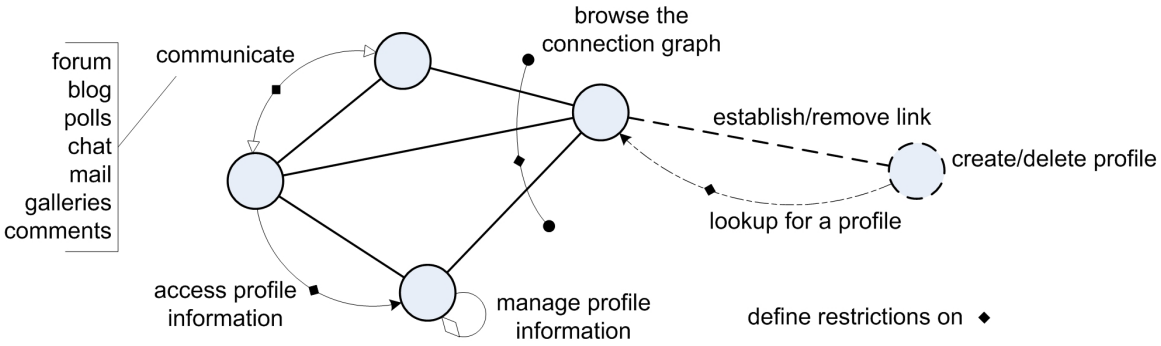

Figure 2.2: Main functionality of a typical OSN platform

2.3 Data contained in Online Social Networks

The core information stored in OSN, the self generated and maintained data of the users and their profiles can be classified into the following five types (see figure 2.3):

1. personal contact details, describing the user's identity;
2. connectivity, representing the connections in the social network graph;
3. interests of the user;
4. information on the curriculum vitae of the user;
5. communication, including all interactions with other OSN users of the SNS.

These types constitute the personally identifiable information which is provided directly by the OSN user. Additional information about the OSN user is often generated and made accessible within the OSN by other users.

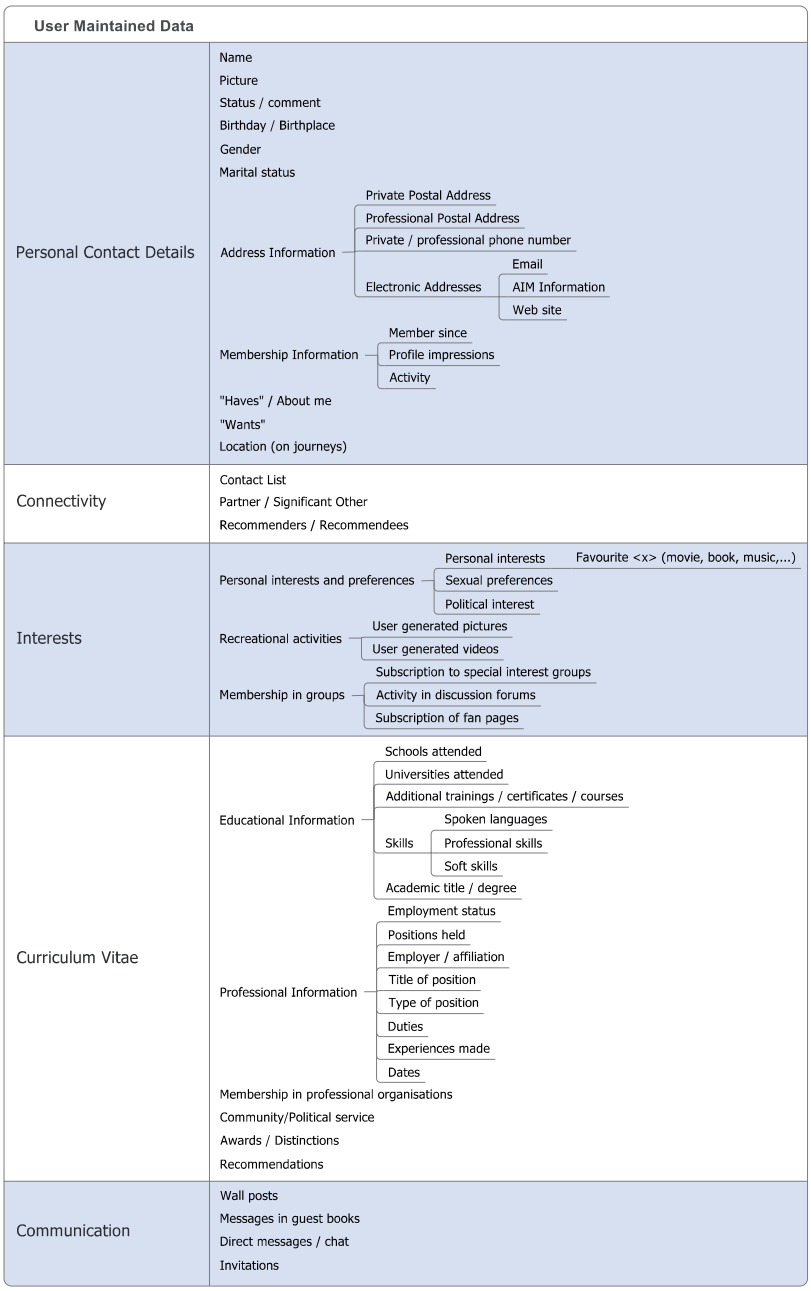

Figure 2.3: Types of data commonly stored in OSN profiles.

Personal contact details describe *'who the user is'*, providing not only some basic information such as the user's name, picture, gender, birthday, birthplace and marital status, but also some additional meta information with regards to the membership in the OSN, the

contact information aside of the OSN platform such as (e)mail addresses, phone numbers, instant messaging identifiers and personal web sites. Furthermore, it describes the personal profile of the user and may report about sexual, personal, political or religious interests and preferences. Users frequently can include a quick summary about themselves, describing their professional expertise, views and opinions, skills they "have to offer", as well as a short text on what they are looking for.

Connectivity describes *'whom the user knows'*, providing the user's contact list, possibly with annotated information about the type of the relationship (cf. family, colleagues, best friend, sports partner). Especially, OSN platforms with more private and leisure-oriented focus frequently ask the user to provide information on the relationship status, and in consequence the name and profile of their significant other contact. Users may further ask for recommendations by others. These recommendations may contain very detailed information about the user and shed light on the relationship between the both.

Interests describe *'what the user likes and is interested in'*. These may contain user's personal interests, hobbies, and preferences: In particular, information about favorite movies or music style, their sexual, religious, and political views, recreational activities of the user (such as personal pictures and videos showing situations from their personal lives), and their subscription to fan-pages as well as membership in special interest groups inside the OSN.

Information on the curriculum vitae describes the *professional career* and *educational background*, including attended schools, colleges, and universities, advanced studies, academic titles and professional certificates, as well as professional and soft skills. Such information may be very detailed and include the description of job positions the user currently holds or has previously had, including information on the duration and type of the position, the duties and responsibilities fulfilled in the job, and experiences being collected.

In addition to the description of the career progression, some OSN platforms ask the user to provide information on her membership in professional organizations (past and present), her community and political services (memberships and positions in clubs, associations, political parties, and professional societies), awards and distinctions, as well as recommendations and references.

Communication describes *'which messages the user has exchanged and with whom'*. OSN platforms generally offer exchange of personal offline messages, asynchronous communication via posts on walls and guestbook entries which the profile owner may hide or disclose to other users, and synchronous communication such as chats. These are examples of direct communications initiated by the user. However, there are also some less direct communications provided by other functionalities of the OSN platforms, such as the utilization of SNS applications (e.g. "poking", "likeness tests", quizzes), as well as public or targeted invitations to organized events.

Indirect information disclosure about OSN users may occur through posted opinions and comments, or any type of annotations to profiles of other users. Even though the owners of the annotated profiles may be able to remove undesired annotations, they need to notice the annotations in the first place. Since many users do not explicitly search for annotations made by other users about their profiles, this indirectly disclosed information may remain publicly accessible over a longer period of time. Similarly, information about users may be disclosed via third party statements about the user made in forums of the interest groups, or as annotations or comments at the profiles of other users.

Any form of user-generated digital content may also cause third party information disclosure. For example, some OSN try to prevent users from posting photographs showing people on their profiles if the owner of the profile is not depicted there[5]. However, this does not prevent users from posting photographs picturing them together with others. Additionally, many OSN platforms offer "tagging" of pictured users, whose profiles will usually be directly linked to that picture. These tags may contain further comments added by the user who uploads the picture.

2.4 Summary

In this chapter we presented Online Social Networks as digital representations of a social network graph whose vertices correspond to the registered users, and whose edges correspond to a subset of those users' relationships in real life. These OSNs are maintained by usually commercial social network service providers and allow users to easily share even sensitive

[5] `http://www.odnoklassniki.ru`

information, such as the user's personal contact details, her contact list, her interests, her professional and educational background, and her communication traces. Such data, often uniquely identifying a user, is stored at the databases being under control of the SNP. Potential misuse of this data from a malicious SNP or an attacker taking control on it may threaten users' privacy as discussed in the next chapter.

Chapter 3

Main threats in OSN

In this chapter we provide an overview of important security objectives for online social networks. First of all we notice that classical requirements (cf. [33]) of *confidentiality*, *integrity*, and *availability*, have a special touch when considered in the scope of OSNs. While integrity and availability have only subtle differences compared to other communication systems, in that they mostly address the content provided by the users, the requirement of confidentiality (usually associated with encryption) is no longer sufficient and should be extended to the more comprehensive security objective that is *privacy*.

While potential breach of user privacy and integrity of user-provided contents may lead to economic damages for the users, cause embarrassing situations, and also tarnish their reputation (even in the real world), the missing availability of contents or services may also decrease the attractiveness of the actual OSN platform and harm its provider. It is extremely difficult to cope with all these goals simultaneously. Especially privacy of OSN users is challenging since the amount of personal information is huge and this information may be available not only from a particular OSN platform but also from the web.

3.1 Security and privacy objectives

In the following, we describe privacy, integrity and availability objectives for on-line social networks, while also mentioning potential threats with regard to not only the profile owner, but also other users and the system itself.

3.1.1 Privacy

Privacy is a relatively new concept, born and evolving together with the capability of new technologies to share information. Conceived as 'the right to be left alone' [119] during the period of newspapers and photographs growth, privacy now refers to the ability of an individual to control and selectively disclose information about him.
The importance of privacy is so relevant to have been reported in the Universal Declaration of Human Rights (art.12):

> *"No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks."*

In the internet age, where huge amount of sensitive data can be easily gathered, stored, replicated and correlated, the protection of privacy is even more seen as the main objective for the services provided by an OSN platform [69, 59].

The problem of users'data privacy can be defined as the problem of *usage control* [93]: usage control ensures access control together with additional control on the later usage of the data, even once information has already been accessed.

Access to the content of a user profile may only be granted by the user directly, and this access control has to be as fine-grained as the profile itself. For example, if the profile contains several information blocks then access to each block has to be managed separately.

In addition, communication privacy calls for inference techniques aiming at deriving any type of information with regard to: (1) *anonymity*, meaning that users should access resources or services without disclosing their own identities; (2) *unobservability*, i.e. the requirement that no third party should gather any information about the communicating parties and the content of their communication; (3) *unlinkability*, which requires that obtaining two messages, no third party should be able to determine whether both messages were sent by the same sender, or to the same receiver; (4) *untraceability*, which demands

that no third party can build a history of actions performed by arbitrary users within the system; in other words, it demands both anonymity and unlinkability.

In summary, the objective of privacy is to hide any information about any user at any time, even to the extent of hiding their participation and activities within the OSN in the first place. Moreover, privacy has to be met by default, i.e. all information on all users and their actions has to be hidden from any other party internal or external to the system, unless explicitly disclosed by the users themselves.

3.1.2 Integrity

In OSN, any unauthorized modification or tampering of user-generated content and profile information have to be prevented (see figure 2.3). This encompasses the protection of real identity of users within the OSN platforms. In this sense, the definition of integrity in such networks is extended in comparison with the conventional detection of modification attempts on data. Moreover, problems with integrity of user profiles and their contents may have devastating impact on the objectives put forth with respect to the privacy of OSN users. Since the creation of profiles in traditional OSNs is easy, protection of real identities is insufficient in today's platforms. In particular, none of the current major OSN providers is able (and perhaps even not interested in) to ensure that a profile is associated to the corresponding individual from the real world.

As users inherently trust the OSN providers, the aforementioned vulnerabilities can be thwarted through the appropriate authentication procedures to assure the existence of real people behind registered OSN profiles. Identity checks do not necessarily have to be performed by a centralized service, however, all identification services have to be trusted by all participants.

3.1.3 Availability

The objective of availability for OSN aims at assuring the robustness of the social network services in the face of attacks and faults. The insufficient guarantees for availability may prevent users from accessing the service and make the OSN platform less attractive. Especially, for OSNs with professional focus, e.g. OSNs that aid their users to foster business relations or find new job positions, it is mandatory to keep users' data continuously available.

Therefore, we consider availability of user-generated data and profiles as a basic requirement that should be provided by the platforms, even though for leisure-oriented OSN platforms the availability of certain content may appear not of prime importance at first sight.

In the context of social network services ***denial-of-service*** attacks aim at either seizuring a victim's profile (or selected parts of it) or disrupting the possibility to communicate with the user. Such attacks have a direct impact on the availability of users' data. Furthermore, also integrity threats like data pollution and cloning may impair the availability of network services by affecting the quality of the service perceived by the users.

Also distributed services, which are implemented in a decentralized way, possibly via peer-to-peer systems, or which follow other types of service delegation, may be vulnerable to a series of attacks against availability as well. These attacks include *black holes*, aiming at collecting and discarding a huge amount of messages; *selective forwarding*, where some traffic is forwarded to the destination, but the majority is discarded; and *misrouting*, which aims to increase the latency of the system or to collect statistics on the network behavior. In any case, attacks on distributed social networks are more effective in case of *collusion* amongst malicious users or in the presence of Sybil nodes controlled by the attacker, which is not the case for the centralized OSNs.

In the following, we introduce and discuss the impact of a series of OSN attacks on the above presented security objectives.

	Security Objectives		
	Privacy	Integrity	Availability
Attacks			
Plain Impersonation	x	x	
Profile Cloning	x	x	
Profile Hijacking	x	x	
Profile Porting	x	x	
Id Theft	x	x	x
Profiling	x		
Secondary Data Collection	x		
Fake Requests	x		
Crawling and Harvesting	x		
Image Retrieval and Analysis	x		
Communication Tracking	x		
Fake Profiles and Sybil Attacks		x	
Group Metamorphosis		x	
Ballot Stuffing and Defamation		x	
Censorship		x	x
Collusion Attacks	x	x	x

Table 3.1: Attacks vs. Security Objectives in Online Social Networks

3.2 Attack Spectrum and Countermeasures

The diversity of available OSN platforms opens doors for a variety of attacks on privacy of the users, integrity of their profiles, and the availability of the user-provided contents. In this section, we will highlight main attack types against OSN platforms and discuss their impact on the aimed security objectives. Table 3.1 will serve as a background for our discussion. It illustrates different types of attacks and shows their relevance for the mentioned security objectives of privacy, integrity, and availability. We will discuss not only the purpose and

impact of each attack but also explain the techniques needed to mount it, while referring to some real-world examples, where possible. We note, however, that technical realization behind an attack may strongly depend on the functionality and in particular on the use of different protection mechanisms within the OSN platform. Therefore, not every attack technique will have the same impact when used against different OSN platforms. Moreover, since OSN providers typically have full control over the network resources, no meaningful protection appears possible if the attacks are mounted by the provider itself.

Plain Impersonation With *plain impersonation* attack the adversary aims to create fake profiles for real-world users as depicted in figure 3.1. In this sense a real-world user will be impersonated within the OSN platform. The success of this attack strongly depends on the authentication mechanisms deployed in the registration process. Since many OSNs tend to authenticate email addresses by requesting confirmations for the registration emails, this attack can be easily performed if an email address is created in advance. The consequence of plain impersonation is that the adversary can participate in the OSN applications on behalf of the impersonated user with all damaging consequences for the user. A currently very prominent secondary effect of all kinds of impersonation (Sections 3.2 – 3.2) is the misuse of the trust that users inherently have in messages from their accepted contacts, and especially the '419' scam [1]: impersonating attackers engage in a dialog with contacts of the impersonated individual, and, by producing a credible story, ('My wallet was stolen in London and now I can't pay my flight home') successfully defraud the victim. This attack can be thwarted only through the deployment of stronger authentication techniques. In particular, it is desirable to require some form of real-world identification from the user prior to switching on her account.

Profile Cloning By *profile cloning* we understand a special type of impersonation attack that occurs within the same OSN platform [39], as depicted in figure 3.1. The goal of the adversary here is to create a profile for some user that is already in possession of some valid profile in the same network. From the technical point of view this attack can be realized through the registration of the new profile using the same (or similar) content as the existing one. This is feasible in most OSN platforms since each profile is associated with some unique administrative id and an email address used during the registration. Furthermore, users can hide their email address so that OSN users would not be able to distinguish between the original profiles and their clones registered with other email addresses. As a consequence the

adversary can create confusion through impersonation of other registered users and possibly gain access to the private information communicated to that users. Moreover, with tools like iCloner [39] profile cloning can be automated. Such tools are able to collect public data of OSNs members, match them, create cloned profiles and then send friendship requests on their behalf. A possible solution for OSN providers to prevent profile cloning is to deploy mechanisms that are able to detect similarities between different profiles, in particular with regard to the personal information that is visible to the OSN users. Since cloned profiles typically have later registration date than the original ones, it should be feasible for the OSN provider to distinguish them and remove from the network.

Profile Hijacking The goal of the adversary mounting a *profile hijacking* attack is to obtain control over some existing profile within an OSN platform. Many OSN platforms protect user access to their own profiles via passwords. Hence, from the technical point of view profile hijacking is successful if the adversary can obtain passwords of other users. This can be done by many means. First, it is a well-known fact that the majority of users choose weak passwords that can be recovered via an automated dictionary attack [64]. However, OSN providers typically deploy protection against such attacks by restricting the number of login attempts or by using techniques that require human interaction such as CAPTCHAs [118]. Nevertheless, there exist effective tools, e.g. as the one included in iCloner [39], that are able to analyze and bypass CAPTCHAs. Alternatively, the adversary may try to obtain passwords via social-engineering attacks such as phishing [73], or obtaining passwords for other online services, relying on the fact that most people use the same passwords across the majority of their accounts at different sites. The OSN functionality can be misused to distribute messages aiming to lure users to fake login websites [5]. Finally, we shouldn't forget that OSN providers themselves have full control over the registered profiles. Therefore, if some profile appears attractive for the OSN provider to be hijacked the password access to the profile can be changed accordingly.

Profile Porting By *profile porting* we understand another type of impersonation where some profile that exists within one OSN platform is cloned into another OSN platform [73, 39], as depicted in figure 3.1. From the technical point of view this attack can be realized via registration of a profile using some new email address. Profile porting is appealing since not every user has her own profile on every available OSN platform. On the other hand, there might be some users that participate in both OSN platforms and thus will not be

able to distinguish amongst ported profiles. The significance of profile porting (e.g. in comparison to profile cloning) is that users may be completely unaware that their profiles have been ported. The impact of profile porting is that the adversary can impersonate users in different OSN platforms. Thwarting profile porting is not that easy. In particular, profile similarity detection tools can still be used but only if they can work across multiple OSN platforms. Since every OSN platform is administrated by a different provider, the deployment of such tools would require cooperation amongst the providers. This is difficult to achieve, since OSN providers are cautious about granting any form of access to their profile database to competitors.

ID Theft Under *ID theft* we consider the impersonation of OSN users in the real-world [39], as depicted by the example of user $\mathcal{A}$ impersonating user $\mathcal{Z}$ in figure 3.1. An adversary mounting the ID theft attack should be able to convince anyone about the ownership of some particular OSN profile. In this way, the adversary can possibly misuse the reputation or expertise of the real profile owner for own benefit, while leaving the owner unaware of the attack. One way for a successful ID theft attack is to take control over the target profile. This requires the same effort as for the profile hijacking attack. However, this effort seems necessary only if the adversary has to actively use the profile for the ID theft attack, e.g. communicate via the OSN platform. Often it would simply suffice to claim the ownership of a profile and perform the actual communication via other channels. In this case, thwarting ID theft attacks by technical means seems impossible. The only solution is to rely on other means of real-world identification such as national identity cards, driver's licenses, etc.

Profiling In addition to the maintenance of own profiles modern OSNs provide users with various applications to express themselves via forums, guest books, discussions, polls, multimedia data, etc. These activities are observable by other users within the OSN platform. By *profiling* we understand an attack against any target OSN user aiming to collect information about OSN activities or further attributes of that user, e.g [36], see also figure 3.2. This attack can be typically performed by OSN users, possibly in an automated way, since the collectable information is usually publicly accessible by all OSN users. The risk of profiling attacks performed by OSN users can be diminished via fine-grained access control and anonymizing techniques. For example, users should be able to allow access to the personal parts of their profile on the individual basis and not only based on roles (e.g. friends) as realized in many current OSN platforms. However, recent studies, e.g. [89], show that even

if the personal information is hidden, it can still be inferred from public information and social activities of the user. An alternative solution could be to let users decide whether their activities (e.g. discussion comments) should be kept unlinkable to their profiles. Although these measures may help to reduce the risk of profiling performed by other OSN users, preventing potential profiling performed by OSN providers [21] appears to be much more difficult.

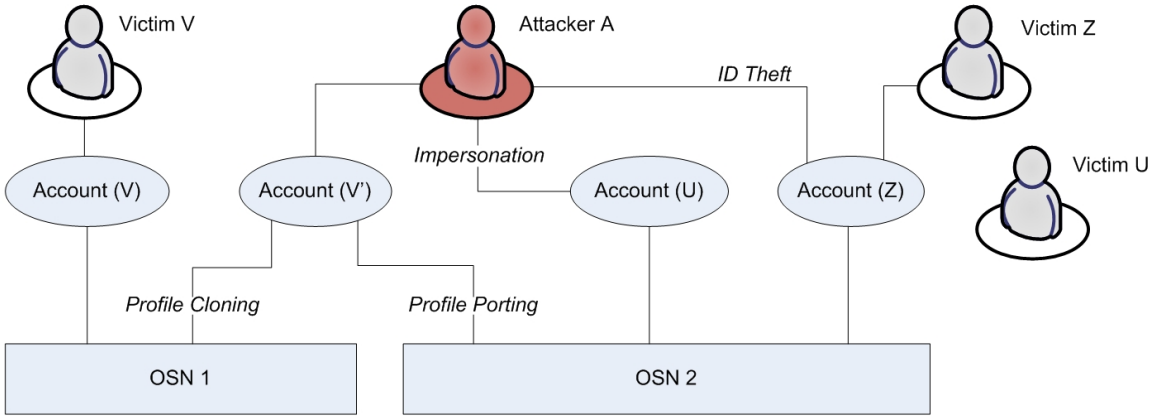

Figure 3.1: Impersonation attacks: victim $\mathcal{U}$ doesn't have any OSN account, victim $\mathcal{V}$ has an account on OSN1 and victim $\mathcal{Z}$ on OSN2. The attacker $\mathcal{A}$ generates $\mathcal{U}$'s account on OSN2, a copy of $\mathcal{V}$'s account on OSN1 and OSN2, and logs on OSN2 with the credentials of $\mathcal{Z}$.

Secondary Data Collection By *secondary data collection* we understand an attack that aims to collect information about the owner of some OSN profile via secondary sources apart of the OSN platform as depicted in figure 3.2. A typical example of secondary data collection is to use some Internet search engine to find information that can be linked to the profile owner. More effective is to use some Internet service[1] that aggregates all information it can find about some particular person. Through such an attack the adversary may obtain much more information about some user than available in the profile and misuse it against the user both in the virtual environment of the OSN platform and in the real life. Another example are recent de-anonymization attacks [121] that misused the group memberships of social network users for their unique identification. Furthermore, the existence of OSNs with public and private profiles simplifies the secondary data collection as many users tend to

[1] `http://www.123people.com/`

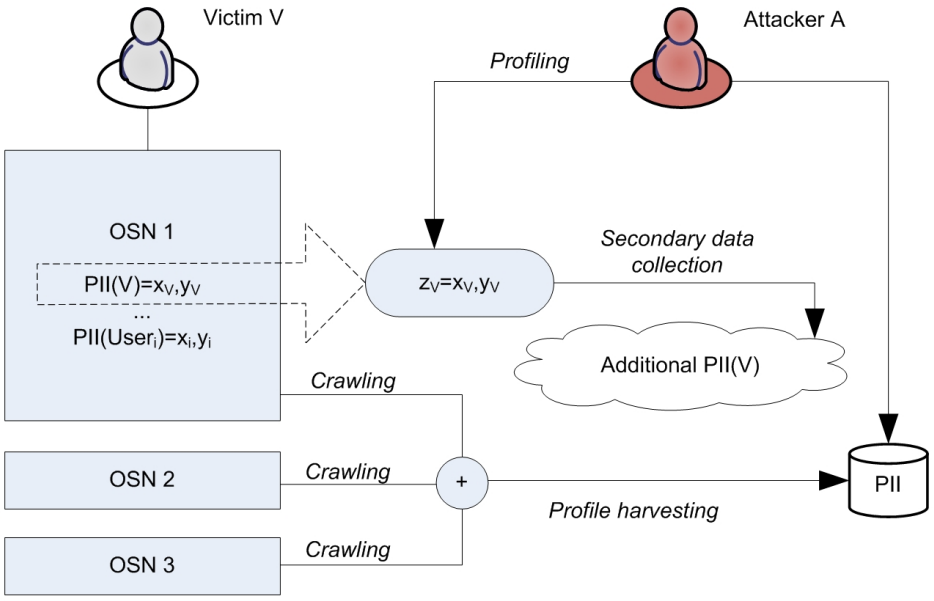

Figure 3.2: Main PII related threats in current OSNs.

have accounts on different platforms [126]. There is no meaningful protection against secondary data collection attacks since the data is typically aggregated from different locations. Therefore, it appears in responsibility of the user to limit information kept in the profile in order to avoid its linkability with secondary sources.

Fake Requests One of the main objectives of OSN platforms is to establish social contacts. This proceeds via connection requests that can be either accepted or rejected by the users. An adversary with a OSN profile that sends *fake requests* to other users aims less on the social contact with these users but is more interested to expand its own network. The dissemination of fake requests can be automated. Since many OSN users tend to accept fake requests [39], the adversary can simplify access to their profiles and activities and possibly obtain additional information whose visibility is subject to the available direct or nth-grade connections. These connections can then be misused for the automated collection and aggregation of information. The actual dissemination of fake requests cannot be prevented since establishment of new connections is an important goal of OSN applications. Therefore, it is desirable that users behave more responsibly upon accepting new connection requests.

Crawling and Harvesting The goal of *crawling* is to collect and aggregate publicly available information across multiple OSN profiles and applications in an automated way

[39, 36]; see also figure 3.2. Unlike profiling this attack does not target any particular user and unlike secondary data collection it is executed within the OSN environment. The expansion of own network connections by the adversary using fake requests can be seen as a preliminary step for crawling. The adversary is simply interested in collecting as much public information within the OSN platform as possible. This information can then be misused for different purposes, for example for selling data to marketing agencies, etc. Also it would allow for the offline analysis of social relationships and user activities, thus paving the way for targeted attacks on OSN users. Although some OSN platforms try to protect from crawling through the deployment of CAPTCHAs, the latter can be passed over with the appropriate solving tools [39]. Another attack by which the adversary simultaneously crawls across different OSN platforms is called *harvesting*. Typically harvesting results in larger datasets with larger amount on private information about the OSN users.

Image Retrieval and Analysis Upload of images or other digital content and its discussion stimulates social interactions of OSN users. However, free accessibility to images and videos bear potential risks to the privacy of users. By *image retrieval and analysis* we understand an automated attack aiming to collect multimedia data (incl. images, videos, etc.) available with the OSN platform. This attack is typically followed by the subsequent analysis via automated pattern recognition tools (see e.g. [125] for a survey on face recognition) to find links to the OSN profiles of displayed users. Information distilled in this way can reveal more private information about users than they are willing to give. In particular, it may reveal information about friends or colleagues that are not necessarily part of the user's social network, or information about visited locations (location-tracking) shown on the photographs. The analysis of digital content can be further strengthened by considering secondary sources such as search over the Internet. Digital content retrieval attacks can be possibly thwarted through a more restrictive access control policies for the digital content.

Communication Tracking OSN users communicate with each other using diverse OSN applications. By *communication tracking* we understand a profiling attack aiming to reveal information about communications of the same user. In this way the attacker may collect more information about the user than available in the profile. This attack can be mounted in an automated way by searching for comments left by the target user in various OSN applications.

Fake Profiles and Sybil Attacks In many OSN platforms users can easily create several profiles under possibly different identities and contents. Since many OSN platforms lack of proper authentication such creation of *fake profiles* becomes easy [39]. On the technical side, the user has only to create a new email for the registration of a fake account. Fake profiles pave the way for *Sybil attacks* [62] that may serve different purposes [80, 8]. For example, owners of fake profiles can establish new connections without disclosing their real identities. In this way they may obtain more information about some person than by using some real account. Sybil account may also be created on behalf of the whole groups [4]. Furthermore, Sybil accounts can be misused against the functionality of the OSN platforms. This includes distribution of spam messages [9] or other illicit content such as malware [10] and phishing links [6, 29], illegal advertisement, bias of deployed reputation systems, etc. Creation of fake profiles can be seen as a special form of impersonation attacks. One solution for OSN providers to recognize fake profiles is to use IP traceback. Indeed, if logins to several profiles come from the same IP address then it is likely that some of these profiles are fake. However, an attacker may try to avoid IP traceback by using different proxies. Therefore, stronger identification and authentication mechanisms for admission of new users would offer a better protection.

Group Metamorphosis A popular application provided by OSN platforms is the establishment of shared interest groups. These groups are usually administrated by OSN users and provide a platform for more focused discussions, specialized contact establishment, and dissemination of information, which may be interesting for a targeted audience. By *group metamorphosis* we understand an attack where group administrators change the group subject to persuade own interests, e.g. political[2]. Other OSN users who joined the group earlier may remain unaware of this change, which in turn may have negative impact on their reputation. A possible solution for OSN providers to thwart group metamorphosis attacks is to restrict control of administrators over the interest groups, in particular to prevent them from modifying any information that may have impact on the group as a whole.

Ballot Stuffing and Defamation OSN platforms serve primarily the contact establishment and interaction amongst users. Hence, attacks biasing public perception and recognition of a target OSN user by others are undesirable. By *ballot stuffing* we understand an

[2]One incident has been reported for facebook, where a multitude of groups have been fostered under general topics and concertedly renamed to support Silvio Berlusconi, in 2009 `http://www.repubblica.it/2009/12/sezioni/politica/giustizia-21/gruppi-facebook/gruppi-facebook.html`

attack by which the attacker wishes to increase public interest to some target OSN user. This attack may increase the amount of personal messages or connection requests received by the target user resulting in a DoS attack on the physical resources of the OSN user. The attack may place the victim into the focus of public, possibly embarrassing discussions. On the other hand, ballot stuffing may increase popularity of the profile belonging to the attacker. This can be achieved through recommendations submitted by the attacker using fake profiles. In contrast, *defamation attacks* aim at decreasing public interest of a target user, in particular by tarnishing the reputation of the latter [23]. In particular, defamation may lead to blacklisting of the user in contact lists of other users and keep the user away from participation in communication applications such as shared interest groups and discussion forums. It may further have negative impact on the user's life in the real world [19]. Another form of defamation is the anti-advertising against companies [22] aiming to damage the reputation of the latter on the market.

Both ballot stuffing and defamation attacks have to be performed at a large scale in order to have a significant impact. An attacker may create fake profiles and use automated tools to disseminate information needed to increase or decrease interest to a specific OSN user. Another technique is to use the poll application provided by many OSN platforms and let users vote on information related to the victim.

Censorship OSN providers typically have control over the whole data available within the network. As such they can deliberately manipulate the user-provided information and contents. In some cases this ability is necessary to prevent dissemination of illicit content. On the other hand, *censorship* when applied without substantial reasons may have negative impact on the OSN users. For example, in OSN platforms focusing on business contacts users often advertise their expertise. In this scenario censorship may be misused to favor some users over their competitors. Censorship may have many facets. It can be performed by active modification of user-provided contents, which might remain unnoticed by the user. Higher impact can be achieved through the target manipulation of search engines within the network. Since censorship can be performed by the OSN provider [20] without involving any other parties, there is little one can do to prevent this threat. Censorship may be applied not only by OSN providers but also by administrators of shared interest groups. They can deliberately modify or drop messages of group members. Although restricting group administrators from modification of other user contents appears to be an effective protection measure, it is unlikely to be used in practice, since this ability contradicts to the

responsibility of group administrators for the content disseminated within the group.

Collusion Attacks The "impact of a crowd" can be exhibited in OSNs through a *collusion* of users. In this attack several users join their malicious activities in order to damage other OSN users or mount attacks against applications of the OSN platform. In particular, colluding users may start defamation or ballot stuffing campaigns, increase each over reputations, bias the outcome of public polls or influence public discussions. Since colluding users have valid OSN profiles these attacks do not require creation of fake profiles. Furthermore, these attacks are more difficult to recognize than similar attacks mounted via fake profiles. The reason is that IP traceback would not help even if colluding users do not deploy any additional proxies.

The analysis of the privacy problems in current OSN demonstrates that even if all participants were aware and competent in the use of SNS, and even if a concise set of privacy measures were deployed, the OSN would still be exposed to potential privacy violations by either the omniscient service provider or an external attacker taking control of the OSN [74, 26].

3.3 The "Big Brother" problem

The complete PII, directly or indirectly supplied by all participants, is collected and stored permanently at the databases of the providing company, which potentially becomes a "Big Brother" capable of exploiting this data in many ways that can violate the privacy of individual users or user groups.

The importance of this privacy exposure is underlined by multiple factors. First of all, according to a recent study from comScore [51], one every five minutes spent on-line is spent in browsing social networking sites, that nowadays reach 82% of the overall on-line population. Secondly, SNS providers make profit through displayed advertisements: emarketer evaluates the worldwide social network advertisement revenue will hit 8 billion US$ in 2012 and 10 billion US$ in 2013 [25]. Finally, the market capitalization of SNS providers was able to reach up to 50 billion US$ as in the case of Facebook Inc, according to the 1.5 billion US$ funding led by Goldman Sachs Group Inc. in January 2011 [12].

In the following, privacy policy aspects of well known OSNs are briefly introduced. The main characteristics of such OSNs are reported in table 3.2.

Facebook Appeared in 2004 as a service accessible by Harvard students only, Facebook reaches now the 55% of the world global audience and accounts for one every seven minutes spent on-line [51]. Owned by Facebook Inc., its value has been always increasing and nowadays is worth 50 billion US$ (according to the investment of Goldman Sachs and Digital Sky Technologies in 2011 [12]).

When a user creates an account in Facebook, according to the terms of service [7], she provides the required information consisting on name, email address, birthday, and gender. While user's name, profile picture, networks, username and User ID are made public, all remaining user generated information can be shared with audience limitations.

Facebook is granted *'a non-exclusive, transferable, sub-licensable, royalty-free, worldwide*

license to use any IP content' the user posted (IP License). Such a license ends when the user deletes this content, that persists at the SNS provider in backup copies *'for a reasonable period of time'*, *'unless it has been shared with others'*.

User has the right to create a single personal profile and guarantees she will provide true information. When Facebook provides this information to advertising partners or customers, PII is always removed. In contrast, the user cannot exploit her own information for personal gain.

Content infringing someone else's copyright can be removed by Facebook. In this case, the censored user is provided with the opportunity to appeal.

When user's friends upload pictures showing the user, automatic face recognition suggests the user's name for the tag by default. The user can opt-out from this service in her privacy control panel accessible from her Facebook profile homepage.

Twitter Twitter is a microblogging platform allowing users to send 140 characters long messages, also known as *Tweets*. Launched in July 2006, Twitter now has 100 million users and is valued at 8 billion US$ (as of October 2011) [27].

Terms of Service [28] specify tweets are public, and limits on use and storage may be applied at any time without prior notice. However, Twitter also gives its users the opportunity to limit the access on their tweets to people whom they approve.

'The Services may include advertisements, which may be targeted to the Content or information on the Services, queries made through the Services, or other information'.

By submitting, posting or displaying Content on or through the Services, the user grants Twitter a *'worldwide, non-exclusive, royalty-free license (with the right to sublicense) to use, copy, reproduce, process, adapt, modify, publish, transmit, display and distribute such Content in any and all media or distribution methods'*.

Twitter reserves the *'right at all times (but will not have an obligation) to remove or refuse to distribute any Content on the Services and to terminate users or reclaim usernames'*.

While non personal information may be shared or disclosed, in the event that Twitter is involved in a bankruptcy, merger, acquisition, reorganization or sale of assets, the user information *'may be sold or transferred as part of that transaction'*.

Process of account deletion starts after 30 days from the reception of the communication from the user, and may take up to a week.

LinkedIn With the mission of *'connect the world's professionals to enable them to be more*

productive and successful', LinkedIn is mainly devoted to business-related social networking. This OSN was launched in March 2003 and accounts 135 million users as of November 2011 [16] for a value of 8 billion US$ (May 2011) [18].

To create an account, according to the User Agreement [17], the user should not be a competitor of LinkedIn, nor use the service for reasons that are in competition with the OSN. User agrees in providing accurate information and update it as necessary, and has to avoid transferring her account to another party.

A LinkedIn user grants the SNP *'a nonexclusive, irrevocable, worldwide, perpetual, unlimited, assignable, sublicenseable, fully paid up and royalty-free right' 'to copy, prepare derivative works of, improve, distribute, publish, remove, retain, add, process, analyze, use and commercialize, in any way now known or in the future discovered, any information '* the user provides *' directly or indirectly to LinkedIn, including, but not limited to, any user generated content, ideas, concepts, techniques or data to the services' 'without any further consent, notice and/or compensation'* [17].

User's data can be deleted at any time, unless the user shared information or content with others and they have not deleted it, or it was copied or stored by other users. Moreover, user information can be provided in response to customer service inquiries, to send service or promotional communications through email and notices on the LinkedIn website, or to create social ads for the user's network on LinkedIn using the user's profile photo and name.

Google+ Launched in July 2011, Google+ reached 25 million unique visitors in just less than a month, faster than any other OSN in history [51]. With 90 million users as of December 2011 [13], this OSN is the last essay of its owner Google Inc. whose market value as of the beginning of February 2012 is 155.47 B according to Yahoo [31], to become a competitor in the OSN market.

Google+ users agree on the new privacy policy effective since the 1st of March 2012 [15]. This policy explicits Google *'may combine personal information from one service with information, including personal information, from other Google services'*. The user is asked to quickly update wrong personal information or delete it. Deletion can be propagated to active servers with some latency and may not be applied to data stored in backup systems. User personal information may be shared with trusted companies, organizations or individuals outside Google whose role is to process the information for Google itself. Additionally, aggregated non-personally identifiable information may be shared publicly or with partners like publishers, advertisers or connected sites.

OSNs	Characteristics		
	SNP	Unique visitors (million)	Market value (billion US$)
Facebook	Facebook Inc.	845 (dec 2011)	50 (Jan 2011)
Twitter	Twitter Inc.	100 (Oct 2011)	8 (Oct 2011)
LinkedIn	LinkedIn Corporation	135 (Nov 2011)	8 (May 2011)
Google Plus	Google Inc.	90 (Dec 2011)	155 (Feb 2012)

Table 3.2: Current OSN and their characteristics.

The privacy policy specific to Google+ [14] tells the users *'need to have a public Google Profile visible to the world, which at a minimum includes the name chosen for the profile. That name will be used across Google services and in some cases it may replace another name used when sharing content under the Google Account'*. Google Profile identity may be shown to people who have the user's email address or other identifying information.

Users may define groups or *circles* of people to share information with. According to the default settings, people in circles except the name of the circle will appear to others.

When uploading photos or videos in Google+ the user aiming at hiding the metadata information associated to such a content needs to remove it before the upload. Automatic face recognition is provided as an opt-in functionality and makes easier for the user to tag her contacts in the picture, that remains however a non automated action since the user needs to validate each tag.

3.4 Summary

In this chapter we defined security requirements for OSNs and investigated their main security and privacy issues. We showed they derive, on one hand, from the user's lack of awareness on the consequences of simple actions such as accepting a friend request, and, on the other hand, from the usability of the privacy controls offered by SNS.

However, even if the OSN provided a satisfying set of privacy tools to privacy aware and competent users, the directly or indirectly high valuable shared user's data could still be exploited by the omniscient service provider, as often stated in the subscribed terms of service.

Even considering the commercial bodies that act as social network service providers to be trusted, hackers may be able to compromise their systems to gain access, unsatisfied employees may abuse their access to the data, or even imprudent publication of seemingly anonymized data may lead to the disclosure of PII.

Researchers realized the importance of this privacy exposure and proposed a set of countermeasures addressing the basis of the problem: the centralized storage of users' data. Their solutions are examined in the next chapter.

Chapter 4

Decentralized OSN

In this chapter we give an overview of the solutions that researchers presented to contrast the "Big Brother" problem together with their limitations. Characterized by a decentralized approach through client-server, cloud or peer-to-peer architectures, these solutions propose to store all users' data in a distributed fashion.

The "Big Brother" problem intrinsically affects all the centralized OSNs. In the last years, several solutions [94] have been proposed with the goal of preventing the presence of any omniscient entity. These solutions, known as ***Decentralized Online Social Networks*** (DOSNs) [65] aim at distributing the user generated contents: in all of them the users' data is made available from multiple locations. Access restriction on such sensitive data is often provided with the adoption of encryption techniques or access control lists.

Current DOSNs can be divided into two main groups:

- ***Client-Server based*** decentralized OSNs;
- ***P2P based*** OSNs.

While in Client-Server based DOSNs every user controls one or more (at least logically) centralized computing and storage services running in a real or virtual infrastructure, in

P2P-based DOSNs every user node joins a well known or dedicated P2P network where computing and storage resources are shared among members.

These two categories are discussed further in this chapter.

4.1 Client-Server based Decentralized OSNs

Distributed dedicated server approaches require acquisition or deployment of web space hosting users' sensitive data whose access is restricted to authorized users only.
At the benefit of guaranteeing full data availability, they often require the OSN user to pay for the storage service, or for the maintenance of proprietary infrastructure. When third party storage service is provided for free, these solutions often lack incentive mechanisms guaranteeing the service reliability.

Yeung et al. [124] propose a framework allowing users to choose one or more trusted servers to host several resources, each of them identified by a URI, such as their activity log, their photo album, and, most importantly, their Friend-Of-A-Friend (FOAF)[1] information, that can be edited through open protocols such as WebDAV[2]. In this framework, users obtain an identity in the form of a URI (a Web ID) pointing to a reference in the user's FOAF file stored on a trusted server, that, in turns, points to their contacts' Web ID. Policy languages such as AIR [75] allow publishers to restrict access to their data, and protocols such as OpenID [99][3] allow requesters to authenticate and access it.
Similarly, in ***Diaspora***[4] several servers called *pods* host users'accounts, or *seeds* . Pods can be run by users or institutions and together form the social network service infrastructure. Newcomers unable to setup their own pod nor to find place in another user or institution pod may host their profile in one of the open pods[5]. Every user generated content is encrypted with a random key in turn distributed to every authorized user.
In ***Vis-'a-Vis*** [107] users store their sensitive data on a paid virtualized cloud-computing infrastructure, such as the Amazon Elastic Compute Cloud (EC2). The infrastructure is assumed to support a Trusted Platform Module (TPM) proving the customer what software is executing under their account. Vis-'a-Vis is designed to interoperate with existing OSNs

[1] http://xmlns.com/foaf/spec/
[2] http://www.webdav.org
[3] http://openid.net/specs/openid-authentication-1_1.html
[4] http://www.joindiaspora.com
[5] http://podupti.me/

rather than replace them. Existing OSNs are treated as untrusted services storing opaque pointers to the user's data in the cloud while compute utility has access to cleartext data.

In ***Persona*** [35] each user is identified using a single public key and stores his encrypted data with a trusted storage service. Public keys and storage service location are exchanged out of band at the act of friendship establishment. Users interact and publish references to their data through Persona applications, providing a set of APIs over which social network facilities like wall posting or profile publishing operate. Access to user data is controlled through Attribute Based Encryption (ABE), and traditional public key cryptography. The attributes a user has determines what data they can access: private user data is always encrypted with a symmetric key, that is in turn encrypted with an ABE key corresponding to the group that is allowed to read this data.

Lockr [114] decouples OSN social information such as the user's published data from functionality such as the social network facilities. Users do not further need to reveal a full copy of their social network to every OSN they use, and may decide which OSN provider or storage service can store their sensitive data, and which third party can access it. Users may also decide to store their data themselves. In Lockr, identities are represented by a public/private keypair, while address books by a list of public keys associated to the user's contacts. Access control policy on the user published data is provided through social attestations: digitally signed metadata encapsulating a social relationship. Social attestation proof of ownership is performed with WHPOK [67], a variant of zero-knowledge protocols, so that the digitally signed attestation is never revealed. Differently from previous solutions, Lockr can also rely on P2P systems such as BitTorrent to verify the attestations and deliver content. In this approach, signed torrent files specify the relationship that downloading peer must have with the torrent's owner.

4.2 P2P-based Decentralized OSNs

In P2P-based approaches OSN members also participate in the setup of a P2P overlay and share data storage and computing facilities. Due to the on-line behavior of peers, these approaches inherently relax the data availability requirements and provide best-effort services. At the benefit of no server acquisition or maintenance costs, these approaches often leverage on existing P2P overlay architectures originally conceived for the purpose of file sharing.

On the other hand, *profile sharing* asks for different requirements. In the context of file sharing, few large data objects have to be reliably distributed among requesting peers that in turn distribute the same object again. Noticeably, when appropriate incentive mechanisms enhance the collaboration of peers, the popularity of the content determines its availability. In the absence of such mechanisms, on the contrary, peers often engage in *free riding* [32, 108, 82] due to their inherent selfishness: they immediately disconnect from the network and not even popular content can be made highly available.
Additionally, access to file objects in file sharing P2P networks is rarely restricted, and delays in the data transfer are often tolerated. On the contrary, profile sharing asks for restricted and fast access to a very high number of protected published contents.
Finally, by relying on existing architectures, current P2P-based approaches suffer not only for this file-sharing effect, but also from the security leaks inherent to the adopted P2P architecture.

Peerson [42] achieves decentralization thanks to an external DHT system such as OpenDHT [102], a centrally managed deployment of the Bamboo DHT on PlanetLab[6]. The security is assured thanks to the encryption of stored objects, and communications between users are directly peer-to-peer when both are online, while the implementation supports asynchronous messaging when this is not the case. In Peerson, a lookup in the DHT provides the meta-data information of the resource a requesting peer is looking for. Such a metadata can contain the ip address of a target peer to be contacted, or user's notifications. Once a target peer's ip is obtained, peers connect directly, then disconnect immediately except when doing instant messaging. Resistance against impersonation attacks is guaranteed by associating each user to a Global Unique Id. Such a GUID is obtained as the result of an hash function applied to a mail address, under the assumption that everyone today has an email address that is unique. Peerson does not assume any kind of trust relationship between peers, but provides access control by encryption and key management.

Lifesocial.KOM[7] [68] is an extendible plugin-based P2P OSN providing totally distributed P2P-based OSN. Initially conceived as a pure P2P solution, it has been extended to allow users for acquiring storage space at a dedicated server [95]. Reliable storage is delegated to FreePastry [104], a p2p overlay based on PAST [63]. Data objects can either

[6] http://www.planet-lab.org
[7] http://ki3.de/lifesocial/

contain final data or link other objects to be retrieved. Protected data objects are encrypted with symmetric cryptographic keys which are further encrypted with the allowed recipient public key and appended to the object itself.

Prometheus [77] is a P2P service managing social information from multiple sources. It allows users to select the peers storing sensitive information based on social trust. Built-in public-key cryptography primitives ensure data access control. Prometheus users allow this OSN to collect their social information from *social sensors*, i.e. applications that report to Prometheus the user's interactions with other users via e-mail, phone, instant messaging et similia. Information collected by these sensors is collected by Prometheus and used to create a social graph where the edges, i.e. the trust relationships, are weighted by the strength of the trust. Both the information of the social graph and that one coming from sensors are stored in an encrypted form and accessible from user's trusted peers. Users' social graph is stored on her trusted peers. Similarly to Lifesocial.KOM Prometheus runs on top of Pastry [104] and uses Past [63] for replicated storage of sensor data. Each user has a group of trusted peers storing replicas of her social subgraph for the purpose of increasing its availability. Prometheus uses Scribe [46], an application-level DHT multicast infrastructure, to manage the communication with the trusted peer group. In Prometheus, every user holds a public-private keypair. At the time of registration, newcomer peer connecting from a trusted device is assigned to a unique UID, and specifies an initial set of trusted peers contributing to the storage of the newcomer's data. While service requests can be sent to any peer, only the trusted peers of a user can provide data about that user.

Finally, ***Likir*** relies on the Kademlia [86] DHT to allow for data storage decentralization. In Likir, the user's peer node is furnished with an identifier in the form of an OpenId [99] by a certification service, and communications are encrypted and authenticated by both communicating parties. Together with the presence of a reputation system (RS), the adoption of OpenId mitigates the impact of Sybil attacks and pollution in the retrieved data. Many RS can be adopted in Likir, under the assumption that they exibit a simple API allowing any application to evaluate other user's behavior to single out the misbehaving peers. In this case, when a resource is inserted with a lookup key unrelated to its content, the resource can be marked as invalid and its publisher as a polluter.

Table 4.1 reports the aforementioned solutions for DOSNs with their main characteristics.

	Characteristics		
DOSNs	Storage place	Incentives for SNS	Access control on published data
FOAF	trusted web server	absent	by means of data encryption
Diaspora	trusted web server	absent	by means of data encryption
Vis-à-Vis	cloud computing infrastructure	commercial contracts	by means of requester authorization
Persona	trusted web server	commercial contracts	by means of data encryption
Lockr	trusted web server	commercial contracts	by means of requester authorization
Peerson	OpenDHT	absent	by means of data encryption
Lifesocial.KOM	Pastry	absent	by means of data encryption
Prometheus	Pastry / trusted peers	absent / social trust	by means of data encryption
Likir	Kademlia	reputation system	by means of data encryption

Table 4.1: Current DOSN proposals as an answer to the "Big Brother" problem in centralized OSNs

4.3 Main Limitations

The access restriction on the user's published data by means of encryption or access control lists together with the migration from a full centralized architecture to a decentralized one constitute two significant steps toward the protection on the user's security and privacy in OSNs. Nevertheless, this thesis claims that these steps are not sufficient. As a matter of fact, current DOSNs still allow the data storage service to link a requester's (often anonymous) identifier to the target user's profile she is looking for, and derive as a consequence trust relationships between users. A series of works pointed out that the information on the sole social network topology in addition to the data that most users publish in current OSNs is

sufficient to de-anonymize the input topology [92, 121, 34] and retrieve information on the social network. Therefore, current DOSNs just propose their users to choose another "Big Brother" with a more limited view of the overall network.

Literature on P2P networks already addressed the problem of anonymous communication [103, 49], and proposed solutions that are suitable for file sharing, but reveal to be inadequate in the context of DOSNs. As an example, the well known Onion Routing technique [101], where a sender node recursively encrypts secret content with the public key of the nodes composing the path this content must follow, when adopted in a Friend-to-Friend (F2F) network, where peers cooperate thanks to their friendship, would require the sender to know the social network graph topology, i.e. the information the DOSN itself aims to protect. On the other hand, when the P2P network is not a F2F one, appropriate incentive mechanisms for cooperation among peers are required.

In this thesis we propose a radically new P2P architecture for secure, privacy preserving, distributed OSN to properly target the user's security and privacy in OSN. Such a solution addresses privacy *by design*, avoids the "Big Brother" problem and guarantees cooperation between peers.

The main characteristics of this new solution are described in the following chapters.

4.4 Summary

In this chapter, we gave an overview of Distributed Online Social Networks (DOSN). We classified such DOSN approaches in two categories: client-server (or cloud), and peer-to-peer.

Client-server (or cloud) approaches require acquisition or deployment of web space hosting users' sensitive data and do not always evade the potential control of a single party, as e.g. a company or an organization, on such data. At the benefit of guaranteeing full data availability, they often require the OSN user to pay for the storage service, or for the maintenance of proprietary infrastructure.

On the other hand, current peer-to-peer approaches inherently relax the data availability requirements and provide best-effort services. Whereas such approaches do not suffer from single party control, they expose users to potential communication tracing by malicious peers. In the context of OSN, such communication traces may disclose details on the structure of the social network graph.

We therefore concluded that none of current approaches is suitable to achieve the goal of preserving user's privacy in OSNs

Part II

A privacy preserving distributed OSN leveraging real life trust

Chapter 5

Safebook

This chapter introduces the design of a newly distributed OSN addressing the security objectives presented in chapter 3. The new mechanism, called Safebook [56, 55], leverages peer-to-peer concepts and capitalizes on the trusted links from the managed social network, thus transferring the trust between the OSN users to the collaborating parties of the system. Decentralization circumvents the need for a central provider and leads to a distribution of data, communication, and control. While the alternative approach of privacy protection based on encryption is considered as a partial solution only, decentralization eliminates the privacy threats resulting from a centralized SNS entirely. However, P2P systems being devoid of any central point of control inherently suffer from a lack of trust between the parties. This problem is addressed through leveraging the trust relationships that are available as part of the SN itself. Trust relationships akin to SN, such as 'friendship' or 'acquaintance' are thus exploited to build trusted links among the nodes of the P2P SNS system.

This chapter begins with the rationale behind Safebook, followed by a description of its main components and functionalities.
At the end, the core protocols of Safebook are described in detail following the main steps a newcomer has to take to join the OSN and benefit from its services.

5.1 Rationale

As mentioned in chapter 3, current Online Social Networks severely suffer from a large number of security and privacy threats that may lead to irreversible loss of sensitive data, and consequently, to loss of money and reputation. These threats are mainly due to two concurrent factors: users lack awareness regarding the consequences of simple and sometimes presumably private actions, like accepting contact requests, tagging pictures, commenting on profiles or leaving wall posts, and SNS providers do not develop, offer or advertise appropriate security and privacy tools.

While the first factor is probably a consequence of the user's implicit trust in other profiles and on the OSN provider itself, the second one is probably due to the OSN business model, where OSN value increases together with the number of its members and the volume of the shared data [40].

One might ask: is it more convenient to propose yet another OSN, or to design and implement a set of security and privacy tools for existing ones?

In current OSNs, the main threat is the lack of protection for the user's data from the SNS provider itself. As a matter of fact, the data directly or indirectly supplied by all participants is collected and stored permanently at the databases of the service provider. This makes it an omniscient entity, that may act as a *Big Brother* monitoring and tracing users.

Uploading encrypted data to current OSNs may appear as a good solution to prevent the Big Brother problem. Unfortunately, even assuming that users store their data in an encrypted form at the OSN servers, still the monitoring of profile data lookups may reveal to the SNS provider insightful information on the social network graph itself, such as the OSN members' contact list.

Therefore a new architecture for OSNs is needed to address the current security and privacy threats, and this architecture should not propose any central entity storing all users' data.

5.1.1 Design principles

As discussed in chapter 4, in order to meet the main privacy objective, distributed data storage for OSNs may be achieved either through a client-server (or cloud) approach, where users do not participate in the storage service and the stored data is always available, or through a peer-to-peer approach, where users participate in the storage service and the stored data may not be always available.

The P2P approach inherently lends itself to the design of an architecture with the main objective of evading control by a single party such as an organization or a company. Hence we decided to opt with the P2P approach taking into account additional advantages thereof such as scalability and fault tolerance.

As a first design principle, we envision a P2P system and we rely on peer nodes to perform basic OSN operations such as:

- storage of user's data;
- lookup of user's data;
- communication among users.

Nevertheless, P2P system severely suffer for a major problem, that is the lack of cooperation among peer nodes. The absence of a-priori trust characterizing any P2P system increases the intrinsic selfishness of nodes, that often engage in ***free riding*** [32, 108, 82] and try to consume as more resources as possible without contributing to the network services. Cooperation enforcement hence poses as a mandatory requirement to effectively setup a distributed P2P OSN.

Cooperation enforcement mechanisms encourage nodes to perform a fair share of both networking and storage operations. Inducing cooperation between nodes can be based either on some reputation or on rewarding mechanisms: reputation mechanisms [41, 88, 105, 61] ensure that each node accepts to cooperate with its neighbors based on their past behavior; credit based schemes [43, 127, 117, 37] provide node collaboration by rewarding cooperating nodes with a certain amount of credits in the form of E-cash [47, 48] or a tradable good/service, that they further can use for their own benefit.

However, due to the specific context of P2P OSN where peer nodes are maintained by OSN members, we decided to fill the lack of cooperation among peer nodes with the real life trust between OSN members derived from the OSN itself. Therefore, as a second design

principle, we connect peer nodes in such a way that if one peer serves another, the user maintaining the serving peer and that one maintaining the served one are real-life friends.

5.1.2 Idea of the solution

Based on these design principles, we propose a new privacy-preserving distributed OSN and call it ***Safebook***. In the following, we discuss how we came up with the core design of the Safebook solution.

As a straightforward implementation of the two design principles, a simple P2P system whereby each user's data is stored and made available by peer nodes operated by users who are friend of that user (***friend nodes***) seems to be a reasonable first step toward the main security objective of privacy against a centralized omniscient party.
Therefore, we define a ring structure where each user is at the center of a ring, and her friend nodes hosting the user's data constitute the first ring.

Nevertheless, such a simple scheme would suffer from a further privacy problem that is due to the scheme itself. Since all the social network services pertaining to a user will be provided by this user's friend nodes, tracing of communications by very simple means would disclose the friendship relationships in the social network.
The adoption of anonymous communication techniques seems then an obvious second step toward the security objective of protection of social trust links against OSN members. However, such anonymous communication technique should be in line with the design principles previously mentioned.
Therefore, we protect the first ring with a second one consisting of nodes that each are a trusted contact of a node on the first ring. Further rings are built through similar trust relationships, without requiring nodes on the same ring to have trust relationships with one another, and without requiring transitivity of trust.
Data requests are then addressed to the nodes in the outermost ring, and are forwarded to the nodes in the first one along hop-by-hop trusted links. Data is served by nodes in the innermost ring and replies are sent back along the same paths. Safebook thus consists of the collection of concentric layers of peers nodes organized around each user in order to assure data storage and communication privacy.

Security and privacy of the system might be compromised if malicious users were able to impersonate legitimate ones. In addition to the attacks discussed in chapter 3, malicious users would then be able to intrude into the rings surrounding a target victim and derive the

friendship relationship we aim to protect. As a consequence, a third step toward ensuring user authentication has been taken.
In Safebook, a Trusted Identification Service (TIS) that does not take part in the OSN provides users with unambiguous certified identifiers associated to their real identities. Such TIS does not contrast with the purpose of decentralization, as it can be implemented in a decentralized fashion. TIS is not involved in any communication or data management operation among users, is contacted only once, and can be provided off-line.
Finally, classical encryption techniques have been adopted to ensure data confidentiality and data integrity.

In summary, Safebook has been designed as an OSN addressing privacy from the very beginning. Privacy against centralized omniscient entities is achieved with the adoption of a decentralized P2P approach. Privacy against malicious users is achieved with communication obfuscation through anonymous routing techniques, data confidentiality through the use of encryption, and profile integrity through certified identifiers. For these reasons, Safebook achieves privacy *by design*.

Availability of basic services such as data storage, data lookup and communication is guaranteed by the cooperation among peer nodes. Such cooperation is enforced by real-life trust among OSN members.

Nevertheless, in the specific context of OSN, we realize that **the real life trust between users** can serve much more than simple cooperation: it **can be used to build the online social network itself** (see figure 5.1). Therefore, the OSN helps user to establish friend relationships, and friend nodes provide the basic services of data storage, retrieval and communication, and consequently build the OSN.

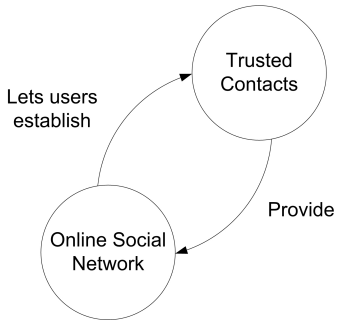

Figure 5.1: Cyclic relation showing how real life trust between users can build the OSN itself.

The characteristics of Safebook are described in detail in the following sections.

5.2 Main components

The real life trust between OSN users as in the SN graph is mapped into ring structures called ***Matryoshkas***, where node neighborhood is based on user friendship. Direct trust relationships are leveraged for the purpose of data storage and data availability. Since friend nodes are considered honest but curious, data is stored in an encrypted form.

In Safebook, a target profile data can be accessed through hop-by-hop trusted paths whose endpoints can be retrieved from an additional ***Peer-to-Peer*** system maintained by the OSN users themselves. Differently from the Matryoshkas, this P2P system is used for indexing purpose only: it does not store user's profile data and does not take into account user's friendship relations.

Safebook can thus be seen as an overlay network composed by two different layers:

- the Social Network Layer consisting of Matryoshkas and providing each member with a set of functions corresponding to social interactions in the real life, such as profile data retrieval, message exchange et similia;
- the Peer-to-Peer layer offering the infrastructure to build and to access the Matryoshkas.

A Safebook user is represented as a host on the Internet, a peer node in the P2P layer and a user in the SN layer (see figure 5.2, left). Different identifiers are used to address the same party in each layer: a ***user Id*** denotes a node in the SN layer, a ***node Id*** in the P2P layer, finally an IP address in the Internet layer.

In addition to Matryoshkas and the P2P system, the last component of Safebook is an off-line ***Trusted Identification Service*** (TIS) (see figure 5.2, center) in charge of generating the identifiers needed to address users in the SN layer and peer nodes in the P2P layer. Since these identifiers are issued together with corresponding certificates, they can never be manipulated nor forged.

5.2.1 Matryoshka

A Matryoshka is a friend-of-friend structure providing the user with data storage and communication obfuscation services.

A user $\mathcal{V}$'s Matryoshka $\Theta_{\mathcal{V}}$ consists of a group of nodes surrounding $\mathcal{V}$'s node (see figure 5.2, right). The nodes of a Matryoshka are organized into several concentric rings, namely

Figure 5.2: Safebook overlays (left), main components (center) and Matryoshka (right).

shells, and several paths lead from the nodes in the innermost shell $\Lambda_{\mathcal{V}}$ to the nodes in the outermost shell $\Omega_{\mathcal{V}}$. With $\theta_{\mathcal{V}}^{j} \in \Theta_{\mathcal{V}}$ being a node in the jth shell, with $j \in [0, \ldots, MaxShell]$, each Matryoshka further features the following properties:

1. $\mathcal{V}$'s node $\theta_{\mathcal{V}}^{0}$ is located at the center of the Matryoshka and is called the ***core***;
2. if a pair of nodes $\left(\theta_{\mathcal{V}}^{j},\ \theta_{\mathcal{V}}^{j+1}\right)$ is connected, a friendship relation between them exists in the social network layer;
3. each node $\theta_{\mathcal{V}}^{1}$, located on the innermost shell $\Lambda_{\mathcal{V}}$ and called a ***mirror***, is a trusted contact of the core $\mathcal{V}$ and stores $\mathcal{V}$'s data in an encrypted form;
4. each node $\theta_{\mathcal{V}}^{MaxShell}$, located on the outermost shell $\Omega_{\mathcal{V}}$ and called an ***entrypoint***, acts as a gateway for all the requests destined to $\mathcal{V}$;
5. each node $\theta_{\mathcal{V}}^{j}, j \in [2, MaxShell - 1]$, located on a shell between $\Lambda_{\mathcal{V}}$ and $\Omega_{\mathcal{V}}$, is called a ***prism*** of $\mathcal{V}$;
6. the set of prisms is denoted as $\Delta_{\mathcal{V}}$.

In summary $\mathcal{V}$'s Matryoshka $\Theta_{\mathcal{V}}$ is the union of the set of mirrors $\Lambda_{\mathcal{V}}$, the set of prisms $\Delta_{\mathcal{V}}$, the set of entrypoints $\Omega_{\mathcal{V}}$ and the core $\mathcal{V}$. The number of $\mathcal{V}$'s mirrors represents the number of available partitions of $\mathcal{V}$'s profile data, while there are as many entrypoints as paths that can lead to a mirror. Each i^{th} mirror $\lambda_i \in \Lambda_{\mathcal{V}}$ represents the root of a subtree with leaves that are lying in the outermost shell $\Omega_{\mathcal{V}}$. The branching of all the subtrees, the ***span factor***, is set by $\mathcal{V}$. The cardinality $\|\cdot\|$ of the set $\Omega_{\mathcal{V}}$ in consequence is $\|\Omega_{\mathcal{V}}\| = \|\Lambda_{\mathcal{V}}\| Span^{MaxShell-1}$.

5.2.2 Peer-to-peer substrate

The P2P substrate of Safebook is a DHT similar to KAD [86, 110] in charge of storing and retrieving the entrypoint references of all the users' Matryoshkas. Such a substrate comprises of all user nodes and allows any node to issue a lookup query to reach the Matryoshka of any user.

The DHT is defined as:

$$DHT = \langle K, N, R, id_n(\cdot), id_r(\cdot), \rho(\cdot) \rangle$$

where K is the DHT keyspace, N and R correspond to the set of nodes and the set of resources, respectively, and $id_n : N \to K$, $id_r : R \to K$ denote the functions associating

a node and a resource to their identifier respectively. Finally, $\rho : K \rightarrow \{N\}$ denotes the mapping function which outputs the set of peers responsible for a resource given the resource identifier.

A resource consist on a list of entrypoint references of a target user's Matryoshka. The corresponding resource identifier $DhtKey$ is represented by a user identifier or by an hash of the user's attributes such as her full name, her birthday etc.

Redundant copies of (key value) pairs ($DhtKey$, resource) can be stored by nodes whose identifier matches $DhtKey$ on a predefined amount of first bits.

Much alike KAD, Safebook implements a greedy routing, minimizing the distance measured in an XOR-metric between the $DhtKey$ to locate and the node Id of neighboring nodes. Due to the privacy-by-design constraint, unlike KAD, the lookup queries are not always processed iteratively: Safebook uses recursive processing with hop-by-hop anonymization as a basic technique to assure the untraceability of requesting parties in case a list of entrypoint reference is queried.

5.2.3 Trusted Identification Service

The TIS is a trusted third party that generates and grants for each Safebook user $\mathcal{V}$ a pair of identifiers: a node Id ($NId_{\mathcal{V}}$), unambiguously identifying $\mathcal{V}$ as a peer in the P2P layer, and a user Id ($UId_{\mathcal{V}}$) unambiguously identifying $\mathcal{V}$ as a user in the social network layer. Both identifiers are computed starting from a set of $\mathcal{V}$'s properties such as $\mathcal{V}$'s full name, birthday, birthplace etc.
A pair of certificates link each identifier to a respective public key provided by $\mathcal{V}$. Corresponding private keys are known by $\mathcal{V}$ and nobody else.

Since the P2P system allows to retrieve a node IP address given a node Id, the separation of node- and user- identifiers is required to prevent malicious users from deriving a victim's IP address. Only trusted contacts of a node are able to link these two identifiers, as they serve as mirrors and in consequence know both. TIS constitutes an exception, as it is the only party in Safebook that is able to link the user Id and node Id of users other than their own trusted acquaintances. If compromised, in addition to the users' location, TIS may also disclose users' participation in Safebook. However, the TIS does not possess any user's private keys, therefore it cannot impersonate any victim, nor retrieve her set of trusted contacts or access data content published with restrictions.

While the TIS is a centralized infrastructure and in consequence might appear to break

the paradigm of a decentralized architecture of Safebook, it can easily be implemented in a distributed fashion. Furthermore, it is an off-line service used only once by each Safebook user and, unlike a central SNS server, it does not threaten the privacy of users, as it is not involved in any communication or data management operation among users or peer nodes.

A collusion of the TIS with the Internet Service Provider would circumvent the concept of separation of identifiers. However, this attack is only successful if the ISP controls the access to all users of Safebook, as only the privacy of users using the directly monitored Internet connections can be disclosed. Entirely protecting the privacy against a malicious ISP is only possible when leveraging much more complex concepts of anonymization, which for the sake of efficiency is refrained of. Safebook indeed does not provide anonymous communications on the network level.

In the following section we present the main functionalities of Safebook, which allow the users to share their sensitive data with limitations and communicate between them.

5.3 Functionalities

The main functionalities of Safebook may be divided into three main categories:

- data management;
- key management;
- communication management.

Each functionality is detailed in the following of this chapter.

5.3.1 Data Management

Data management functionalities allow users to generate, modify and delete sensitive information in the OSN.

In Safebook, data objects, also referred as ***data items***, are user-generated pieces of information describing the user's profile as detailed in chapter 2.3.

A data item $\mathscr{D}$ is represented as a tuple $< DId, type, value, version >$, where *type* describes the nature of the data, such as personal contact details, connectivity, interests etc. (see figure 2.3), *value* constitutes its content, and *version* its current version. A data item identifier $DId_{\mathscr{D}}$ unambiguously identifies $\mathscr{D}$ among all the data objets, and allows for the basic operations of item storage, retrieval or deletion.

A ***Distributed Data Storage Space*** (DDSS) $\mathcal{S}_{\mathcal{V}}$ is defined for each user based on her friendship relations. Authorization to store content on such a space comes from the real life friendship relations of $\mathcal{V}$, and therefore is granted to $\mathcal{V}$ only.

The size of $\mathcal{S}_{\mathcal{V}}$ is dynamic: at friendship establishment, each friend $\mathcal{F}_i$ of $\mathcal{V}$ reserves an arbitrary amount of her own ***Local Data Storage Space*** (LDSS) $\mathcal{L}_{\mathcal{F}_i}^{\mathcal{V}}$ for $\mathcal{V}$. The sum of each friend's LDSS allocated for $\mathcal{V}$ finally builds $\mathcal{V}$'s DDSS.

Due to the distributed nature of DDSS, data is partitioned into n blocks and for a given amount of redundancy these blocks are coded in $n+l$ fragments such that any n fragments are sufficient to reconstruct the original object.

Before being partitioned, encryption operations may be performed on $\mathcal{D}$ to guarantee its confidentiality and limit the access on it.

5.3.2 Key Management

As stated previously, user's data may be encrypted based on the willingness of the owner. Key management functionalities allow users to limit access on their shared sensitive data.

Safebook ensures data confidentiality thanks to traditional public-key and symmetric cryptography. Access to the content can be restricted to several user-defined (overlapping) groups of contacts.

In order to minimize the storage overhead at the DDSS, data is encrypted with only one key, namely the ***Data Encryption Key*** (DEK). This DEK needs to be distributed among all users that are authorized to decrypt the data. The distribution of a DEK requires the encryption of it with a ***Key Encryption Key*** (KEK) which is previously distributed among members during friendship establishment. Users do not rely on any third party to perform key distribution; they send the keying material to all the group members they manage.

Friends of $\mathcal{V}$ access $\mathcal{S}_{\mathcal{V}}$ within the limits of the ***Access Control Policy*** (ACP) defined by $\mathcal{V}$.

Basically, users in Safebook create groups of contacts by defining several attributes, such as 'Family', 'Colleagues' etc. and associate them to each contact. Data protected under these attributes will then be accessible to all the contacts associated to the appropriate attributes only.

In Safebook, attributes are defined through ***Badges***. Users in Safebook know which badges they provided to which contact, but cannot know how many badges they received from a given contact, nor the description of the associated attribute. For instance, $\mathcal{V}$ may grant $\mathcal{U}$ a 'Professional' badge without disclosing the attribute 'Professional' to $\mathcal{U}$, and without revealing who among $\mathcal{V}$'s contacts holds this badge too. This happens since, from a system perspective, a badge b corresponds to a set of DEKs used to encrypt the data accessible to all the contacts provided with that badge. Such a set is defined as:

$$\mathcal{D}_b^n = \left\{ h^i\left(s_b\right) : i \in \{1 \ldots n\} \right\}^{1}$$

where $h^i\left(s_b\right)$ denotes a well known one-way hash function $h\left(\right)$ sequentially applied i times to an initial seed s_b. The idea of sequential password hashing was originally proposed in [79] and afterward leveraged to design one-time password authentication systems such as S/Key [72]. Safebook does not perform authentication of requesting users and uses each hash as a

[1] In this notation, the colon (':') means 'such that'.

DEK rather than a one-time password.

When $\mathcal{V}$ grants $\mathcal{U}$ a badge b, $\mathcal{U}$ receives a DEK $h^i(s_b)$ that does not reveal anything about the badge attribute nor the list of $\mathcal{V}$'s friends who received that badge too.
After the reception of $h^i(s_b)$, $\mathcal{U}$ is able to derive all the keys $\left\{h^j(s_b) : j \in \{i \dots n\}\right\} \subseteq \mathcal{D}_b^n$ and access all the data stored in $\mathcal{S}_\mathcal{V}$ encrypted with such DEKs.

Safebook does not ensure ***backward secrecy***: in a Social Network context, in fact, users are likely to allow a new group member to access the data previously shared for that group.

When $\mathcal{V}$ revokes b from $\mathcal{U}$, $\mathcal{V}$ advertises $h^{i-1}(s_b)$ to all the contacts previously granted with b, one by one, except $\mathcal{U}$. Future data previously accessible by contacts granted with b will be encrypted by $\mathcal{V}$ with the DEK $h^{i-1}(s_b)$. Previously published data encrypted with $h^j(s_b)$ (being $j \in \{i, \dots, n\}$) will not be encrypted again and will thus still be accessible by $\mathcal{U}$.

Since reversing the hash function $h()$ is computationally infeasible, Safebook ensures ***forward secrecy*** as future communication will not be accessible by the leaving member $\mathcal{U}$.

Generally speaking, $\mathcal{V}$ defines her ACP by specifying a set of ***badge rules*** $r \in \mathcal{R}_\mathcal{V}$ and assigning a seed s_r to each rule.

When a contact $\mathcal{U}$ is granted with a set of badges $\mathcal{B}_\mathcal{V}^\mathcal{U}$ from $\mathcal{V}$, a DEK set

$$\mathcal{E}^\mathcal{U} := \left\{h^i\left(s_{r_j^\mathcal{U}}\right) : r_j^\mathcal{U} \in \mathcal{R}_\mathcal{V}^\mathcal{U} \forall j \in \{1, \dots, \|\mathcal{R}_\mathcal{V}\|\}, i \in \{1, \dots, n\}\right\}$$

corresponding to the rules $\mathcal{R}_\mathcal{V}^\mathcal{U}$ satisfied by $\mathcal{U}$ is sent to him.
Table 5.1 shows an example of ACP.

When revoking a badge b from $\mathcal{U}$, $\mathcal{V}$ advertises a new set of DEKs $E^\mathcal{X}$ to every contact $\mathcal{X}$ satisfying one or more rules $\mathcal{U}$ was also satisfying before b was revoked from him. From this point on, $\mathcal{V}$ encrypts her data with the new DEKs.

Figure 5.3 shows a communication scenario between users in Safebook.

5.3.3 Communication Management

Communication management functionalities allow users to establish unobservable friendship links and to communicate with each other while ensuring confidentiality and message integrity.

Communication between two users $\mathcal{V}$ and $\mathcal{U}$ can take place either in a synchronous or asynchronous fashion.

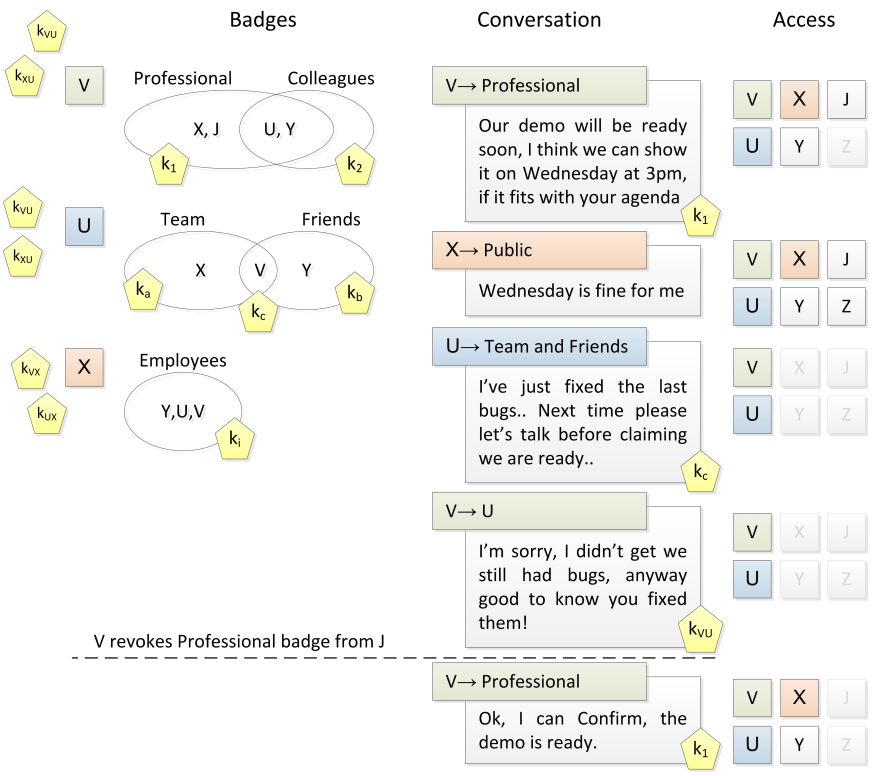

Figure 5.3: An example of communication between users with different ACPs.

Rule r	Seed s_r	Current exponent i
B_{Prof}	s_{r_1}	n-3
B_{Family}	s_{r_2}	n-2
B_{Team}	s_{r_3}	n-1
$B_{Prof} \vee B_{Family}$	s_{r_4}	n-3

Table 5.1: An example of ACP based on set operations between contacts granted with user-defined badges.

In the first case, both parties exchange messages in real time. Each user stores such messages in her own DDSS and shares it with trusted contacts if needed.
In the second case, $\mathcal{V}$ generates a message for $\mathcal{U}$ and stores it in her DDSS $\mathcal{S}_\mathcal{V}$. Once $\mathcal{U}$ looks up for new available $\mathcal{V}$'s data, she retrieves the message. To reply, $\mathcal{U}$ follows the same steps: she stores the reply in her own $\mathcal{S}_\mathcal{U}$, then $\mathcal{V}$ retrieves this reply while querying for $\mathcal{V}$'s new

data.

Message integrity is guaranteed by the use of digital signature, while communication confidentiality is achieved by encrypting messages with a symmetric DEK computed (in case of synchronous communication) or previously shared (in case of asynchronous one) between the sender and receiver.

Communication is obfuscated through multi-hop routing of messages along friend-of-friends chains in such a way that information on data requester cannot be retrieved. In case of synchronous communication, this hides the IP address of communicating parties[2] and therefore their location. In case of asynchronous communication, this also prevents a user $\mathcal{V}$'s friend $\mathcal{F}_i$ storing $\mathcal{V}$'s data from deriving the trust relationships between $\mathcal{V}$ and the data requester $\mathcal{U}$.

[2]However, synchronous communication between trusted parties may be directly established.

5.4 Core protocols

Core functions of Safebook implement three main groups of operations:

- ***profile creation***, where the user's identity is created and granted with the certificates issued by the TIS;
- ***SNS setup and maintenance***, where user's node takes part in the distributed SNS architecture of Safebook;
- ***SN communication and relations management***, where user benefits from the SNS.

Each operation calls for the execution of a series of secure protocols aiming at obtaining credentials, building and keeping the consistency of the Safebook overlays and establishing secure communication channels. Throughout the description of these protocols, $\{M\}_{S_{\mathcal{K}_\mathcal{X}}}$ denotes a message M being signed by user $\mathcal{X}$'s private key $\mathcal{K}_\mathcal{X}^-$, and $E_{\mathcal{K}_\mathcal{Y}}\{M\}$ denotes the message M being encrypted with the user $\mathcal{Y}$'s public key $\mathcal{K}_\mathcal{Y}^+$[3]. The distinct identifiers of Safebook users are associated with keypairs: while $\mathcal{N}_\mathcal{X} = \{\mathcal{N}_\mathcal{X}^-, \mathcal{N}_\mathcal{X}^+\}$ denotes the keypair for the node ID, $\mathcal{U}_\mathcal{X} = \{\mathcal{U}_\mathcal{X}^-, \mathcal{U}_\mathcal{X}^+\}$ denotes the keypair for the user ID of node $\mathcal{X}$[4].

To assure integrity and confidentiality, all messages at each hop are signed with the sender's ($\mathcal{X}$) node ID private key and encrypted with the receiver's ($\mathcal{Y}$) node ID public key. For the sake of clarity, the resulting term $E_{\mathcal{N}_\mathcal{Y}}\left\{\{M\}_{S_{\mathcal{N}_\mathcal{X}}}\right\}$ is simplified and is denoted as M in the remainder of this thesis.

5.4.1 Profile Creation

The identity creation protocol (see figure 5.4) is responsible for providing a new user $\mathcal{V}$ with the credentials required to participate in Safebook.

In order to join, a new node $\mathcal{V}$ must be invited by a registered user $\mathcal{A}$ that needs to be an acquaintance in real life. Initially, $\mathcal{A}$ sends out-of-band $\mathcal{V}$ an invitation request *invReq* message, signed using the private key $\mathcal{U}_\mathcal{A}^-$. It contains a tuple $Name_\mathcal{A}$ of properties that

[3] More precisely, session keys are used to encrypt the payload. Such keys are advertised at the beginning of the message encrypted with the target node Id public key.

[4] Each private key associated with a node- or user- identifier is generated by the owner of the identifier and known to nobody else.

identify the user $\mathcal{A}$, the certificate $Cert\left(h\left(Name_{\mathcal{A}}\right), \mathcal{U}_{\mathcal{A}}^{+}\right)$, as granted by the TIS, and the public key TIS^{+} of the TIS. The *invReq* message is the only message that is sent in clear text, since $\mathcal{V}$'s public node- and user- ID keys haven't yet been generated nor certified and it is sent out of band anyway.
Upon reception of the *invReq* message, $\mathcal{V}$ generates the two keypairs $\mathcal{N}Id_{\mathcal{V}}$ and $\mathcal{U}Id_{\mathcal{V}}$. Subsequently, it starts another out of band process: it creates its own identity tuple $Name_{\mathcal{V}}$ together with a proof of ownership of $Name_{\mathcal{V}}$, and transmits both together with the public key $\mathcal{U}_{\mathcal{V}}^{+}$, in a *credReq* message, to the TIS.
The TIS then generates $\mathcal{V}$'s user ID $UId_{\mathcal{V}}$ and node ID $NId_{\mathcal{V}}$ by applying two distinct keyed hash functions $h_{MK_1}(\cdot)$ and $h_{MK_2}(\cdot)$ on $Name_{\mathcal{V}}$. Additionally, it generates and signs the registration keys of $\mathcal{V}$ $DhtKey_{\mathcal{V}}$ by hashing and signing all permutations of elements in $Name_{\mathcal{V}}$.

The TIS responds with a *credRep* message out of band, with the generated identifiers and DHT keys, together with the respective certificates $Cert\left(UId_{\mathcal{V}}, \mathcal{U}_{\mathcal{V}}^{+}\right)$, $Cert\left(NId_{\mathcal{V}}, \mathcal{N}_{\mathcal{V}}^{+}\right)$, and $Cert\left(DhtKey_{\mathcal{V}}, \mathcal{U}_{\mathcal{V}}^{+}\right)$.

On reception of *credRep*, $\mathcal{V}$ joins Safebook and hence the P2P substrate and can start creating her own Matryoshka. Subsequently, all messages sent from and received by $\mathcal{V}$ in the P2P overlay are signed using the sender's $\mathcal{N}^{-}$ and encrypted using $\mathcal{N}^{+}$ of the receiver.

5.4.2 Social Network Service setup and maintenance

Once created her account, the user $\mathcal{V}$ is able to setup her Matryoshka and to get reachable by other users.

Matryoshka Setup Protocol

The Matryoshka setup protocol (see figure 5.5) allows for the creation of Matryoshkas. During the first execution of this protocol, the initiating node $\mathcal{V}$ sends the inviting node $\mathcal{A}$ a path creation request *PathReq*. This message contains a ***registration token*** *RegTok*, a data structure *TtlMatr* for the number of hops on the created paths, the span factor *Span* for the tree through the Matryoshka, and a signed random number *Rnd*. The registration token includes the DHT keys to be registered, in order for $\mathcal{V}$ to be found in the OSN, $\mathcal{V}$'s user ID certificate, authenticating $UId_{\mathcal{V}}$, and the lifetime *ExpireTime* of the DHT registration of the $DhtKey_{\mathcal{V}}$. The *TtlMatr* is a recursively signed data structure generated by $\mathcal{V}$ including

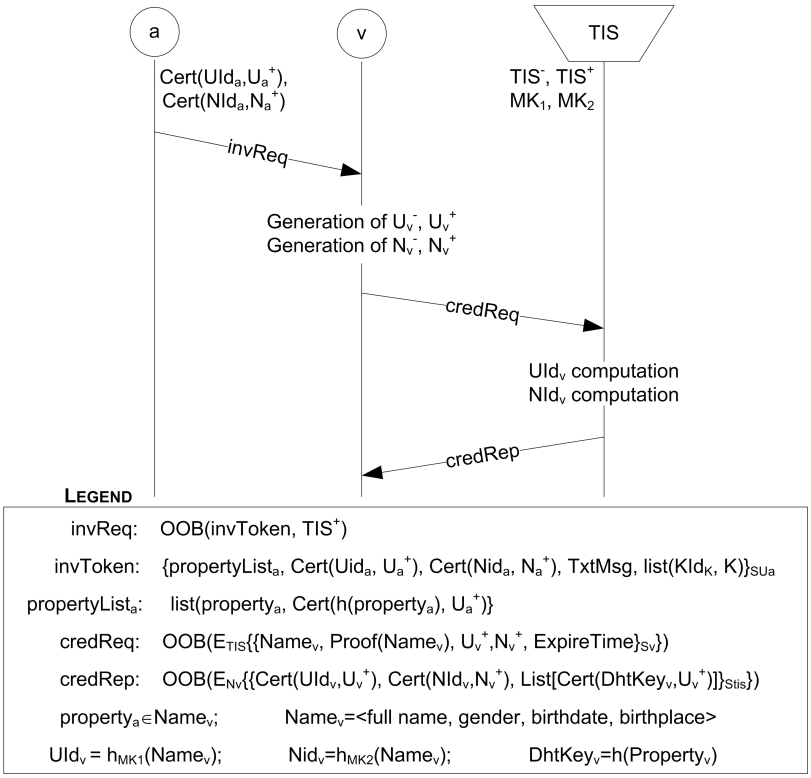

Figure 5.4: Account creation for user $\mathcal{V}$.

a set of decreasing time-to-live values based on the desired hop length from $\mathcal{V}$'s core to one of its $\Theta_{\mathcal{V}}$'s entrypoints. Each node on reception of the *PathReq* removes one or more of *TtlMatr*'s signatures, thus potentially causing a continuous decrease of the ttl value at each hop. The value in *Span* indicates to the mirrors and prisms of $\mathcal{V}$ how many next hop nodes should be selected in order to guarantee the desired availability of the data that $\mathcal{V}$ publishes. Upon receipt of a *pathReq* message, each mirror of $\mathcal{V}$ verifies the integrity of the registration token by checking its signature with the key $\mathcal{U}_{\mathcal{V}}^{+}$ contained in the TIS certificate. It then removes one or more signatures[5] from *TtlMatr* and selects a next hop $\mathcal{B}$ from its friendlist for the path and forwards the updated *pathReq*. In case the core has set a spanning factor greater than 1, it selects further nodes to forward the updated *pathReq* in order to achieve the requested branching. This process is recursive: $\mathcal{B}$ removes a signature from *TtlMatr* and forwards the updated *pathReq* to a number *Span* of his selected trusted contacts, and

[5]Removing more signatures allows Matryoshka chains to have different lengths. However, the value of a *TtlMatr* can never be increased to protect against DOS attacks.

so on, until, at one node $\mathcal{D}$, the last signature from $TtlMatr$ is removed. $\mathcal{D}$ becomes in consequence an entrypoint for the Matryoshka of $\mathcal{V}$.
For this purpose it routes one registration request for each key in $DhtKey$ through the P2P system.

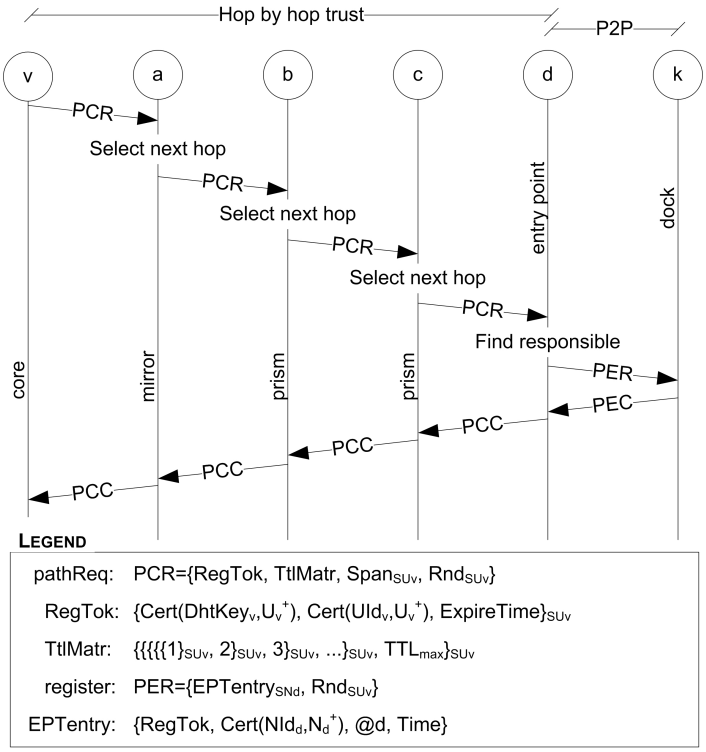

Figure 5.5: Matryoshka setup for user $\mathcal{V}$.

Since $\mathcal{D}$'s reference as a $\Theta_{\mathcal{V}}$'s entrypoint is going to be a public domain information, $\mathcal{D}$ can find a node $\mathcal{K}$, whose node ID is closest enough to the registration key, in an iterative way: $\mathcal{D}$ selects from its neighbors the node $\mathcal{N}_1$ with the node ID being closest to the registration key, measured using the XOR-metric, as the next hop. $\mathcal{N}_1$ provides $\mathcal{D}$ with the reference to (one or more) closest node $\mathcal{N}_2$ and so on, until a sufficiently close node $\mathcal{K}$ is reached. Such a node $\mathcal{K}$, called ***dock***, is in charge of storing the association ($DhtKey$,$EPTentry$) in the P2P system. $\mathcal{D}$ then sends $\mathcal{K}$ a *register* message containing the field $EPTentry_{S_{\mathcal{N}_\mathcal{D}}}$, and the random number $Rnd_{S_{\mathcal{U}_\mathcal{V}}}$ (see figure 5.6). The $Rnd_{S_{\mathcal{U}_\mathcal{V}}}$ poses as an authorization and $\mathcal{D}$ can claim to be a valid entry point for $\mathcal{V}$. $EPTentry$ is the new

record that $\mathcal{K}$ adds to its ***Entrypoint Table*** (EPT) and contains the registration token $RegTok$, $\mathcal{D}$'s node ID certificate, $\mathcal{D}$'s ip address and a timestamp $time$. $\mathcal{K}$ then updates its EPT and responds with a $pathRep$ message that is forwarded back to $\mathcal{V}$ along the inverse path. Additionally, much alike KAD, in Safebook $\mathcal{K}$ stores all registered values in k nodes around the target node of a registration request, the $RespArea$ of docks for a registration key.

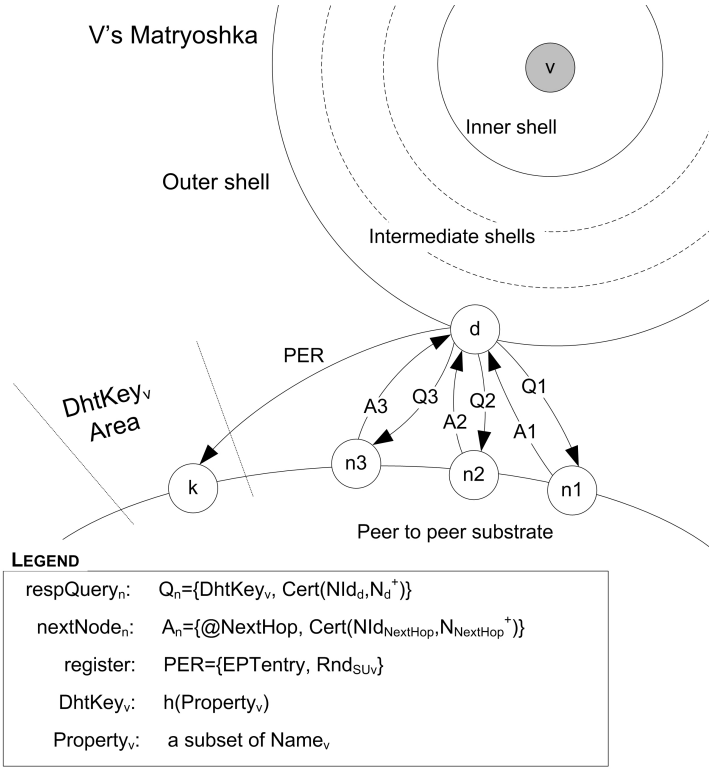

Figure 5.6: Entrypoint registration for user $\mathcal{V}$'s Matryoshka.

Matryoshka Update Protocol

User $\mathcal{V}$'s Matryoshka plays a fundamental role in guaranteeing both communication privacy to $\mathcal{V}$ and the availability of $\mathcal{V}$'s data to all the other users, without the need for $\mathcal{V}$ to be online. For this reason, the structure of $\Theta_{\mathcal{V}}$ always automatically has to be kept valid using the Matryoshka update protocol (see figure 5.7), even in case of node arrival and departure, the latter possibly being due to choice (a user logging out from Safebook) or failure (an Internet

connection problem). Considering that a node $\mathcal{B}$ leaves Safebook, it sends a *nodeLeaving* message to the neighbors inward ($\mathcal{A}$) and outward ($\mathcal{C}$,...) on the path through the Matryoshka. The message contains the user identifier $UId_{\mathcal{V}}$ of the affected Matryoshka and is forwarded to all entry points, thus pruning the subtree rooted in $\mathcal{B}$. The entry points send an *unregister* message to all the docks $\mathcal{K}$ previously addressed in the registration phase. $\mathcal{A}$ at the same time resends the *pathReq* message and sends it to a new selected contact $\mathcal{E}$ without requiring $\mathcal{V}$ to be online. From this point on, the update process is analogous to the path creation.

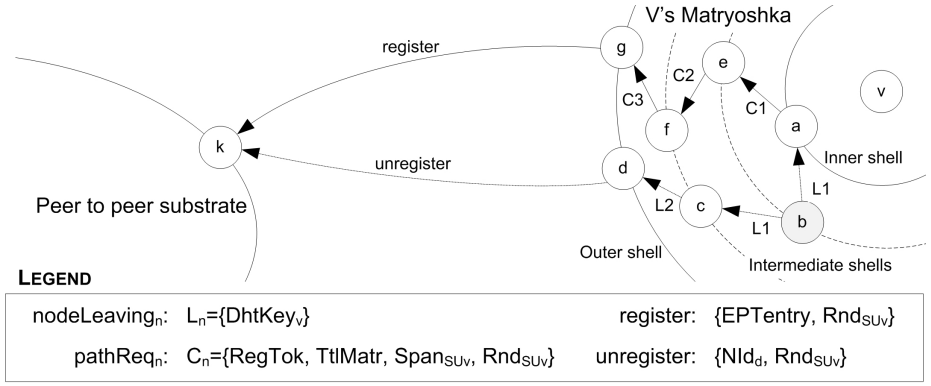

Figure 5.7: A $\mathcal{V}$'s prism is leaving $\Theta_{\mathcal{V}}$.

5.4.3 Social Network Communication and Relationship management

Matryoshkas allow users to access the OSN facilities (see figure 2.2). In the following, we will examine in detail the protocols in charge of looking up for a target profile data *friend lookup/data retrieval*, befriending an user *friendship establishment*, and store data at friends node *data storage*.

Lookup Protocol

The lookup protocol (see figure 5.8) allows for the retrieval of the entrypoint list of a user $\mathcal{V}$'s Matryoshka $\Theta_{\mathcal{V}}$. A requesting user $\mathcal{U}$ initiates a recursive lookup in the P2P system by computing $DhtKey_{\mathcal{V}}$. As soon as the lookup message *epLook* reaches one of $\mathcal{V}$'s docks, the dock responds with an *epRep* message, containing the EPT entry corresponding to

$DhtKey_{\mathcal{V}}$[6]:

$$epRep = \left\{EPTentry\left(DhtKey_{\mathcal{V}}\right), Cert\left(NId_{\mathcal{K}}, \mathcal{N}_{\mathcal{K}}^{+}\right)\right\}_{S_{\mathcal{N}_{\mathcal{K}}}}.$$

The EPT entries are cached on reception in order to avoid multiple redundant requests.

Data Retrieval Protocol

Once the entrypoints of the Matryoshka of a user $\mathcal{V}$ are discovered, the data retrieval protocol enables the user $\mathcal{U}$ to retrieve $\mathcal{V}$'s profile data $Prof_{\mathcal{V}}$ in an encrypted form. First of all, $\mathcal{U}$ delegates one of his innermost shell nodes $\mathcal{Z}$ to send a profile request message $profReq$ for $\mathcal{V}$'s data to $\mathcal{D}$, one of the entrypoints of $\mathcal{V}$'s Matryoshka. This request is recursively forwarded through the Matryoshka to $\mathcal{A}$, one of the mirrors of $\mathcal{V}$, that is storing $Prof_{\mathcal{V}}$. $\mathcal{A}$ then responds with a profile reply message $profRep$ containing a list of encrypted signed data items of $\mathcal{V}$. This message reaches $\mathcal{U}$ by following the same path in the reverse order (see figure 5.8). According to its privileges, $\mathcal{U}$ subsequently is able to decrypt and access certain parts of this data.

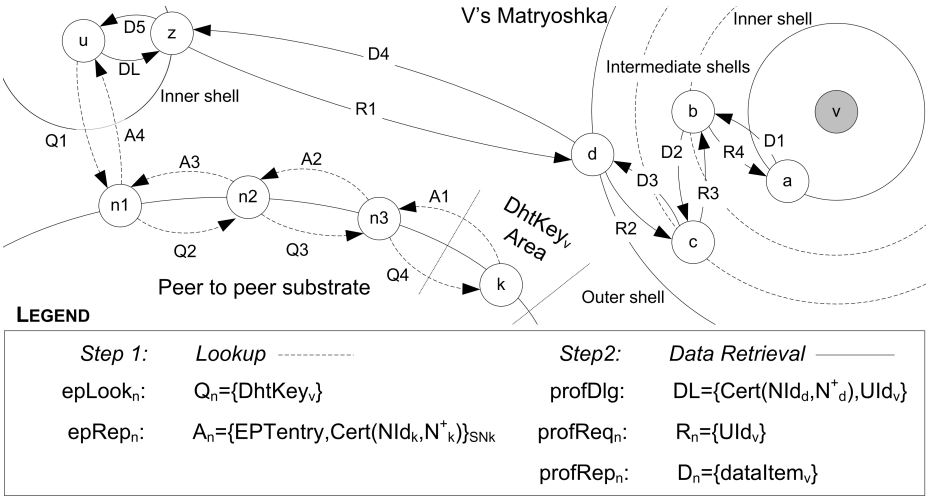

Figure 5.8: $\mathcal{V}$'s data lookup.

[6]Several DHT lookup keys for the same target user can be computed starting from different properties such as first name, birthday etc. and served from different docks.

Friendship Establishment

In Safebook the trust relations are not considered as symmetric. Rather than requesting for a target user $\mathcal{V}$'s friendship, a Safebook user $\mathcal{U}$ *advertises* her friendship to $\mathcal{V}$.

This advertisement takes place in three steps: first of all, $\mathcal{U}$ looks up for all the publicly available data of the users holding a set of properties corresponding to several $DhtLkey$; secondly, among all the retrieved profiles, $\mathcal{U}$ selects the target user $\mathcal{V}$ to be advertised; finally, a friendship advertisement message $frAdv$ is sent to $\mathcal{V}$ through $\mathcal{V}$'s Matryoshka (see figure 5.9). Such a message includes $UId_{\mathcal{V}}$[7] and a friend token consisting on the certified identity of $\mathcal{U}$, her node and user identifiers, a short friendship message, and a list of symmetric keys to be used to decrypt $\mathcal{U}$'s protected data.

Friendship advertisements may be repeatedly delegated to a trusted contact $\mathcal{Z}$ in the advertiser (or her friend-of-friend) Matryoshka through a $frDel$ message containing $frAdv$ together with the entrypoint list of $\mathcal{V}$'s Matryoshka.

In case $\mathcal{V}$ is offline, her mirror $\mathcal{A}$ will act as a mailbox and keep the friendship advertisiement until $\mathcal{V}$ will get on-line again.

If $\mathcal{V}$ replies $\mathcal{U}$ with her friendship advertisement, the trust relation becomes symmetric: $\mathcal{U}$ can become a new mirror of $\mathcal{V}$ and vice-versa.

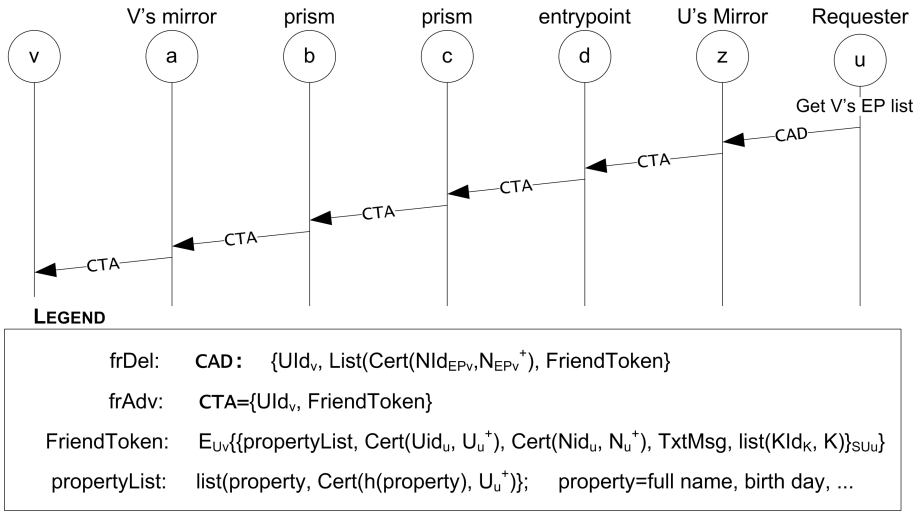

Figure 5.9: Friendship advertisement in Safebook.

[7] nodes may serve more than a single Matryoshka.

Data storage

A user $\mathcal{U}$'s data item $\mathscr{D}$ is assembled into a token together with the corresponding DId and $Dversion$. Tokens are further signed with $\mathcal{U}_{\mathcal{U}}^{-}$ and encrypted with DEK. Such an encrypted signed data token $ESDtok$ is further stored at $\mathcal{U}$'s new mirror $\mathcal{V}$ in a $dataStore$ message together with DId, $Dversion$ and the DEK identifier $DEKid$ that are used by $\mathcal{V}$ as a filter while replying to a profile data request addressing $\mathcal{U}$.

At the reception of $dataStore$, $\mathcal{V}$ indexes $ESDtok$ with Did, $Dversion$, $DEKid$, at $\mathcal{U}$'s DDSS, before replying with a $storeConf$ message. Upon the confirmation reception, $\mathcal{U}$ can keep track on which (partitioned) (encrypted) item is stored at which mirror.

Figure 5.10 shows the details of the data storage protocol.

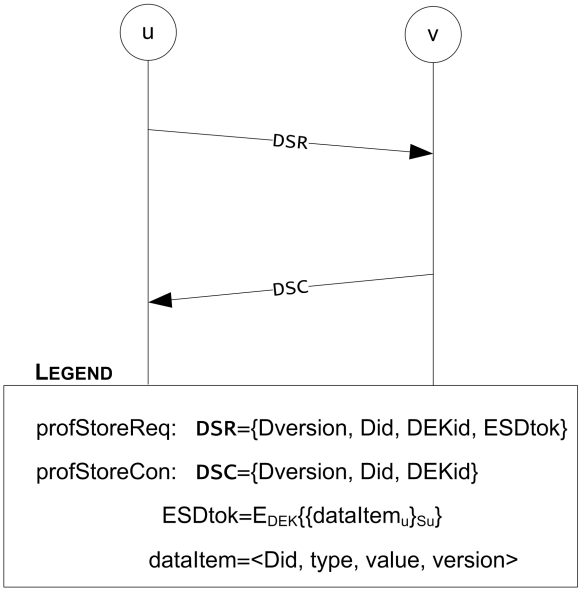

Figure 5.10: Profile data storage for $\mathcal{V}$.

5.5 Summary

In this chapter we pointed to the centralized architecture of existing on-line social networks as the key privacy issue and suggested a solution that aims at avoiding any centralized control. Our solution, namely Safebook, is an on-line social network based on a peer-to-peer architecture. Thanks to its fully distributed nature, the peer-to-peer architecture inherently avoids centralized control by any potentially malicious service provider. In order to cope with the lack of trust and lack of cooperation that plague peer-to-peer systems and to assure basic privacy among the users of the social network, Safebook leverages the trust relationships that are part of the social network application itself. Privacy in basic data access and exchange operations within the social network is achieved thanks to an anonymization technique based on multi-hop routing among nodes that trust each other in the social network. Similarly cooperation among peer nodes is enforced based on hop-by-hop trust relationships derived from the social network itself.

Chapter 6

Performance of the Approach

The new architecture described in chapter 5 raises new challenges with respect to performance. This chapter therefore presents a performance evaluation, and analyzes the feasibility of Safebook.

We evaluate the feasibility of Safebook in an incremental way. First, we evaluate the probability of reaching at least one mirror in order to retrieve data, based on the behavior of users (on-line probability) and the privacy degree. We further focus on the feasibility of a real Matryoshka graph since data can be large and therefore be partitioned. Finally, we evaluate the performance of the underlying data storage and data availability mechanisms.

Preliminary discussion As explained in the previous chapter, Safebook allows each core to set two parameters $Span$ and h for the purpose of building its Matryoshka. Such parameters indicate to mirrors and prisms the number of next hops to be selected, and the number of shells to be built, respectively. However, since a mirror or a prism can select an arbitrary number of next hops, and may also decrease $TtlMatr$ by 1 or more (or may not decrease it at all), Matryoshkas are dynamic structures with varying branching and variable number of shell for each branch.

Nevertheless, for the purpose of our analysis, we will assume Matryoshka as static structures with h shells for each branch. In this setting, with $Span > 1$, different groups of entrypoints share the same predecessor. A malicious user with extra knowledge could derive then the cut set of the entrypoints' contact lists it obtained, thus generating a good estimate for some of the nodes on the first hidden shell. Therefore, we will also assume $Span = 1$.

6.1 Mirror reachability - building one chain

In order to retrieve a target Safebook user's data, a requester contacts the entrypoints of the target user's Matryoshka. Each entrypoint then forwards the request to the next hop in the predefined path in the Matryoshka graph. Since the data retrieval is successful only if all nodes in the path are on-line at the same time, we first evaluate the probability of building one chain and further compute the residual lifetime of a chain. Reaching one mirror strongly depends on the length of the chain which basically corresponds to the number of shells h of the Matryoshka graph. Since h also plays a role on the privacy degree of the application, we analyze the impact of its increase on the performance assuming a homogeneous behavior of the nodes in terms of their on-line probability p and their average number of friends $\overline{f}$.
The probability of building a $h-1$ hop chain, p_{chain}, connecting a mirror to an entrypoint in the h-th shell is defined by the probability for each node in the chain, excluding the entrypoint, to consecutively find at least one online friend among its friends. Since the probability of finding at least one online friend among an average number $\overline{f}$ of friends is defined by a binomial distribution, the mirror reachability is defined by the following equation:

$$p_{chain} = \left[\sum_{j=1}^{\overline{f}} \binom{\overline{f}}{j} p^j \left(1-p\right)^{\overline{f}-j} \right]^h \tag{6.1}$$

In addition to the computation of the online probability of h nodes, the residual lifetime of the chain, during which data retrieval is performed, should be evaluated. Assume S_{on}, S_{off} are random variables drawn by the distributions $On\left(x\right)$, $Off\left(x\right)$ of online and offline session times of a single node respectively, and R is a random variable from the residual lifetime distribution $Res\left(x\right)$. Authors in [123] define the probability of reaching a lifetime t

by the following equation:

$$Pr\left[R < t\right] = \frac{1}{E\left[S_{on}\right]} \int_0^t \left(1 - Pr\left[S_{on} < u\right]\right) du \tag{6.2}$$

Since all nodes in one chain are assumed to be online at the same time, we define the residual lifetime Rc_h of a $(h-1)$-hops chain as the minimum node residual lifetime Rn_i among all the h nodes involved in that chain. Hence:

$$Rc_h = min\left\{Rn_1 \ldots Rn_h\right\} \tag{6.3}$$

In [70], authors conducted some measurements of online and offline session times of users using the Skype application[1]. Figure 6.1 plots the residual lifetime deriving from equation 6.2 based on this real data set: since R is stochastically larger than S_{on}, the lifetime of a newcomer is likely to be lower than the residual lifetime of an already online node.

We compute the online node probability p as it is defined in [123]:

$$p = \frac{E\left[S_{on}\right]}{E\left[S_{on}\right] + E\left[S_{off}\right]} \tag{6.4}$$

Based on this equation and the length h of a chain, we evaluate the residual lifetime of a chain using simple Monte Carlo techniques on the distribution $Res\left(x\right)$. Figure 6.2 shows the result based on different values of h. From this analysis, we conclude that the lifetime of a chain rapidly decreases with the increase of h. For example, a 3-hop chain composed by 4 nodes will keep being online for at least 19.2 minutes with 90% of probability. As a result, h should be as large as possible to enforce privacy but small enough to offer a better performance in terms of reachability.

6.2 Data availability - Matryoshka feasibility

In order to ensure both data availability and reliability in Safebook several mirrors have to be reached. Indeed, data can be partitioned or replicated. We therefore extend our feasibility analysis to the complete Matryoshka graph, with respect to the users' online probability and the number of shells h.

[1] We rely on this data since Skype, as Safebook, operates on a P2P model and its application client, once executed, runs in the background.

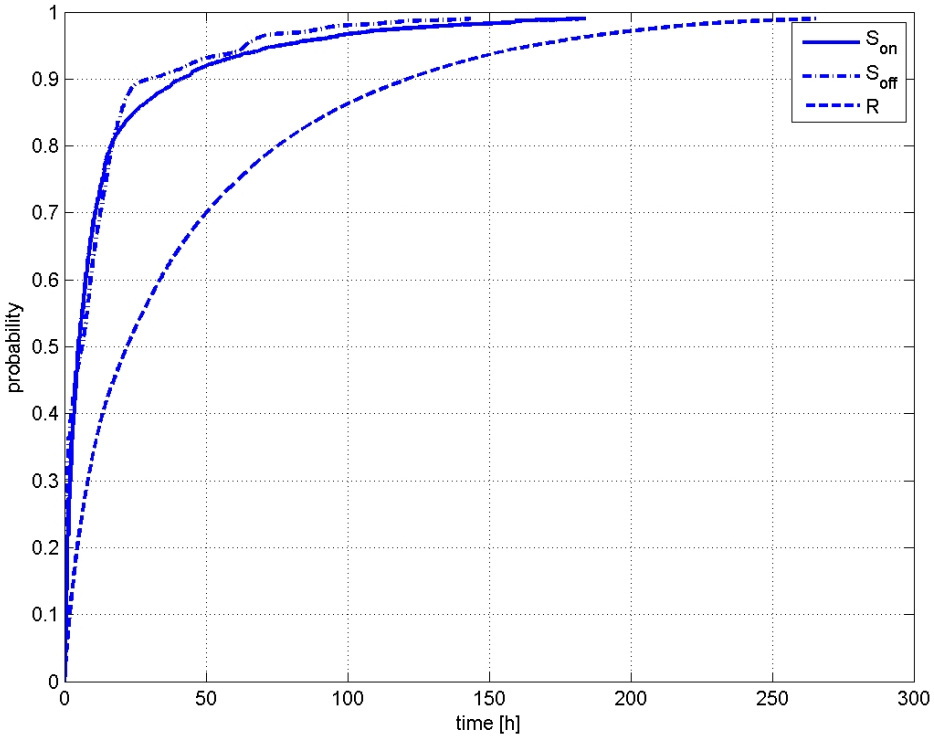

Figure 6.1: online, offline and the corresponding residual life distributions derived from the Skype dataset

Reaching at least m mirrors requires the construction of m independent chains. Based on the probability of building a single chain already defined in equation 6.1, since m mirrors among $\overline{f}$ have to be online, the probability of building a complete Matryoshka with depth h is computed as follows:

$$p_{Matr} = \sum_{i=m}^{\overline{f}} \binom{\overline{f}}{i} p^i (1-p)^{\overline{f}-i} \left[\sum_{j=1}^{\overline{f}-1} \binom{\overline{f}-1}{j} p^j (1-p)^{\overline{f}-j-1} \right]^{h-1} \tag{6.5}$$

In this equation, we assume that friends of each node are chosen independently. However, while choosing the next hop, there is a chance of choosing a friend who is already involved in the Matryoshka. Therefore, while defining p_{Matr}, we introduce a new parameter $\overline{f}_l$, which

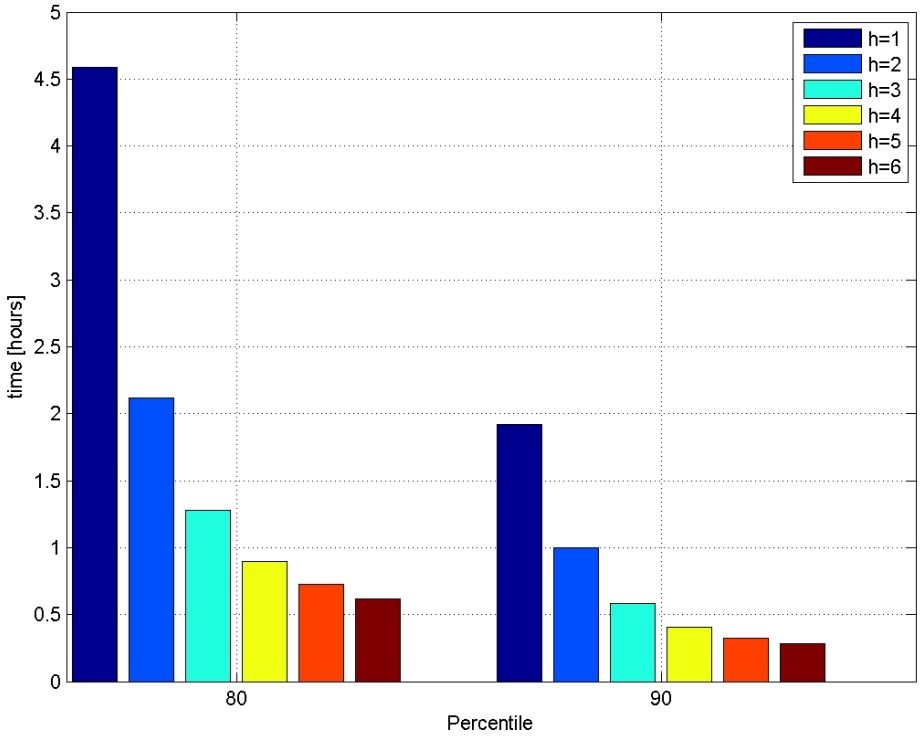

Figure 6.2: Chain residual lifetime with respect to h

corresponds to the average of eligible friends of one node at shell l. We then have:

$$p_{Matr} = \sum_{i=m}^{\overline{f}} \binom{\overline{f}}{i} p^i (1-p)^{\overline{f}-i} \prod_{l=1}^{h-1} \left[\sum_{j=1}^{\overline{f}_l} \binom{\overline{f}_l}{j} p^j (1-p)^{\overline{f}_l - j} \right] \tag{6.6}$$

Therefore, p_{Matr} depends on p, h, m and $\overline{f}_l$. Evaluating $\overline{f}_l$ is not trivial; nevertheless we distinguish the best and worst $\overline{f}_l$ by taking the overlapping ratio between friend lists into account. In the best case, with no overlapping between friend lists, we have $\overline{f}_l = \overline{f} - 1$, while in the worst case, with full overlapping between friend lists, we have $\overline{f}_l = \overline{f} - ml$.

The probability p_{Matr} is evaluated through some experiments where the online probability is set to $p = 0.53$ (based on equation 6.4 and the Skype dataset[2]), and the average

[2] http://www.cs.uiuc.edu/homes/pbg/availability

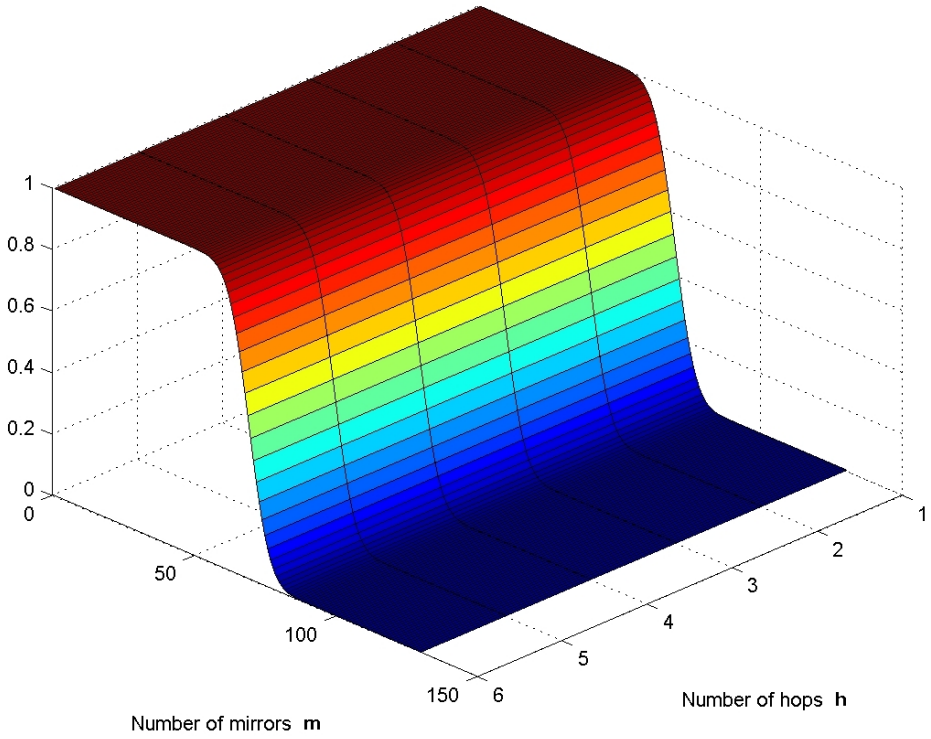

Figure 6.3: Data availability where $\overline{f} = 130$, $p = 0.53$ and $\overline{f_l} = \overline{f} - 1$ (no overlapping between friend lists

number of friends is evaluated as $\overline{f} = 130$[3]. The results in both cases are shown in figure 6.3 and figure 6.4. In figure 6.3, illustrating the best case where there is no overlapping between friend lists, p_{Matr} does not depend on h, therefore the equation can be simplified to:

$$p_{Matr} = \sum_{i=m}^{\overline{f}} \binom{\overline{f}}{i} p^i (1-p)^{\overline{f}-i} \tag{6.7}$$

On the contrary, in the worst case, as shown in Figure 6.4, h plays an essential role. The effect of overlapping between friend lists thus has a major impact on the performance. Such an impact will be evaluated in the next chapter.

[3] According to Facebook statistics, the average number of friends is 130[3].

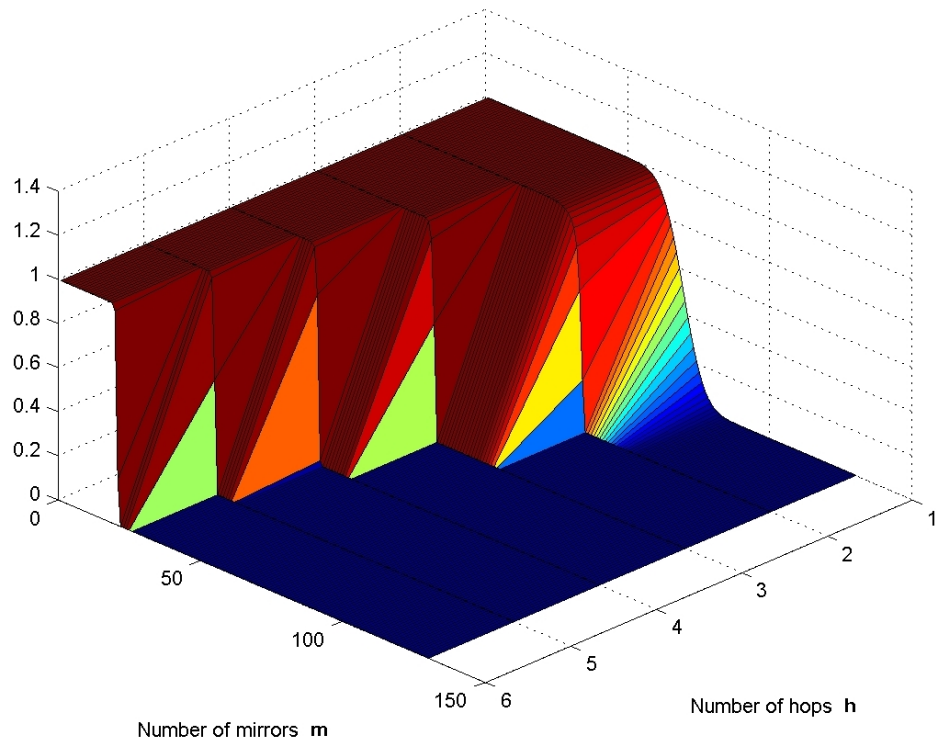

Figure 6.4: Data availability where $\overline{f} = 130$, $p = 0.53$ and $\overline{f_l} = \overline{f} - ml$ (full overlapping between friend lists

6.3 Data storage and availability

Given the new privacy preserving communication architecture, we would like to evaluate the amount of data that a peer is required to serve in order to achieve an efficient data retrieval operation.

As illustrated in chapter 5.4.3, at every profile request $profReq$ one mirror replies with a $profRep$ message. Such message contains one or more encrypted signed data tokens $ESDtok$, i.e. data items signed by the core and encrypted with a symmetric DEK. In the rest of this analysis, we consider the worst case in which only one $ESDtok$ is contained in each $profRep$.

In order to propose an efficient resource management solution for Safebook, the following questions should be answered:

- how to replicate data? can we fragment it?

- how much should one fragment size be?
- how large can the retrieved *ESDtok* be?

Notwithstanding the significant amount of research on data storage and reliability in P2P networks [30, 2, 81, 38], we propose an initial evaluation of the load at each peer using a very simple redundancy mechanism.

We propose to implement a simple parity encoding mechanism [100] where each *ESDtok* is partitioned into n blocks and for a given amount of redundancy these blocks are coded in $n + l$ fragments such that any n fragments are sufficient to reconstruct the original object. As a result of the previous section, since a node can contact m online mirrors in order to reconstruct the core's *ESDtok*, we set $n = m$. Moreover, assuming that all f friends store one fragment, the number of additional fragments is set to $l = f - m$.

6.3.1 Maximum fragment size evaluation

The amount of data a peer is required to serve for each of its friend depends on the number of requests it receives, the residual life time of the chain and its capacity.

We first define $X(t)$ as the random variable corresponding to the number of requests that are served by a peer during time t. With probability α, $X(t)$ does not exceed a value $X_t^{(\alpha)}$:

$$p\left(X(t) \leq X_t^{(\alpha)}\right) = \alpha \tag{6.8}$$

We assume that the request rate originating from one user for a single profile follows a Poisson process with rate λ. Since a node is involved in q Matryoshkas, the total request rate follows a Poisson process with rate $\lambda q f_{on}$ where f_{on} is the number of online friends and thus who can send the request. Therefore, the maximum number of requests $X_R^{(\alpha)}$ that are encountered with probability α during the residual life time R of a chain respects the following condition:

$$p\left(X(R) \leq e^{-\lambda \times q \times f_{on} \times R} \sum_{k=0}^{X_R^{(\alpha)}} \frac{(\lambda \times q \times f_{on} \times R)^k}{k!} \leq X_R^{(\alpha)}\right) = \alpha \tag{6.9}$$

The last parameter which is required to compute the size of a data fragment is the capacity of the user. We assume that each node $\mathcal{N}_i$ has a fixed capacity c_i which represents the bandwidth in this particular environment. As previously mentioned, a peer node $\mathcal{N}_i$

involved in the Matryoshka of another peer $\mathcal{N}_j$ serves requests originating from $\mathcal{N}_j$'s friends. In order to serve the corresponding $X_R^{(\alpha)}$ requests, the fragment size $s(b_i)$ should be computed as follows:

$$s(b_i) = \frac{c_i \times R}{X_R^{(\alpha)}} \tag{6.10}$$

We consider that both the capacity is identical for any node in the network and we set $\forall i \ c_i = c$. Thus, the size of a data fragment in a homogeneous network is defined by the following equation:

$$s(b) = \frac{c \times R}{X_R^{(\alpha)}} \tag{6.11}$$

6.3.2 Retrieved data evaluation

Once the size of each fragment is computed based on the capacity of each user, the residual lifetime of chains and the request rate, the maximum size of an *ESDtok* can easily be computed using the underlying reliability parameters. Hence, as previously stated, the number of fragments n which is sufficient to reconstruct the whole content, that is, the requested data, is equal to the number of mirrors m estimated based on the online probability of users. Thus, the size of an *ESDtok*, that is $s(ESDtok)$, is computed as follows:

$$s(ESDtok) = m \times s(b) \tag{6.12}$$

If we take into account the expected value of $X(R)$, from 6.11 we have:

$$s(ESDtok) = m \times \frac{c \times R}{E\left[X(R)\right]} = m \times \frac{c}{\lambda \times q \times f_{on}} \tag{6.13}$$

However, since the number of online friends corresponds to the number of mirrors for a user's Matryoshka, $m = f_{on}$. Then we can simplify equation 6.13 and obtain:

$$s(ESDtok) = \frac{c}{\lambda \times q} \tag{6.14}$$

A user $\mathcal{V}$ takes part in several friend of friends Matryoshkas, depending on the value of h.More precisely, $\mathcal{V}$ takes part in each of her m^h online friend of friends with probability $\frac{1}{m^{h-1}}$[4].

[4]We assume a single next hop can always be found in the Matryoshka creation so that equation 6.7 can

Therefore, q can be further related to the Matryoshka parameters m and h as follows:

$$q = \sum_{i=1}^{h} \frac{1}{m^{h-1}} \times m^h = m \times h \tag{6.15}$$

We thus have:

$$s(ESDtok) = \frac{c}{\lambda \times m \times h} = \frac{c}{\lambda \times f_{on} \times h} \tag{6.16}$$

Then, even if with a high f_{on} the possibility to build Matryoshka chains increases, the traffic in the network also increases, therefore the size of data decreases that can reliably be retrieved by the data requester in a residual time of R.

6.3.3 Profile size evaluation

For every data request *dataReq*, a data reply *dataRep* cannot exceed $s(ESDtok)$ without incurring in the risk of connection breakdown due to chain residual lifetime expiration[5]. However, multiple *dataReq* can be triggered[6] and several $ESDtok$, thus several data items, can be collected to enrich and update the profile $prof_{\mathcal{V}}$ of a target friend $\mathcal{V}$.

In section 5.3.1 we already mentioned that the total amount of profile data $prof_{\mathcal{V}}$ a user $\mathcal{V}$ can setup at her DDSS $\mathcal{S}_{\mathcal{V}}$ is dynamic, and depends on the amount of LDSS $\mathcal{L}_{\mathcal{F}_i}^{\mathcal{V}}$ each friend $\mathcal{F}_i$ of $\mathcal{V}$ allocates for $\mathcal{V}$.

If a tit-for-tat strategy is adopted, $\mathcal{V}$ benefits from a DDSS $\mathcal{S}_{\mathcal{V}}$ and makes available to friends the same space in her LDSS. The remaining $\mathcal{L}_{\mathcal{V}} - \mathcal{S}_{\mathcal{V}}$ is available for $\mathcal{V}$ to store a local copy of her generated data items, i.e. her profile. Therefore, the maximum profile size $s(prof_{\mathcal{V}})$ can be computed as:

$$s(prof_{\mathcal{V}}) = \frac{\mathcal{L}_{\mathcal{V}}}{2} \tag{6.17}$$

6.3.4 Example

We assume that the number of online friends, i.e. the number of profile requesters, ranges between the 10th and the 90th percentile, respectively f_{min} and f_{max}, of the binomial

be applied.

[5] A high value of α decreases this risk but leads to a lower $s(ESDtok)$ at the same time.

[6] Up to $\lambda \times m$ per friend every R time, even if such friend is offline, assuming her Matryoshka is connected and served by m mirrors.

distribution where the online probability for nodes is 0.53 and the average number of friends is 130. Therefore, with the 90% of probability, no less than $f_{min} = 61$ friends and no more than $f_{max} = 75$ ones will be online in Safebook.
We assume the number of shells in Matryoshkas h is 4.
We assume $\alpha = 0.9$, therefore no more than $X(R)^{0.9}$ requests will be served by a user in Safebook with 90% of probability in the residual lifetime R being previously computed, and set to 19.2 minutes.

Figure 6.5 shows, for different upload bandwidth values, the maximum size of fragments served by a user based on the number of users $q = f_{max} \times h$ for which the peer is involved in the Matryoshka and the request rate λ expressed in data requests per day. From this figure, we can conclude that, if ADSL connection with 0.5Mbps upload bandwidth is used[7], when 3 profile requests per friend (target Matryoshka) per hour are triggered (i.e. $\lambda \left[\frac{1}{hrs}\right] = 36$), the maximum size of fragments to be forwarded is around 3.4 KB. Therefore, with 90% of probability, at least f_{min} of such fragments will be collected by the requester, that will download at the end around 200KB of data (see figure 6.6).

6.4 Summary

In this chapter we have investigated the performance of Safebook. In the first part of the study, an analytical model of the feasibility of Safebook has been provided in an incremental fashion: since the correct execution of Safebook depends on the reachability of at least one mirror, the residual life time of a path between a mirror and an entrypoint was first defined and evaluated based on the Matryoshka setup parameters and the online probability of nodes. This analysis confirms the fact that choosing a large value for the depth h of the Matryoshka (which is a privacy requirement) can have a severe impact on the data retrieval and thus, h should be defined as small as possible but should still allow for a good privacy degree. Furthermore, in order to ensure data availability, the feasibility of the complete Matryoshka graph is evaluated. Since a successful data retrieval depends on the reachability of more than one mirror towards different and independent paths, the ratio of overlapping between friend lists has to be taken into account. Finally, Safebook's feasibility being confirmed, the maximum amount of data that may be retrieved at every request is also evaluated based on

[7] Standard ITU G.993.2.

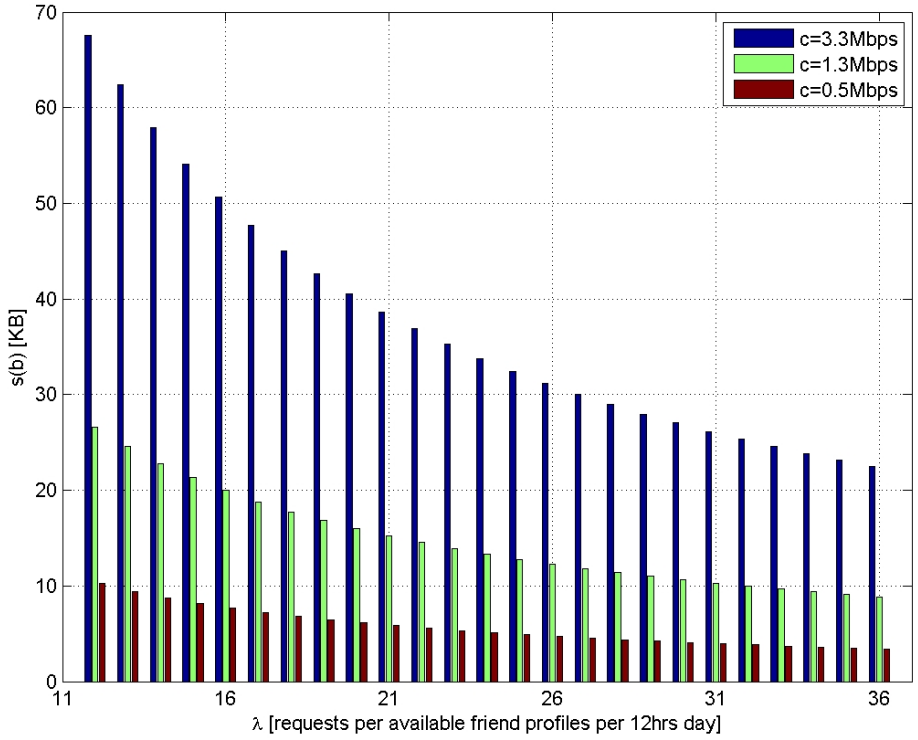

Figure 6.5: Fragment size evaluation for different upload bandwidth c with varying request rate λ.

the use of a simple redundancy mechanism.

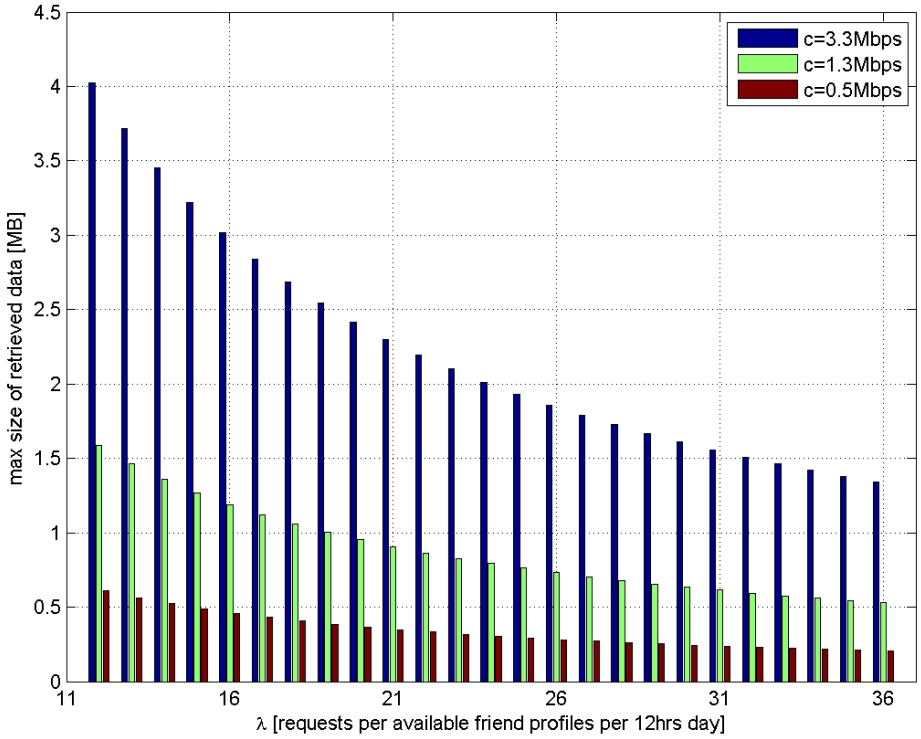

Figure 6.6: Maximum size of data retrieved at every request for different upload bandwidth c with varying request rate λ.

Chapter 7

Impact of social graphs on performance and privacy

In this chapter, we analyze the impact of social network graph topology on security and performance and we prove that regardless of the particular centralized or distributed nature of the OSN, the achievable security and privacy degree strongly depends on the graph-theoretical properties of the social graph representing the real friendship relations between the users. We first observe three metrics, namely the degree, the clustering coefficient and the mixing time, and show that they give fundamental insights on the privacy degree of the OSN. We further evaluate the privacy degree of Safebook based on the previous analysis. Finally, we observe a strong trade-off between privacy and performance such that delay and reachability are inversely proportional to privacy.

7.1 Privacy from the graph theory perspective

An Online Social Network can be represented as an undirected ***social graph*** $G(V, E)$ comprising a set V of users and a set E of edges representing social ties, such as friendship, kinship, trust and the like.

In this section, we first remind the definition of three main characteristics of graphs

and compute them for existing social graphs. These characteristics are the ***node degree***, the ***clustering coefficient***, and the ***mixing time***. The impact of the evolution of these characteristics is also evaluated based on existing social graphs: in September 2005, Facebook published anonymous social graphs of 5 universities in the United States[1]: California Institute of Technology (Caltech), Princeton University (Princeton), Georgetown University (Georgetown), University of North Carolina (UNC), Oklahoma University (Oklahoma). Each graph is represented by an adjacency matrix A whose non diagonal elements a_{ij} are set to one if user $\nu_i \in V$ is a friend of user $\nu_j \in V$, or zero otherwise. As each adjacency matrix is symmetric, the represented social graph is undirected.

7.1.1 Node degree

In graph theory, the degree of a vertex ν, denoted by $deg(\nu)$ is defined as the number of edges incident to the vertex. Since in a social graph $G(V, E)$ a vertex represents a user and the edges represent social links such as friendship, acquaintanceship etc., a user's degree corresponds to the number of contacts a user has. This degree has a direct impact on privacy since with the increase of the degree the number of contacts increases, hence the probability of connecting to a misbehaving user increases. Therefore, the impact of the degree of a node on security can be evaluated by computing the probability of having at least one misbehaving contact.

Assume p_{mal} denotes the probability a user η is malicious, and assume the events of befriending a malicious user are independent. The number of malicious contacts $\mathcal{F}_{mal}(\nu)$ of ν then follows a binomial distribution with parameters p_{mal} and $deg(\nu)$:

$$\mathcal{F}_{mal}(\nu) \sim B(p_{mal}, deg(\nu))$$

In particular, the probability p_ν of having at least one misbehaving contact is:

$$p_\nu = 1 - (1 - p_{mal})^{deg(\nu)} \tag{7.1}$$

Once a malicious contact η gets access to ν's sensitive data because of the simple befriending operations, η can disclose them out of band, or inside the social network itself. In this latter case, the disclosure targets, among all η's friends, the common contacts between η and ν,

[1] http://people.maths.ox.ac.uk/ porterm/data/facebook5.zip

and can turn out to severely damage ν.

Therefore, the outdegree of a node is directly related with usage control (see section 3.1.1). The more a node has friends, the larger the probability of having a malicious friend which can disclose sensitive personal data.

Figure 7.1 shows the distribution of the node degree for the five Facebook datasets. We observe the degree of the Caltech social network is much lower than the degree of the other four social networks. This is probably due to the fact that the Caltech dataset is significantly smaller than the others, and, as a consequence, the opportunities to add friends are lower. Assuming that the probability of choosing a malicious friend is the same for all the five graphs, then Caltech network would be the most secure network given its lowest node degree. Indeed, following eq.7.1, we observe that the probability of having at least a misbehaving contact is lower in the Caltech network. Table 7.1 shows in Caltech, when p_{mal} is set to 0.01, p_ν is on average as high as 0.35, while in the other networks this value ranges from 0.59 to 0.64.

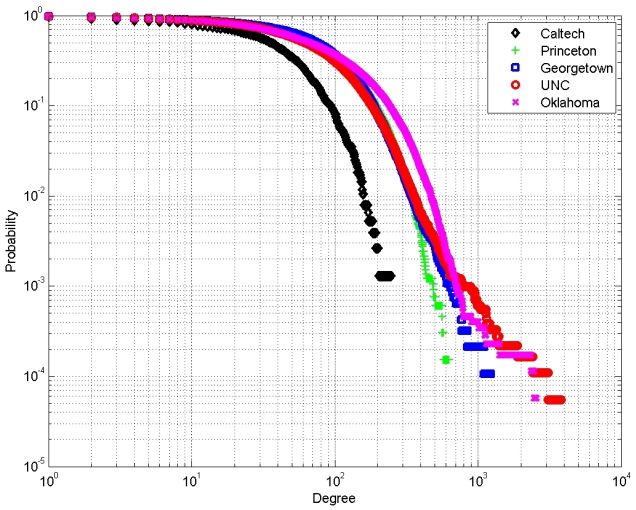

Figure 7.1: Log-log plot of the degree complementary cumulative distribution of real-life social networks.

7.1.2 Clustering Coefficient

In an undirected graph, the clustering coefficient $c(\nu)$ of a node ν with $deg(\nu)$ edges is defined as the number of existing links between these nodes, denoted as $e_{deg(\nu)}$, divided by

the number of all possible links which by definition is $\left(\frac{deg(\nu)(deg(\nu)-1)}{2}\right)$. We therefore have:

$$c(\nu) = \frac{2e_{deg(\nu)}}{deg(\nu)(deg(\nu)-1)} \tag{7.2}$$

The clustering coefficient of the overall graph denoted as $C(G)$ is defined as the average clustering coefficient of all nodes in the graph, hence:

$$C(G) = \frac{\sum_{\nu \in V} c(\nu)}{\|V\|} \tag{7.3}$$

Computing or estimating the clustering coefficient of a graph can give an idea on the impact of the propagation of unauthorized information by malicious users on nodes friendship. Once a malicious node, η, is added in the contact list of ν, η can access ν's sensitive data, and disclose it indiscriminately using the social network facilities like wall posting, picture publishing and the like. In particular, if η clones a user profile ν strongly trusts, all sensitive data that ν shares with η will be disclosed.

Such an impact can be measured by computing the average ratio $\mathcal{Q}_\nu$ of ν's friends which can obtain sensitive information disclosed by a malicious η as follows:

$$\mathcal{Q}_\nu = p_\nu c(\nu) \tag{7.4}$$

From this equation, we conclude that the degree of propagation is proportional to the clustering coefficient. The tighter the friendset, the broader the disclosure of sensitive data to the user's contacts.

Figure 7.2 shows the distribution of the clustering coefficient for the different social networks that were previously introduced with respect to the degree of the graph since $\mathcal{Q}_\nu$ depends both on p_ν and $c(\nu)$. Similarly to the previous analysis, the clustering coefficient of the Caltech social network strongly differs from those of other networks, as it is almost twice in size. This is probably due to the small size of the Caltech dataset. A smaller community is in fact more likely to be tightly knit.

We observe that in case a friend misbehaves, the victim in the Caltech social graph exposes his sensitive data to a ratio of friends two times higher compared with the one of a victim in the other networks. Nevertheless, due to the lower p_ν derived from the graph degree, the average ratio $\mathcal{Q}_\nu$ does not strongly vary in all networks, ranging from 0.11 to 0.14.

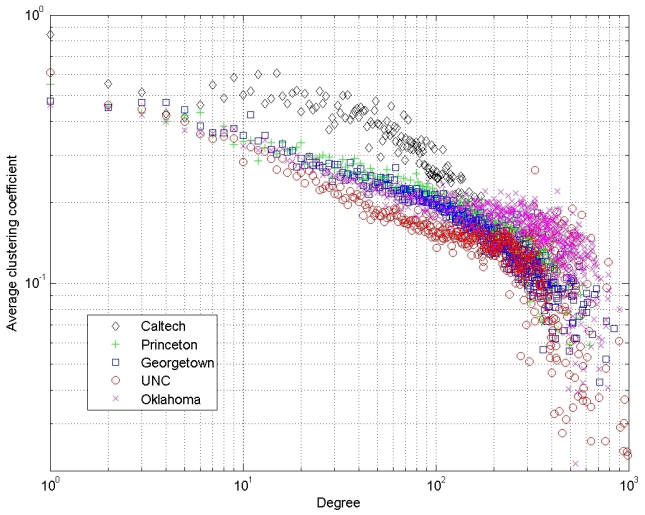

Figure 7.2: Average clustering coefficient of real-life social networks with respect to node degree.

7.1.3 Mixing time

Random walks [90] in a graph have an important property: when the random walk approximates its steady state distribution after a sufficient number of hops, the startpoint and endpoint of the walk are uncorrelated. This number of hops is called ***mixing time***, and the smaller it is, the faster the abovementioned property is met.
We will introduce the mixing time starting from the steady state distribution.
The ***steady state distribution*** [83] for a node θ represents the probability that a random walk reaches θ after a sufficient number of hops no matter where this random walk originated from:

$$ssd(\theta) = \frac{deg(\theta)}{2\|E\|} \tag{7.5}$$

The mixing time [90] $\tau_x(\epsilon)$ is then computed as follows:

$$\tau_x(\epsilon) = min\{h : \Delta_x(h) \leq \epsilon\} \tag{7.6}$$

$\Delta_x(h)$ the the variation distance between the random walk distribution $R^h(x)$ after h hops,

and the steady state distribution $ssd(x)$:

$$\Delta_x(h) = \|R^h - ssd\| = \frac{1}{2}\sum_{x \in V} \|R^h(x) - ssd(x)\| \tag{7.7}$$

For the complete social graph, the mixing time is:

$$\tau(\epsilon) = \max_{x \in V} \tau_x(\epsilon) \tag{7.8}$$

In social networks, mixing time is varying widely: in [60] authors found that mixing time is much higher in social networks where links represent face-to-face interactions. Recently, further measurements [91] confirmed this concept. The mixing time of a social network graph is directly related with both profile integrity and communication untraceability[2]. Figure 7.3 plots the mixing time $\tau(\epsilon)$ of each of the five Facebook social graphs for different values of a predefined maximum variation distance ϵ. As the Caltech network presents a faster mixing time, solutions leveraging on random walks on the social network graph would perform better if applied on the Caltech network rather than in the Georgetown one, whose mixing time is approximately five times higher.

7.1.4 Results

Table 7.1 summarizes the main topological properties of the analyzed social network dumps, where the probability p_{mal} of befriending a misbehaving user is set to 0.01. In this scenario, even if on the average p_ν is high (ranging from 0.35 to 0.64), the UNC network ensures the best privacy protection (in terms of anonymity and usage control) with respect to the other networks because it shows the lowest average value for Q_ν. In terms of communication untraceability and profile integrity, the Caltech network provides the best protection due to the faster mixing time.

Regardless of the particular centralized or distributed architecture, the above mentioned findings are relevant for any security solution in OSN.

[2]We presented such properties in section 3.1.1.

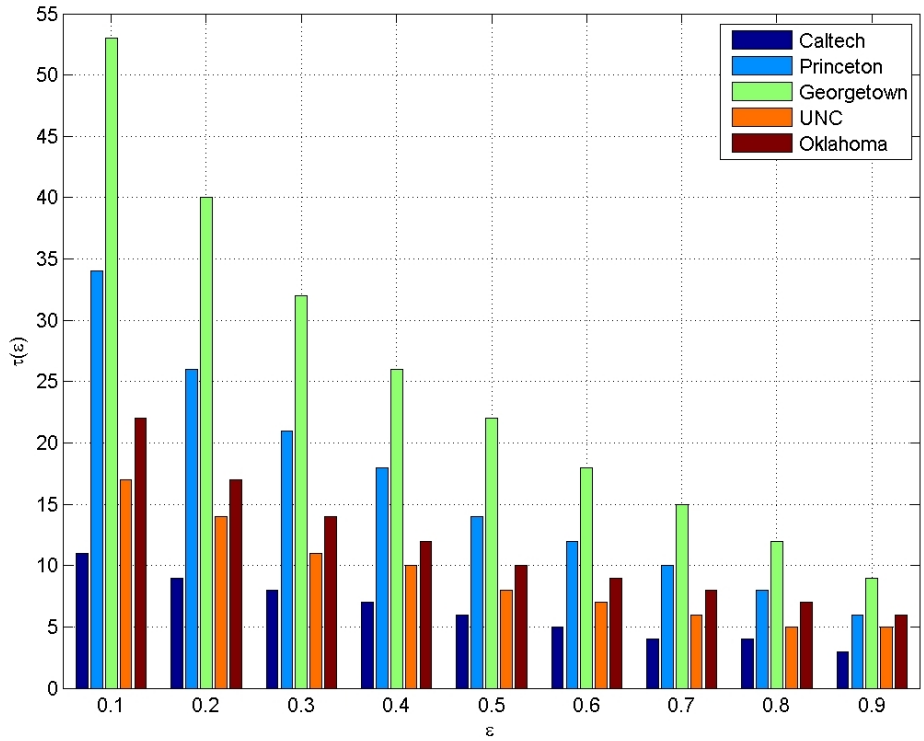

Figure 7.3: Mixing time of real-life social networks.

7.2 Impact of social graphs on Safebook

This section focuses on the particular architecture proposed as Safebook, and further investigates the impact of social graph topology on privacy. However, the social graph topology also have an intrinsic impact on the performance of Safebook, since according to the second design principle thereof (see section 5.1.1) peer nodes are connected depending on their maintainers' real life trust. This section also analyzes such impact.

7.2.1 Impact on privacy

In Safebook, the social network characteristics play an important additional role on privacy: a malicious user $\mathcal{M}$ can guess a core $\mathcal{V}$'s friend list if in the underlying social network the ratio of common friends between $\mathcal{M}$ and $\mathcal{V}$ is very high.

Such a ratio u_h does not decrease indefinitely with the increase of the number of hops in the social network graph h required to connect $\mathcal{M}$ to $\mathcal{V}$, but rather converges to a value

	$\|V\|$	$\overline{deg(\nu)}$	$C(G)$	$\overline{p_\nu}$	$\overline{\mathcal{Q}_\nu}$	$\tau(0.1)$
Caltech	769	43.32	0.41	0.35	0.14	11
Princeton	6596	88.93	0.24	0.59	0.14	34
Georgetown	9414	90.43	0.22	0.60	0.13	53
UNC	18163	84.44	0.20	0.57	0.11	17
Oklahoma	17425	102.44	0.22	0.64	0.14	22

Table 7.1: Main characteristics of five social graphs from Facebook ($\overline{p_\nu}$ computed assuming p_{mal}=0.01).

u_∞.

Assume to start a random walk connecting a user $\mathcal{V}$ with a user $\mathcal{M}$. Assume the number of hops h_∞ of such a random walk is sufficiently high to reach its steady state distribution.

Given x the number of friends of $\mathcal{V}$ and y the number of friends of $\mathcal{M}$, we can compute u_∞ as follows:

$$u_\infty = \sum_{x,y} \frac{xy}{\overline{f}\|V\|} f(x)\, s(y) \tag{7.9}$$

$f(x)$ and $s(y)$ are the density functions of x and y respectively. Since $s(y)$ corresponds to the steady state density function resulting from a random walk, $s(y)$ is defined as follows:

$$s(y) = \sum_{z \in V} \{ssd(z) \| f(z) = y\} = \frac{yf(y)}{\sum_{j \in V} jf(j)}$$

Thus, equation 7.9 defines the number of all common edges between the first node and the last one divided by the number of all the edges of the graph $\|E\| = \frac{\overline{f}\|V\|}{2}$.

Table 7.2 reports the value of u_∞ for all the Facebook datasets. Again, due to the high clustering, Caltech network offers the worst privacy level.

7.2.2 Impact on performance

Social graph topology have a strong impact on data availability in Safebook.

As discussed in Chapter 6.2, the probability of building a complete Matryoshka (see equation 6.6) depends on the probability of finding a sufficient number of online friends that can act as mirrors and on the probability that each mirror will manage to build a chain of $h-1$ hops where h is the desired number of shells. Since a user cannot take two positions

in a single Matryoshka, the probability of building chains may drastically decrease in case of low node online probability and high overlapping factor between friend lists.

The strong clustering coefficient of the Caltech network (see table 7.1) suggests Safebook will not perform as well as in the other four Facebook datasets. We simulated the creation of 4-shells Matryoshkas in a challenging environment where the on-line probability of nodes p has been set to 0.1. Results reported in table 7.2 confirm our belief: in Caltech, the ratio between the chains successfully built ch and the average number of friends $\overline{f}$[3] is much lower than in the other datasets.
This simulation also confirms the number q of Matryoshkas a node participates in is almost h times higher than the average number of reachable mirrors, thus, chains, in the system (see eq. 6.15).

	$\overline{f}$	ch	q	$\frac{ch}{\overline{f}}$	$\frac{q}{ch}$	u_∞
Caltech	43.32	2.35	9.4	0.05	4.00	0.097
Princeton	88.94	9.06	36.24	0.10	4.00	0.024
Georgetown	90.43	8.59	34.37	0.10	4.00	0.017
UNC	84.44	8.18	32.73	0.10	4.00	0.010
Oklahoma	102.44	10.32	41.26	0.10	4.00	0.013

Table 7.2: Characteristics summary of examined SN graphs.

7.2.3 Performance and privacy trade-off

As stated in chapter 5.1.1, since the data storage operation strongly depends on some sensitive information such as the list of users' friends, there is a strong link between the depth h of a Matryoshka and the privacy degree: indeed, h should be as large as possible to prevent a malicious requester from discovering these friendship information and achieve a good privacy level.

While increasing the number of shells decreases the chance of discovering one core's contact list, on the other hand, messages should still follow a $(h-1)$-hop path to reach the mirror (see section 6.1). During the data retrieval process, all nodes along this path are required to be simultaneously on-line for a sufficient amount of time. Increasing h

[3]The value of $\overline{f}$ corresponds to the average degree of the nodes in the social network (see table 7.1)

naturally decreases the probability of reaching one mirror and can have a severe impact on the performance of data retrieval. Hence, there is a trade-off between the privacy and the data availability depending on the value chosen for h.

We analyze the trade-off between privacy and performance in order to come up with optimal values for the Matryoshka parameter h. Assume a malicious requester $\mathcal{M}$ retrieves $\mathcal{D}$ as an entrypoint of $\mathcal{V}$, and by chance $\mathcal{D}$ is one of $\mathcal{M}$'s friends. In this case, when $h = 1$, $\mathcal{M}$ will derive $\mathcal{D}$ is also a friend of $\mathcal{V}$. Assume h_∞ as the minimum number of hops to make the ratio of common friends between $\mathcal{D}$ and $\mathcal{V}$ $u_{\mathcal{DV}}$ approach u_∞. When $h < h_\infty$ and $\mathcal{M}$ has access on $\mathcal{D}$'s contact list[4], $\mathcal{D}$'s contacts are more likely to $\mathcal{M}$'s ones to be $\mathcal{V}$'s contacts too. Finally, when $h > h_\infty$, even assuming $\mathcal{M}$ has access on $\mathcal{D}$'s contact list, $\mathcal{M}$ will not derive any additional information from any retrieved entrypoint of $\mathcal{V}$.

We run several simulations where, for each user, a h-shell Matryoshka with varying h is built, and based on this set-up the ratio u_h of common friends between the entrypoint θ_h and the core $\mathcal{V}$ is evaluated. Figure 7.4 shows the results with respect to the distance between the two nodes. We observe that h_∞ is lower for social network with faster mixing time, while u_∞ is in line with the values computed from eq. 7.9. Hence, a Matryoshka with a depth higher than h_∞ will not increase the privacy level of $\mathcal{V}$ noticeably.

To summarize, the highest privacy degree that can be reached given a social network is the one where $h = h_\infty$: increasing h after this optimal value does not have an impact on the privacy level anymore. Furthermore peer-to-peer based OSN applications implemented over social networks with high clustering coefficient and slow mixing time unfortunately show a lower privacy degree with respect to fast mixing networks without strong local clustering.

Privacy preserving OSN architectures, including Safebook, should address this problem by discouraging the indiscriminate action of adding friends. Moreover, the OSN should guarantee the fast mixing property to the network. This can be done by ensuring the small world property of the social network graph, and encouraging 'long links' connecting different clusters together, otherwise most of the random walks would be confined to the originating cluster.

[4]In Safebook, user may not decide to share their contact list, or share it with limitations.

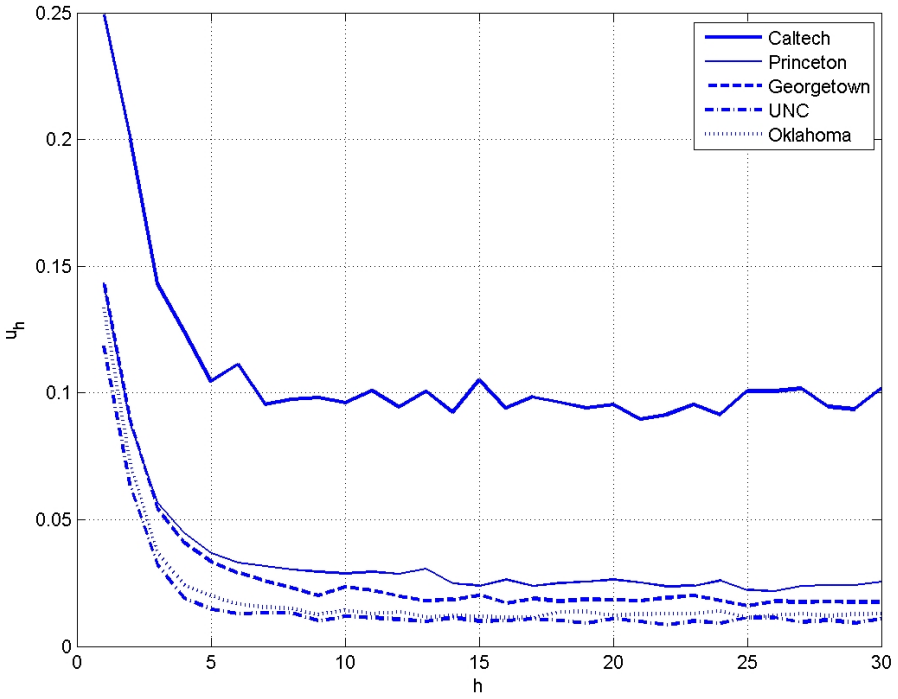

Figure 7.4: Ratio of common friends between two nodes $\mathcal{V}$ and θ^h at social distance h in the social network

7.3 Summary

This Chapter investigated the strong relationship between the topological properties of the social network graph and the achievable users' privacy in centralized or distributed OSN. We observed that metrics such as the degree and the clustering coefficient of nodes severely affect users' privacy as defined in chapter 3 with respect to identity/friendship privacy and usage control, while the mixing time of random walks in the social network graph plays an essential role in preserving the users' communication untraceability.

An analysis on real social network dumps reveals that the probability of befriending at least a misbehaving contact is not negligible. In this case, the number of nodes which can discover unauthorized data depends on the number of common friends between the victim and the attacker.

Further specific analysis on Safebook confirms that the presence of strong local clustering

negatively impacts the privacy of the solution, as measured as the ratio of common friends between an entrypoint and a core in one user's Matryoshka. When choosing a large value for the depth h of the Matryoshka, which is a privacy requirement, such ratio decreases. On the other hand, a large value of h leads to the creation of a Matryoshka with fewer branches, therefore has a direct impact on data availability. However, increasing h after an optimal value h_∞ does not have an impact on the privacy level anymore. We observe that social networks with faster mixing time reach such optimal value faster.

Chapter 8

Implementation

In previous chapters we have motivated the need for new privacy preserving OSNs and we proposed Safebook, an OSN based on a distributed architecture where real life friends provide communication and storage services, ensuring user privacy. This chapter describes a first implementation of Safebook.

The current prototype of Safebook[1] consists of 50 files and 14000 lines of code, one half of the latter consisting in the python scripts running the main application, one other half consisting in HTML, CSS, and Javascript used to design the user interface.

Once the python interpreter and a series of prerequisite libraries[2] have been installed, execution of Safebook is started with the command:

```
>python safebook.py
```

Double clicking on the Safebook executable identified by the icon in figure 8.4. Due to the high availability of python interpreters for different operating systems, including Windows, Linux and MacOs, Safebook can be executed by all end-users' personal computers.

[1] The Safebook client is available for download [24] under GNU General Public License Version 3 [11].
[2] Twisted, pyOpenSSL, M2Crypto, pysqlite, PIL.

8.1 Overall Architecture

Safebook client, as depicted in figure 8.1, is composed by four different managers:

1. the ***Communication Manager***, in charge of sending and receiving packets;
2. the ***S2S Manager***, building the P2P overlay;
3. the ***Matryoshka Manager***, building the Matryoshka overlay;
4. the ***User Manager***, implementing the user interface.

This client is a multithread event-driven application: all managers send *requests* and *responses* to a dispatcher, and receive back indications or *confirmations* (internal messages). When two Safebook clients communicate, their respective communication managers send and receive PDUs (external messages) (see figure8.2).

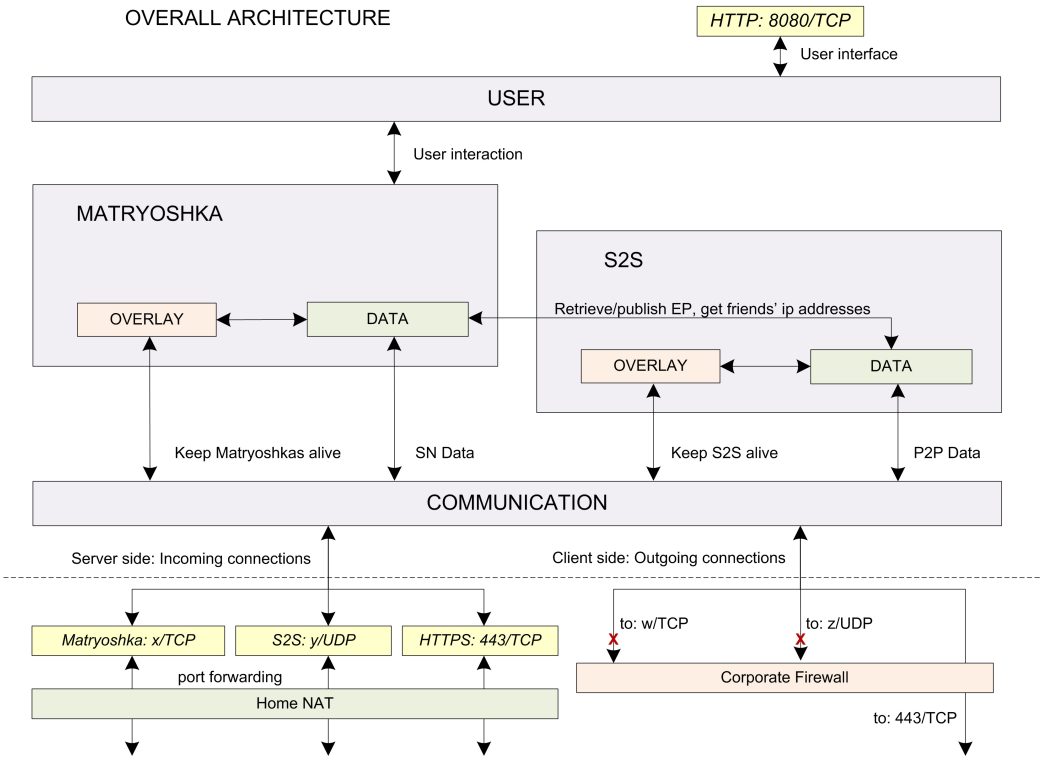

Figure 8.1: Overall architecture of Safebook.

Once started, the Safebook client spawns a dispatcher thread spawning, in turn, all the managers in the following order: Communication, S2S, User and Matryoshka. While the

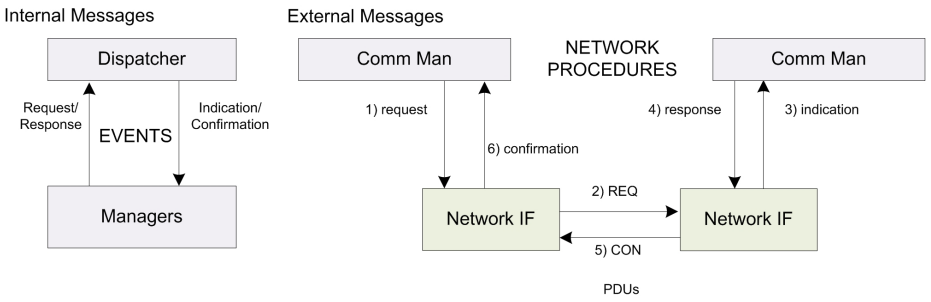

Figure 8.2: Internal (left) and external (right) message exchange in Safebook

main process runs the program console, the dispatcher dequeues internal messages and sends them to the interested managers according to a classical publish/subscribe paradigm. Once a manager dequeues a message, it runs the appropriate routine to execute a particular job, then generates a message and enqueues it to the dispatcher or sends it in the network in the case of the Communication manager.

While the Communication Manager launches a UDP and a TCP server processes at two random ports, respectively targeting the peer-to-peer and Matryoshka communication, the User Manager runs a local web server at port 8080. Similar to current SNSs, the user interface has in fact been implemented under the form of a web page. Additionally, an HTTPS server is run for the purpose of receiving Matryoshka and P2P traffic from nodes behind strict policy firewalls.

In case the Safebook host is assigned a private address, the application tries to open and forward the UDP and TCP server ports from the NAT via UPNP protocol.
When no UPNP compatible NAT device is found, the client proposes the user to manually configure port forwarding on her NAT. However, if the user is behind a corporate firewall, the user cannot operate any port forwarding operation, therefore the Safebook host cannot be reached by any node in the Internet. Moreover, under strict corporate firewall policies, outgoing UDP connections could also be filtered out, while TCP ones could be allowed to contact a predefined web proxy only. To overcome this limitation, Safebook establishes a tunnel over HTTPS to peer nodes assigned with a public IP address, or behind a home NAT[3] under their control. The network packets generated by the S2S manager (S2S traffic) and those generated by the Matryoshka Manager (Matryoshka traffic) are then sent in the

[3]Of course, when more than one Safebook host is behind the same NAT, only one of them can run a HTTPS server.

HTTPS payload, and the usual P2P and Matryoshka services are granted.

As a main limitation, being unreachable from the outside, user nodes behind corporate firewalls can build their Matryoshka and take part in those of other users, but cannot play the role of entrypoints.

8.2 Account creation

Account creation in Safebook can take place in two ways: by invitation, as explained in Chapter 5.4.1, or without, as implemented in the current prototype. While joining Safebook by invitation allows the newcomer to bootstrap her Matryoshka with the trusted inviter, in the second approach the newcomer needs to refer to an untrusted contact to build her first Matryoshka chain. Once created the first chain, the newcomer will start establishing friendship links, therefore creating new Matryoshka chains, and remove the untrusted bootstrapper from her friendlist.

The identifiers and the list of certified *DhtKey* are computed by the TIS starting from a set of the newcomer $\mathcal{V}$'s properties

$$name_v =< firstName, lastName, gender, birthDate, birthPlace, nationality >$$

The TIS holds an asymmetric key pair $\{TIS^-, TIS^+\}$ used for message confidentiality and integrity, and three master keys MK_1, MK_2, MK_3 used to compute UId_v and NId_v.

The account creation takes place in two steps: in the first one, out of band, $\mathcal{V}$ sends to the TIS his $name_v$ together with a chosen password *pwd*, and receives a symmetric key $K = h_{MK3}(name_v, pwd)$; in the second one, in band, $\mathcal{V}$ generates two keypairs $\{\mathcal{U}^-, \mathcal{U}^+\}$ and $\{\mathcal{N}^-, \mathcal{N}^+\}$ and sends the public keys $\mathcal{U}^+$ and $\mathcal{N}^+$ together with $name_v$ and *pwd* to the TIS. This message is double signed to let the TIS verify the ownership of $\mathcal{U}^-$ and $\mathcal{N}^-$. The TIS then checks the identity of $\mathcal{V}$ with a challenge, computes and certifies the user and node identifiers, and provides a set of certified *DhtKey* by applying a well known hash function to all the possible combinations of elements in $name_v$. In our prototype, since $name_v$ is composed by six elements but we don't take the gender into account, the TIS will provide $2^5 - 1$ *DhtKey*[4].
Finally, the TIS selects a random Matryoshka bootstrapper[5] and sends its certified user and node identifiers together with its IP address to $\mathcal{V}$. Once received the Matryoshka bootstrapper information, $\mathcal{V}$ sends a bootstrap request containing a friend token (see section 5.4.3) to the bootstrapper, that automatically accepts the friendship and allows $\mathcal{V}$ to start the usual Matryoshka creation procedure.

[4] we don't consider the empty set among the set of combinations.

[5] Matryoshka bootstrappers are syntetic Safebook nodes that do not belong to real users and are trusted by the TIS.

The account creation steps are resumed in figure 8.3.

8.3 User interface and OSN facilities

Once created the first Matryoshka chain, the newcomer can use Safebook facilities.

The user interface (see fig. 8.5) has been implemented as a webpage (see fig.8.5), such as all current social network services which are accessible via internet browsers. The User Manager runs a http server in localhost at port 8080 intercepting and executing the user's commands, and drawing XHTML 1.0 webpages[6] as a result.

A Safebook user $\mathcal{V}$ can browse into three main sections:

- ***Square***, containing the information shared by $\mathcal{V}$'s contacts;
- ***Podium***, containing $\mathcal{V}$'s shared information;
- ***Contacts***, containing the $\mathcal{V}$'s contact list.

The Square section is similar to the Diaspora and Google+ *stream* page, and to the Facebook *wall*. In this section, the user quickly gets updates and sends comments on the new content shared by her trusted contacts or people she is simply interested in.

The podium section hosts the user-generated information, including the subset of the wall threads where user's posts appear, the profile page containing the user's identity, the gallery page containing user's pictures, and the user's mailbox[7]

Finally, the Contacts page displays the user's contacts and offers links to the contacts' profile, wall and gallery pages.

The user may share new content with a subset of friends. Access restriction is achieved by associating the new content with one or more ***badges*** (see section 5.3.2), that may be compared to Google+ *circles*, Diaspora *aspects*, or Facebook user-defined lists. Safebook users sharing new content without specifying an associated badge are asked to perform this action as a requirement for the successful sharing. In case the user does not associate any badge, a default private one is chosen and the new information is made available for the user only and nobody else. Badges can be sent to trusted contacts at the act of friendship establishment or later. Badge reception is transparent for the recipient user, as it simply consists on the reception of new symmetric data encryption keys (DEK).

[6] http://www.w3.org/TR/2000/REC-xhtml1-20000126/

[7] The mailbox has not yet been implemented in the current prototype.

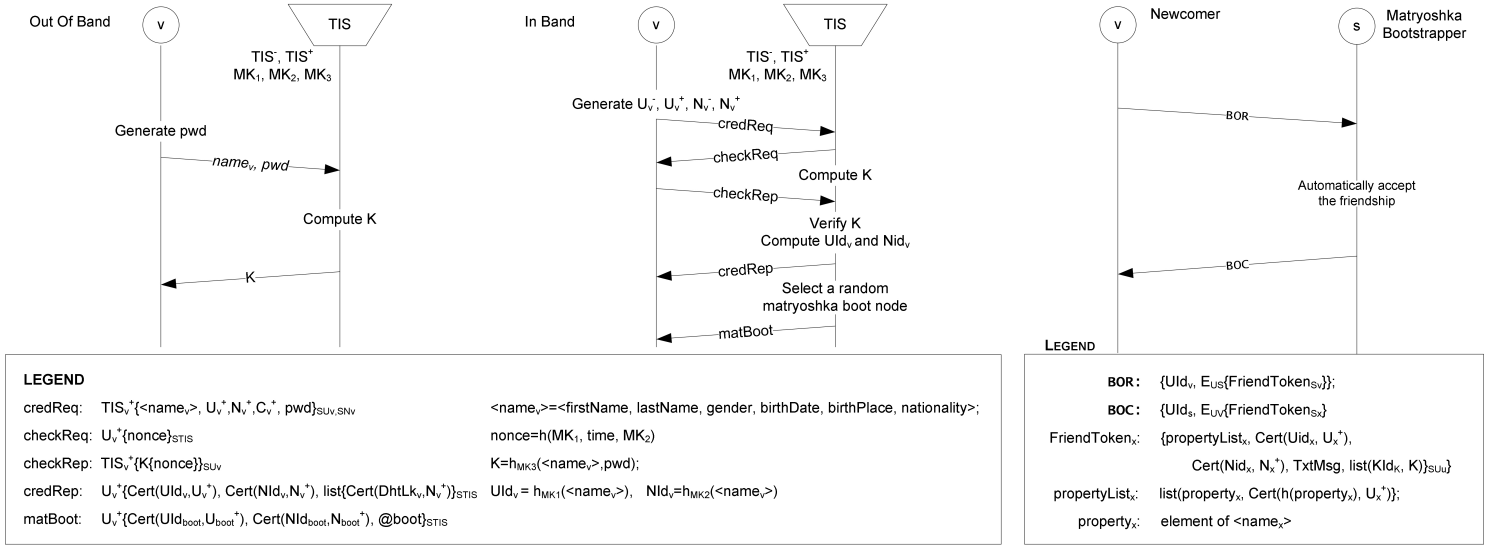

Figure 8.3: Account Creation: out of band step on the left, in band step on the right.

Friendship lookup can always be performed thanks to a form at the left of the page. The user can choose more than a keyword[8] and get the publicly available information on all users associated with these keywords. The user further selects her trusted contacts and sends them a friendship advertisement together with a list of DEKs corresponding to the associated badges.

Friendship lookup leverages on the P2P overlay of Safebook. The characteristics of the S2S Manager in charge of building such overlay are discussed in the following section.

Figure 8.4: The Safebook logo: two persons shaking hands represent the process at the basis of Matryoshka and, more generally, of Safebook.

[8] A keyword must correspond to an element in $name_v$ as explained in the previous section.

Figure 8.5: Graphical interface of Safebook: on the top-left the Safebook join; on the top-right the profile page in the podium section; on the middle-left picture sharing in the gallery page; on the middle-right wall posting in the square; on the bottom-left friendship advertisement; on the bottom-right friend browsing in the contacts page.

8.4 S2S: the P2P overlay of Safebook

The P2P substrate of Safebook, namely S2S, is a DHT similar to Kademlia[86] where nodes are arranged according to their identifier and store information on where to locate resources. Every resource can be identified by a hash value $DhtKey$ in the same 160-bit keyspace of the node identifiers, so that an XOR based distance function can be used to determine which set of nodes is responsible for which resource.

S2S differs from Kademlia since the lookups are performed in a recursive fashion to hide the identity of the real requester. Moreover, messages are signed with the sender's node private key $\mathcal{N}^{-}_{sender}$ and encrypted with the receiver's node public key $\mathcal{N}^{+}_{receiver}$[9].

Contact Management and Refresh Mechanism Similarly to Kademlia, the peer routing table of S2S is organized into several 'K-buckets' which are the leaves of a binary routing tree. Each K-bucket represents one subtree and has up to K representative contacts sharing the same distance range with respect to the peer itself. The routing table is managed dynamically: contacts are added as they are encountered and the buckets are splitted or merged as needed.

In S2S, a contact $\mathcal{N}$ is represented as:

$$< Cert\left(NId_{\mathcal{N}}, \mathcal{N}^{+}_{N}\right), @N, ConnSince, LastPing, Zombie, https >$$

The node certificate contains $\mathcal{N}$'s node Id and its node public key, @N represents $\mathcal{N}$'s IP address, while ConnSince and LastPing keep track of the first and last interaction between $\mathcal{N}$ and the current node. Finally, *Zombie* and *https* are independent boolean values: the first one is set to true in case $\mathcal{N}$ is not reachable, the second one when $\mathcal{N}$ can be reached through an HTTPS connection.

In S2S every bucket is periodically refreshed. When called, the refresh procedure pings the contact, preferably a zombie, with the lowest $LastPing - ConnSince$ value[10]. In case the peer answers, the Zombie flag is set to false, and the $LastPing$ value is updated. Every time a peer $\mathcal{A}$ is contacted by a peer $\mathcal{B}$, $\mathcal{A}$ inserts $\mathcal{B}$ in the appropriate bucket, if needed.

Store and Delete function In S2S, the function $store\left(DhtKey, Value\right)$ is called by an entrypoint $\mathcal{D}$ of a core $\mathcal{V}$'s Matryoska while registering the chain leading from $\mathcal{D}$ to a mirror

[9] More precisely, session keys are used to encrypt the payload. Such keys are advertised at the beginning of the message encrypted with the target node Id public key.

[10] Uptime is on average a good indicator of the remaining uptime as measured in [112].

$\mathcal{A}$ of $\mathcal{V}$. A *tolerance zone* is defined as the area of the key space around a target $DhtKey$ which is populated by all the nodes in charge of storing the value associated to that $DhtKey$.

A $DhtKey$ can be both the user Id of $\mathcal{V}$ or an hash of her identity properties, and is always certified by the TIS and associated to a user Id public key to prevent malicious users from claiming a bogus identity. The value associated to $DhtKey$ is an entry $EPTentry$ represented as:

$$< RegTok, Cert\left(NId_{\mathcal{D}}, \mathcal{N}_D^+\right), @D, Time >$$

where $RegTok$ is the registration token containing information on the target user $\mathcal{V}$, while the following information proves the entrypoint $\mathcal{D}$ is a valid node in the system. Finally, $@D$ represents $\mathcal{D}$'s IP address and $Time$ is used to synchronize $Value$ between neighboring docks.
More specifically, RegTok is represented as:

$$< Cert\left(DhtKey_{\mathcal{V}}, \mathcal{U}_V^+\right), Cert\left(UId_{\mathcal{V}}\mathcal{U}_V^+\right), ExpireTime >$$

Therefore $\mathcal{V}$ can never register an hash of an identity property she does not hold. After $ExpireTime$ is reached, the dock storing this $EPTentry$ removes it from its hash table.

In case a malicious user $\mathcal{M}$ aims at intruding to a target user $\mathcal{V}$'s Matryoshka by colluding with the dock $\mathcal{K}$ responsible for storing the entrypoint references of the victim's Matryoshka, an additional protocol explained in Appendix B.1 is needed[11].

When a chain gets broken or $\mathcal{D}$ leaves the network, to prevent a requester $\mathcal{U}$ from connecting to a node $\mathcal{D}$ that is not anymore an entrypoint for $\mathcal{V}$, S2S provides a function $remove\left(DhtKey, Value\right)$ giving the node $\mathcal{D}$ the opportunity to unregister itself as an entrypoint of $\mathcal{V}$.

Since the information on the role of entrypoint is not sensitive, the store and remove operations can be executed after an iterative search for $DhtKey$. Moreover, in S2S it is sufficient to reach a single dock $\mathcal{K}$ in the tolerance zone of $DhtKey$ to store or remove a value, since the command is by default forwarded to all the on-line neighbors of $\mathcal{K}$ within the tolerance zone of $DhtKey$.

Iterative and Recursive lookup

S2S provides two different ways to look up for a $DhtKey$: a classical iterative one, as in

[11] Such a protocol has not been implemented in the current prototype.

Kademlia, and a recursive one. While the first one is more robust against the node churn, the second one protects the node identifier of the real requester. Two parameters α and β can be set up to tune the lookup parallelism: in the iterative case, the requester triggers a lookup to the α nodes closest to the $DhtKey$, that will answer, in turn, with their β closest contacts; in the recursive case, these α nodes forward the request to their β closest contacts, each of them forwarding the request, in turn, to their own β closest contacts and so on. Both the iterative and recursive search succeed when a node sufficiently close to $DhtKey$ is reached and answers with the $Value$ associated to the lookup key.

Join and Leave To join the S2S network, a node needs to obtain a certified node identifier from the TIS. Afterward, to fill its buckets, the newcomer triggers an iterative lookup for its node identifier. Even if no $Value$ is associated to any node Id, still the iterative lookup can take place and allows the newcomer to get in touch with a set of new peers that may populate its routing table. However, the newcomer refers to a set of predefined bootstrapping peers at the very first join, when its buckets are empty. When logging out, all the buckets are saved.

8.5 Additional challanges

Future develompent of Safebook can address both usability and ubiquity. Many users, even those concerned about privacy in OSNs, may see the software installation or their home NAT configuration as a barrier. It would be interesting to evaluate a possible development of Safebook as a web application (webapp) too. One user could connect to the Safebook website and execute the Safebook client coded in a browser-supported language. Users' data may be downloaded from the online friends. Such users could also download their friends' data items and serve as a mirror for them.
Open listening sockets would be a main challenge to tackle. In case users executing Safebook as a webapp could not play the role of entrypoints, an evaluation on the impact on data availability should be conducted too. Due to the ubiquity of web browsers, Safebook could also be executed by several user devices without requiring the user to install each time the Safebook client.

8.6 Summary

This Chapter presented the first prototype of Safebook, a multi-thread event driven application written in python that can be executed on multiple operating systems such as Windows, Linux and MacOs.

Once started, the client runs locally and opens four main services: a local web server to implement the user interface, a UDP server to receive P2P traffic, a TCP server to receive Matryoshka traffic, and a HTTPS server to receive encapsulated Matryoshka and P2P traffic from those peers that are behind strict policy corporate firewalls.

The web based user interface helps user to benefit from Safebook as easily as in current OSNs. A simple interface helps a Safebook user to discriminate which information was generated by her or specifically needs her attention, that is collected in the *Podium*, from all the information generated by the user's contacts, that is collected in the *Square*. A last section, the *Contacts* page, allows the user to browse her friend list.

User generated content is shared with limitations, and is private by default. Access control is achieved with the use of badges, representing user-defined overlapping groups of contacts, consisting on symmetric Data Encryption Keys distributed among trusted contacts at the act of friendship establishment or later.

Friendship lookup requires the untrusted environment of the P2P overlay to achieve

the same security and privacy properties of the Matryoshka overlay. To this goal, a DHT inspired by Kademlia has been implemented to achieve unforgeability of node identifiers, recursiveness of lookups, and message integrity and confidentiality. Moreover, to increase the responsiveness of the system, entrypoint references can be also removed from the DHT so that requesting users don't retrieve endpoints of broken Matryoshka chains.

Chapter 9

Conclusion and future work

Social Network Services (SNS) maintain Online Social Networks (OSN) as digital representations of users and their relationships, and allow even users with limited technical skills to share a wide range of personal information with a theoretically unlimited number of partners. This advantage comes at the cost of increased security and privacy exposures for users for two main reasons: first of all, users tend to disclose private personal information with little guard, and secondly, existing SNSs severely suffer from vulnerabilities in their privacy protection or the lack thereof. Even assuming a perfect protection from malicious users, legitimate users are still exposed to a major orthogonal privacy threat, since in all existing SNSs the service provider has access to all the data, including some private information, stored and managed by the SNS itself, and can misuse such information easily. Since the access to users' private data is the underpinning of a promising business model, current SNSs are not likely to address this problem in the near future.

This thesis tackled the security and privacy issues in Online Social Networks with a special emphasis on the privacy of users against the omniscient Social Network Service provider.

In the first part of this thesis, we discussed the security and privacy issues in OSN.

In chapter 2, we introduced Online Social Networks and illustrated their main functionalities. We identified and classified the large amount of core information stored in OSNs into several main areas. Misuse of such information has been examined in chapter 3 that defined privacy, integrity and availability objectives for OSNs, and discussed a detailed spectrum of attacks that can be perpetrated in OSNs against such objectives. Most of the countermeasures we proposed to prevent such attacks revealed to be ineffective against the Social Network Service provider itself which plays the role of an omniscient centralized entity. An overview of the main centralized OSNs underlined the market capitalization of their providers.

Researchers recently proposed to design OSN applications based on a distributed architecture in order to avoid centralized control over users' data. In chapter 4 we classified existing solutions, known as Distributed Online Social Networks (DOSN), and analyzed them together with their limitations. We showed that client-server (or cloud) approaches do not always evade the potential control of a single party, as e.g. a company or an organization, on the hosted user's data. On the other hand, although current peer-to-peer approaches do not suffer from such control, they expose users to potential communication tracing attacks. In the context of OSN, such communication traces disclose details on the structure of the social network graph. We therefore concluded that none of current approaches is suitable to achieve the goal of preserving user's privacy in OSNs.

In the second part of this thesis, we proposed Safebook as a solution to security and privacy threats in OSN.

In chapter 5 we presented the main design principles of Safebook: a P2P architecture to avoid the need for a central provider, and the real life trust among peers resulting from the OSN application itself to enforce their cooperation. In Safebook, friend nodes provide the basic services of data storage, retrieval and communication, and consequently build the OSN. A first ring of friends serves all requests for one user's data even when such user is offline, and further rings of friend-of-friends build such user's Matryoshka and prevent the disclosure of the user's friendship relationship from tracing attacks. Privacy is achieved with communication obfuscation through anonymous routing techniques, data confidentiality through the use of encryption, and profile integrity through the adoption of certified identifiers.

In chapter 6 we analyzed and evaluated the feasibility of Safebook. Based on the Matryoshka setup parameters and the online probability of nodes we computed the probability of building Matryoshkas and their lifetime. We observed that choosing a large value for the depth h of the Matryoshka (which is a privacy requirement) can have a severe impact on the data availability and retrieval. Thus, h should be set as small as possible while still allow for a good privacy degree. We further discussed the data availability of Safebook by evaluating the amount of data a user can retrieve at each request and showed an example based on a realistic scenario.

In chapter 7 we analyzed privacy in centralized or distributed OSNs from the graph theory perspective with the help of real social network dumps. We observed that metrics such as the degree and the clustering coefficient of nodes severely affect users' privacy as defined in chapter 3 with respect to identity/friendship privacy and usage control, while the mixing time of random walks in the social network graph plays an essential role in preserving the users' communication untraceability. Further specific analysis on Safebook confirmed that the presence of strong local clustering negatively impacts the privacy of the solution. Experiments also confirmed a strong trade-off between privacy and performance such that delay and reachability are inversely proportional to privacy.

In Chapter 8, we presented the first prototype of Safebook, a multi-thread event driven application written in python that can be executed on multiple operating systems such as Windows, Linux and MacOs. The web based user interface helps users to benefit from Safebook as easily as in current OSNs. User generated content is private by default. Access control is achieved with the use of badges, representing user-defined overlapping groups of contacts, consisting on symmetric Data Encryption Keys distributed among trusted contacts at the act of friendship establishment or later.

In conclusion, this study showed that privacy concerns in current OSNs are mainly due to the centralized storage of all users' data at the SNS provider's databases. Among all the current solutions that allow for the distributed storage of the user generated contents, Safebook has been designed with the main goal of preserving user's privacy from the very beginning, and achieves such goal. Safebook does not only protect the digital representation of users, but also their relationships, from any malicious party. The evaluation of Safebook shows that a realistic compromise between privacy and performance is feasible. Furthermore, the underpinnings of Safebook can serve as a model to tackle various well known problems

in the area of secure communications. Thus, a decentralized approach relying on social links can shed new light on hard problems of the past such as anonymous communications, secure routing, or cooperation enforcement in self-organizing systems.

9.1 Directions for future research

We envision three main directions for future research towards privacy, performance, and business model of Safebook.

The first research direction is toward privacy in terms of usage control over the shared profile data. The problem of usage control, which refers to the control of the data after its publication, is becoming a very challenging problem due to the rapid growth of the number of users involved in content sharing. Since Safebook relies on the collaboration of users for any operation including data management and security, the collaboration of a sufficient number of legitimate peers may be leveraged to enforce control on the data forwarded along the trusted Matyroshka chains. For instance, the message could have to follow a dedicated path of sufficient intermediate nodes which perform the dedicated tasks defined in the usage control policy before reaching its final destination. Thanks to this multihop enforcement mechanism, users would be able to control the usage of their shared data since the very beginning stage of its publication.

We propose an initial solution using Safebook for the particular picture sharing application (see appendix C). Nevertheless, the protection of the picture and the enforcement of this control is only efficient in the confined environment of Safebook and when pictures are not encrypted.

The second research direction is towards performance in terms of data availability. Due to the particular approach in Safebook, data served by friend nodes with limited bandwidth and storage capacities may not be always 100% available. To increase such availability, Safebook users may store their data at some non friend nodes, or in an untrusted storage service outside Safebook. In the first case, new cooperation enforcement mechanisms should be designed to prevent selfishness. Such mechanisms could take advantage of the distributed nature of P2P networks. We have started to design an initial incentive mechanism where transactions should be approved by a predefined certain number of peers (see appendix D).

In the second case, new features should prevent the external storage service provider from deriving sensitive information such as friendship relationship between Safebook users.

One last research direction is towards the definition of a business model that can attract users to Safebook. Privacy alone may not be sufficient to serve this goal, at least for non professional users. On the contrary, economic incentives may facilitate users' migration on Safebook. Even if one user's profile data is valueless, it may be valuable if it is aggregated with other users'profile data. In this case, Safebook users profiles may be aggregated and sold to provide anonymous statistics to companies, that would in turn pay users for this data, improve their products, sell them and return on their investment.

Appendices

Appendix A

Résumé étendu

Les ***Services des réseaux sociaux*** (SNS) tels que *Facebook*, *LinkedIn* ou *Google +*, sont desormais devenus un facteur prédominant de l'Internet. En s'addressant à une population d'utilisateurs très grande avec une grande différence de provenance social, éducative et national, ils permettent même aux utilisateurs ayant une connaissance des moyens techniques limitée de publier des renseignements personnels et de communiquer facilement.

En général, les ***réseaux sociaux en ligne*** (OSN) résultant de ces SNSs sont des représentations numériques d'un sous-ensemble des relations que leurs participants, des utilisateurs inscrits ou des institutions, entretiennent dans le monde physique. En comprenant les participantes avec leurs relations, ils modélisent le réseau social sous forme de graphe. Cependant, la popularité et l'acceptation générale des services de réseau social comme plateformes de messagerie et de socialisation attire pas seulement les utilisateurs fidèles qui tentent d'ajouter de la valeur à la communauté, mais aussi les parties ayant des intérêts plutôt défavorables, soit commerciales, soit malveillants.

La motivation principale pour les membres d'adhérer à une OSN, de créer un profil, et d'utiliser les différentes applications offertes par le service, est la possibilité de partager facilement des informations avec des contacts sélectionnés ou avec le public soit pour fins *professionnels*, soit *personnelles*. Dans le premier cas, l'OSN est utilisé avec les objectifs de gestion de carrière ou d'une entreprise, d'où des SNS avec une image plus sérieuse,

comme *XING* ou *LinkedIn*, sont choisis. En tant que les membres dans ce cas sont conscients de l'impact professionnel de l'OSN, ils paient généralement beaucoup d'attention sur le contenu des données qu'ils publient que regarde eux-mêmes et d'autres. Dans le cas d'une utilisation plus privé, le utilisateurs partagent des informations plus personnelles telles que les données de contact, photos personnelles, ou des vidéos. Des autres membres peuvent être marquées ("*tagged*") dans les photos partagées, et des liens vers leurs profils respectifs sont créés automatiquement.

L'activité principale des membres des OSNs est la création et l'entretien de leurs listes de contacts, qui permit de construire le graphe numérique de l'OSN. En informant automatiquement les membres sur les modifications des profils des leurs contacts, le SNS permet ainsi aux utilisateurs de rester à jour avec les nouvelles de leur amis et très souvent la popularité des utilisateurs est mesuré avec le nombre de contacts.

L'analyse de l'OSN à l'égard des propriétés de sécurité et de la vie privée de leurs utilisateurs revèle une serie évidente des menaces. En règle générale, une multitude de données personnelles des participants est stockés par les fournisseurs de SNSs, en particulier dans le cas d'OSN avec fins non professionnelles. Ces données sont soit visible pour le public, ou, si l'utilisateur est conscient des problèmes de la vie privée et capables de régler les paramètres de le SNS, à un groupe sélectionné d'autres membres. Comme les profils sont attribués à des personnes vraisemblablement connues du monde réel, ils sont implicitement évaluées avec la même confiance du propriétaire présumé du profil. En outre, toutes les actions et les interactions couplées à un profil sont de nouveau attribué au propriétaire présumé de ce profil.

Différentes études ont montré que les participants représentent clairement le maillon faible de la sécurité dans les OSNs et qu'ils sont vulnérables à plusieurs types d'attaques d'ingénierie sociale. Ceci est en partie causée par un manque de sensibilisation aux conséquences des actions simples et sans doute privé, comme accepter les demandes d'amitié, ou marquer les images, ainsi que les opérations de communication comme commenter les profils ou afficher des messages sur les murs. Le faible degré de facilité d'utilisation de contrôles de confidentialité offertes par le SNS, et enfin et surtout la confiance dans les autres profils aggravent certainement le problème.

En analysant les problèmes de la vie privée dans les OSNs actuels, il devient évident que, même si tous les participants fussent au courant des expositions et ils fussent compétents

dans l'utilisation des SNS, et même si un ensemble concis des mesures de protection de la vie privée fût déployé, l'OSN serait toujours exposé à violations potentiels de la vie privée par le fournisseur omniscient du service : les données, directement ou indirectement fournis par tous les participants, sont en fait collectées et stockées de façon permanente dans des bases de données du fornisseur de service, qui devient potentiellement un ***Grand Frère*** capable d'exploiter ces données à de nombreux égards qui peuvent violer la vie privée des utilisateurs individuels ou des groupes d'utilisateurs.

L'importance de cette exposition de la vie privée est soulignée par la capitalisation boursière de ces fournisseurs, qui atteint 50 milliards de dollars (Facebook Inc, selon l'investissement de Goldman Sachs et Digital Sky Technologies en 2011) [12], et par les revenus globales des annonces publicitaires, qui ont atteint 5 milliards de dollars en 2011 et sont estimés à doubler avant le 2013 (selon eMarketer [25]).

Cette thèse affirme que la vie privée de l'utilisateur peut être facilement mis en péril en raison de l'architecture centralisée de l'OSN, et les fournisseurs de SNSs actuels ne sont pas susceptibles de remédier à ce problème en raison de leur modèle d'affaires. Ce travail considère plutôt la protection des données privées dans les OSNs comme un sujet urgent et propose une nouvelle architecture pour OSN conçue avec le but de proteger la vie privée de ses utilisateurs.

A.1 Objectifs de recherche

Cette thèse suppose que ***la protection de la vie privée de l'utilisateur contre le fournisseur omniscient des SNSs*** soit l'objectif principal pour l'OSN et vise à identifier les caractéristiques principales qu'une OSN devrait satisfaire pour atteindre un tel objectif ainsi que à fournir une nouvelle architecture pour préserver la vie privée dans l'OSN. Comme objectif supplémentaire, cette these vise à protéger aussi la ***vie privée des utilisateurs honnêtes contre les utilisateurs mailvellants***.

Nous définissons l'objectif de protéger la vie privée comme la possibilité de dissimuler des informations au sujet de n'importe quel utilisateur à tout moment, même dans la mesure de cacher la participation des utilisateurs et des activités au sein de l'OSN. Par conséquent, la vie privée englobe pas seulement la protection des renseignements personnels que les utilisateurs publient sur leurs profils, mais prend également en compte la communication entre les utilisateurs, c'est-à-dire, il faut qu'aucune des parties sauf que celles adressées directement

ou explicitement approuvé doit avoir la possibilité de suivre la communication. Les détails concernant les messages doivent être cachés, afin que que les parties qui communiquent connaissent l'identité l'un de l'autre et le contenu de la communication. L'accès au contenu du profil de l'utilisateur doit être accordée directement par l'utilisateur, et ce contrôle d'accès doit être aussi fine que le profil lui-même.

Ainsi que l'objectif de protéger la vie privée, cette thèse vise aussì à des objectifs de sécurité supplémentaires de ***intégrité*** et ***disponibilité***, qui se déclinent, dans les OSNs, dans une façon légèrement différentes par rapport aux systèmes traditionnels. Dans le cadre des OSNs, l'intégrité doit être prolongée au-delà de l'objectif fondamental de protéger les données des utilisateurs et leurs identités contre toutes modifications non autorisées, pour couvrir une variété d'attaques telles que la création de profils, fictifs ou clonés, ou d'autres types d'usurpation d'identité. Chaque profil doit alors être associé sans ambiguïté à un individu dans le monde réel.
La proprieté de disponibilité devrait empêcher les attaques de déni de service que visent à interrompre la possibilité de communiquer avec la victime. En outre, la disponibilité ne devrait pas seulement atteindre l'objectif de base d'assurer le SNS, même face à des attaques et des failles, mais elle devrait aussi garantir la robustesse contre la censure.

A.2 Contributions principales

Le caractère centralisé des OSNs permet aux fournisseurs de SNS de surveiller et intercepter des données sensibles des utilisateurs. Ce problème a récemment attiré un certain intérêt dans la communauté des chercheurs et les résultats des recherches peuvent être résumées dans une famille de solutions connues sous le nom de ***réseaux sociaux en ligne décentralisés*** (DOSN). Ces DOSNs visent à diffuser les données de l'utilisateur avec l'adoption d'un approche client-serveur (ou nuage) où les utilisateurs ne participent pas au service de stockage et les données stockées sont toujours disponibles, ou via un approche peer-to-peer (P2P), où les utilisateurs participent au service de stockage et les données stockées ne peuvent pas toujours être disponibles.

Même si dans tous les DOSNs actuelles les données partagées de l'utilisateur sont protégés par le chiffrement, telles solutions ne sont pas aptes à atteindre nos objectifs de recherche. Les approches client-serveur (ou un nuage) ne peuvent pas toujours se soustraire au contrôle

potentiel d'un parti individuelle, comme par exemple une entreprise ou une organisation, sur les données des utilisateurs. Un tel contrôle pourrait être eludé si les utilisateurs missent en place et maintinissent leurs propres serveurs pour héberger leurs données et ceux-là des autres utilisateurs, ce qui conduit à une approche P2P.
Toutefois, les DOSNs P2P actuels souffrent de l'exposition au traçage de la communication par les pairs malveillants. Dans le cadre des OSNs, ces traces de communication sont susceptibles de correspondre à des relations d'amitié dans le réseau social, et donc ils peuvent même divulguer les détails sur la structure du graphe du réseau social.

Parmi les approches actuelles de DOSNs basés sur P2P, aucun d'entre eux répond à ce problème. En outre, les DOSNs P2P actuels se basent souvent sur des architectures P2P existantes et souffrent des problèmes bien connus de l'absence de coopération en raison de l'égoïsme de noeuds et ainsi des attaques de déni de service en raison de la création de multiples identités des pairs sous contrôle d'une partie malveillante. Pour ces raisons, les DOSNs P2P actuels ne semblent pas approprié pour le but de préserver la vie privée des utilisateurs.

La première contribution de cette thèse consiste en une analyse des réseaux sociaux en ligne qui comprend les acteurs principaux des OSN (resumés dans la Figure A.1), les fonctionnalités OSN (resumées dans la Figure A.2), la nature des données sensibles partagées par les utilisateurs (resumés dans la Figure A.3), et les principales menaces résultant de l'utilisation abusive potentielle de ces données (resumés dans la Figure A.4 et dans la Figure A.5).

La deuxième contribution de cette thèse consiste à remplir le manque des OSNs securisés en proposant une nouvelle architecture décentralisée pour OSNs ayant comme objectif principal la protection de la vie privée des utilisateurs. La décentralisation est obtenue par un nouveau système P2P qui *se base sur la confiance entre les utilisateurs de l'OSN dans la vie réelle*. Cette confiance est obtenue comme résultat de l'application OSN et est utilisée comme un mécanisme naturel de renforcement de coopération *pour construire l'application de réseau social lui-même* (resumé dans la Figure A.6). Dans la solution proposée, appelée ***Safebook***, les pairs sont disposés selon la confiance qui leurs utilisateurs entretiennent dans la vie reelle, c'est-à-dire, selon le graphe du réseau social.

Les noeuds gérés par les amis d'un utilisateur stockent les données de cet utilisateur et le servent même lorsque l'utilisateur est hors ligne. Comme avec le routage anonyme, les demandes de données et les réponses sont récursivement déléguées à différents pairs

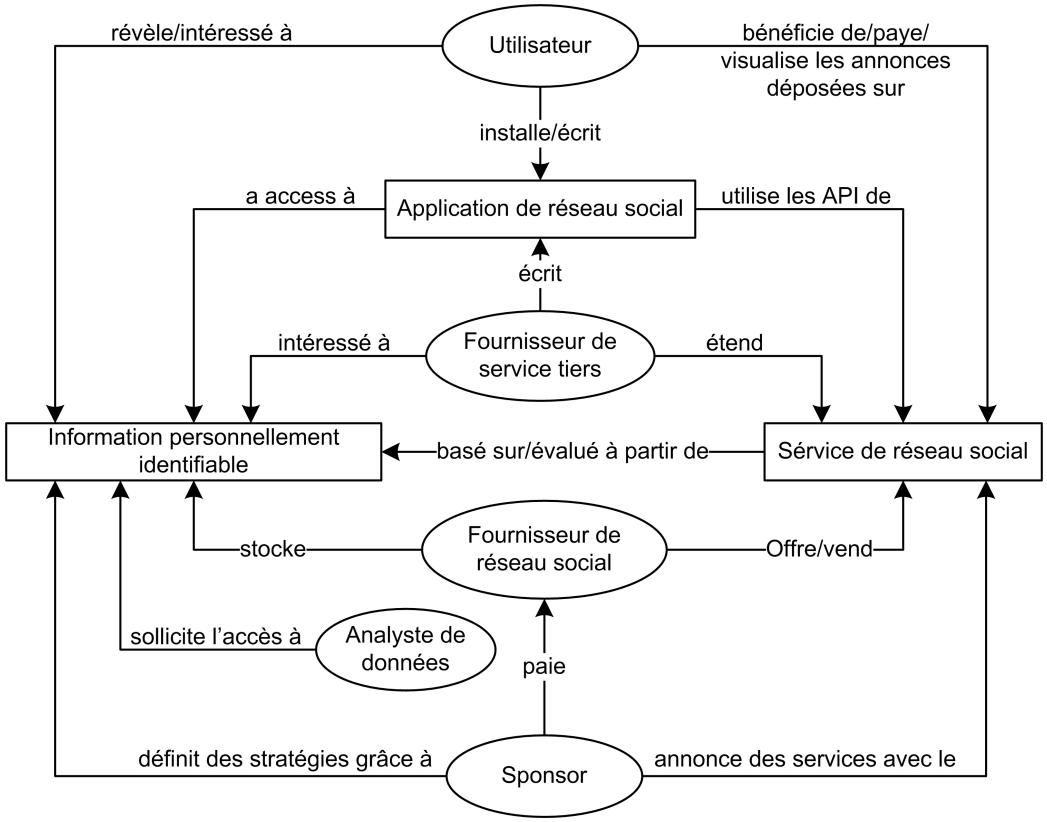

Figure A.1 – Clients des services de réseaux sociaux et leur relations avec les informations personnellement identifiables.

afin de cacher l'identifiant du demandeur réel et d'empêcher la divulgation des relations de confiance entre les membres de l'OSN. La confidentialité des données est assurée par l'adoption de techniques de cryptage et l'intégrité des profils est assurée par un service (ou plusieurs) d'identification de confiance hors ligne dont la compétence est limitée à des fins d'identification seulement.

L'architecture Safebook a été conçu avec l'objectif principal de préserver la vie privée des utilisateurs : l'intégrité des profils, obtenue grâce à l'adoption d'identificateurs certifiés qui sont signés par un service d'identification de confiance, avec la confidentialité et l'intégrité des données, obtenues grâce à l'adoption de techniques de cryptage classiques, protégent les sommets du graph de la reseau social représenté par l'OSN, c'est-à-dire, òrd profiles

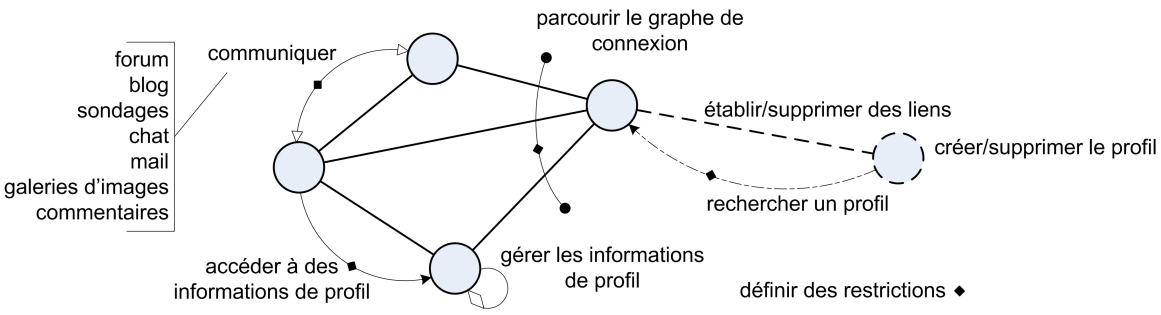

Figure A.2 – Fonctionnalité principale d'un typique réseau social en ligne.

des utilisateurs ; le routage multi-hop des messages et des techniques de chiffrement supplémentaires offrent depistage et confidentialité à la communication, et protègent les arcs du graphe du réseau social représenté dans l'OSN, c'est à dire la liste de contacts de l'utilisateur.

La troisième contribution de cette thèse consiste en l'évaluation de la faisabilité et des performances de Safebook. A partir de la probabilité en ligne de ses pairs (resumé dans la Figure A.9), du nombre d'amis de l'utilisateur, et de la longueur des chemins fournissant le depistage de la communication, des modèles analytiques estiment la probabilité de récupération des données d'un utilisateur cible et la taille maximale de chaque message contenant ces données (resumé dans la Figure A.11).

La quatrième contribution de cette thèse consiste dans l'enquête sur la forte relation entre les propriétés topologiques du graphe de réseau social et la protection maximale de la vie privée des utilisateurs des réseaux sociaux en ligne. Nous observons trois métriques, le clustering coefficient (resumé dans la Figure A.13), la distribution des degrés (resumé dans la Figure A.13) et le temps de mélange (resumé dans la Figure A.14), et montrons qu'elles donnent des aperçus fondamentaux sur le degré de protection de la vie privée soit dans les reseaux sociaux centralisés soit distribués.

Une enquête plus approfondie est menée sur l'impact de la topologie du graphe de réseau social à la fois sur la performance et la confidentialité des Safebook. En Safebook il ya un fort compromis entre la performance et la sauvegarde de la vie privée, car le retard et l'accessibilité sont inversement proportionnels à la protection efficace de la vie privée. En fait, avec des chemins très courts, la confiance n'est pas élevé : la probabilité de l'obtention des relations d'amitié entre les utilisateurs Safebook est plus grande. Néanmoins, avec des

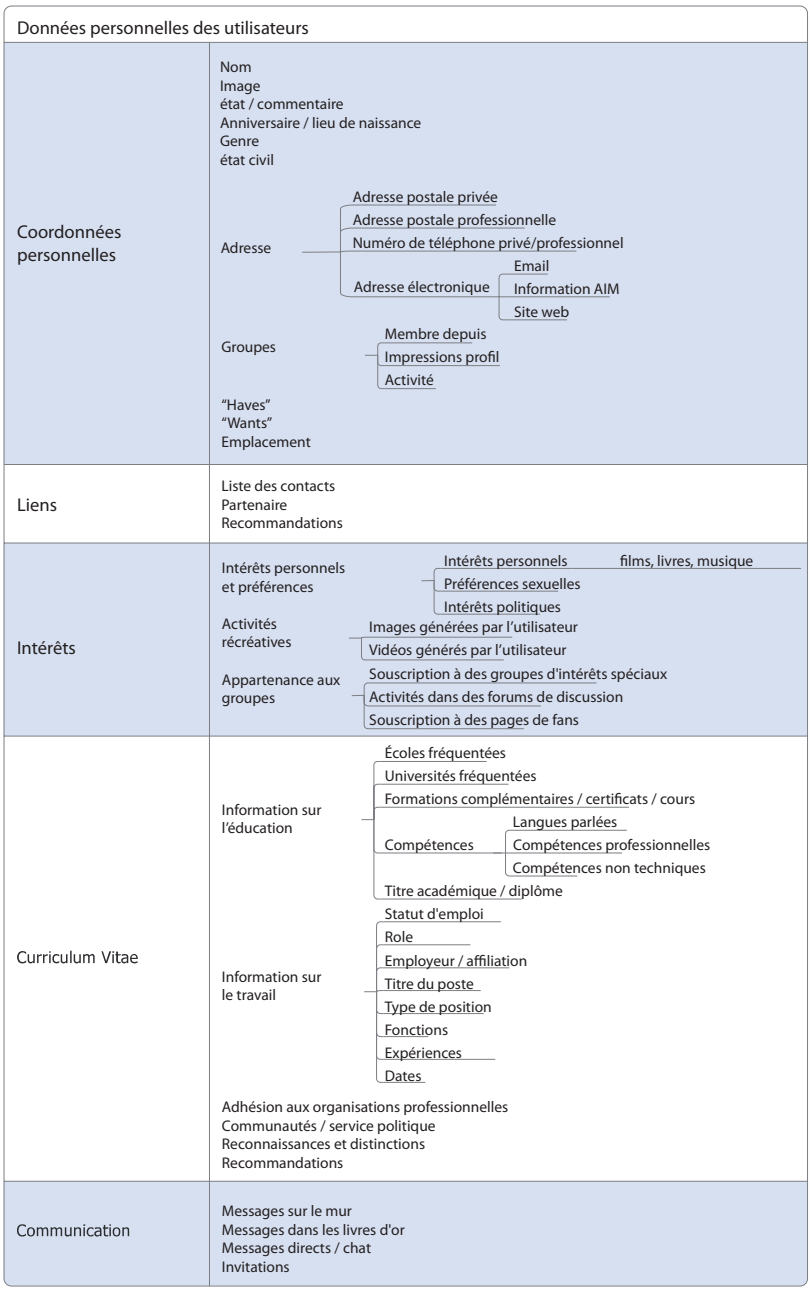

Figure A.3 – Types de données généralement enregistrées dans les profils des réseaux sociaux en ligne.

chemins très longues, la probabilité de récupération des données decroit, et le délai de récupération augmente.

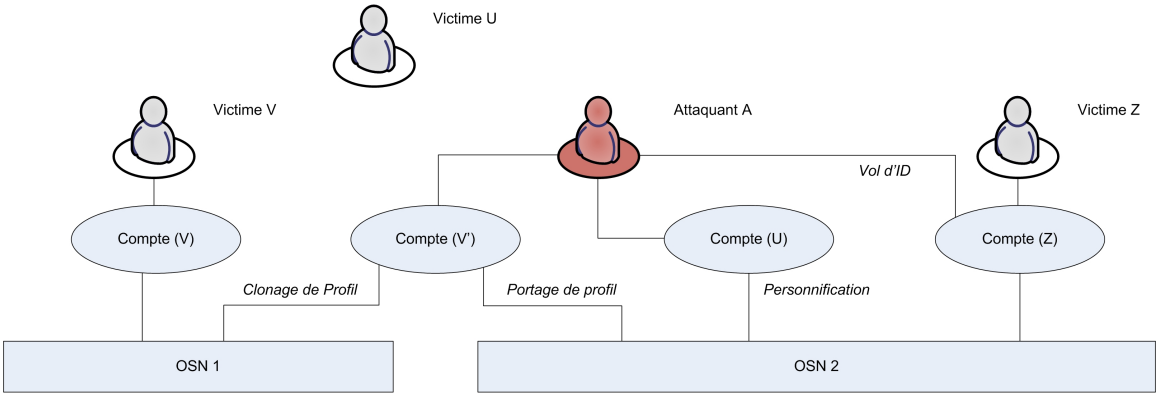

Figure A.4 – Les attaques d'usurpation d'identité : la victime $\mathcal{U}$ ne posséde aucun compte de reseau social, la victime $\mathcal{V}$ a un compte sur OSN1 et la victime $\mathcal{Z}$ sur OSN2. L'agresseur $\mathcal{A}$ génère un compte $\mathcal{V}$ sur l'OSN2, une copie du compte de $\mathcal{V}$ sur OSN1 et OSN2, et il s'enregistre sur OSN2 avec les informations d'identification de $\mathcal{Z}$.

Nous observons que le choix optimal pour cette longueur dépend du graphe social lui-même (resumé dans la Figure A.15).

La cinquième et dernier contribution de cette thèse consiste à la mise en oeuvre et le déploiement de Safebook (resumé dans la Figure A.16 et dans la Figure A.17). Le prototype Safebook est écrit en python et peut être exécuté sur plusieurs systèmes d'exploitation tels que Windows, Linux et MacOs. Une interface utilisateur basée sur le Web (resumé dans la Figure A.18) permet à l'utilisateur de bénéficier des outils de confidentialité disponibles telles que celles qui lui permet de partager des données avec des limitations.

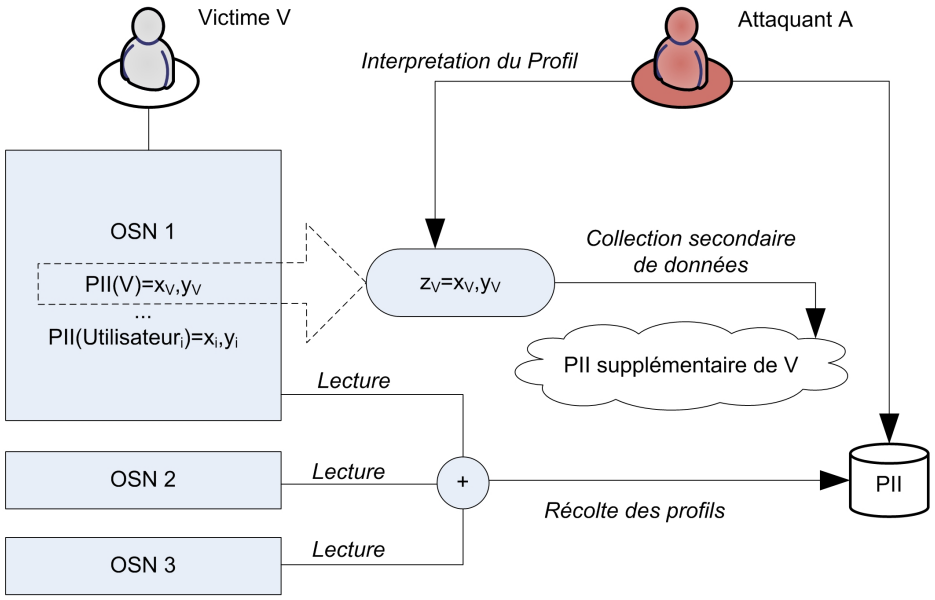

Figure A.5 – Principales menaces liées à l'access aux informations personnellement identifiables dans les OSNs actuelles.

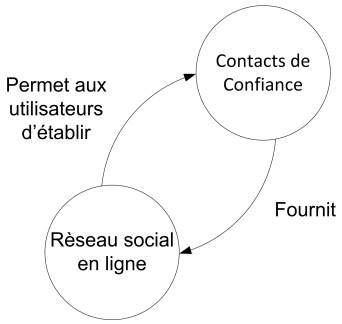

Figure A.6 – la relation cyclique montrant comment la confiance entre les utilisateurs dans la vie réelle peuve construire l'OSN elle-même.

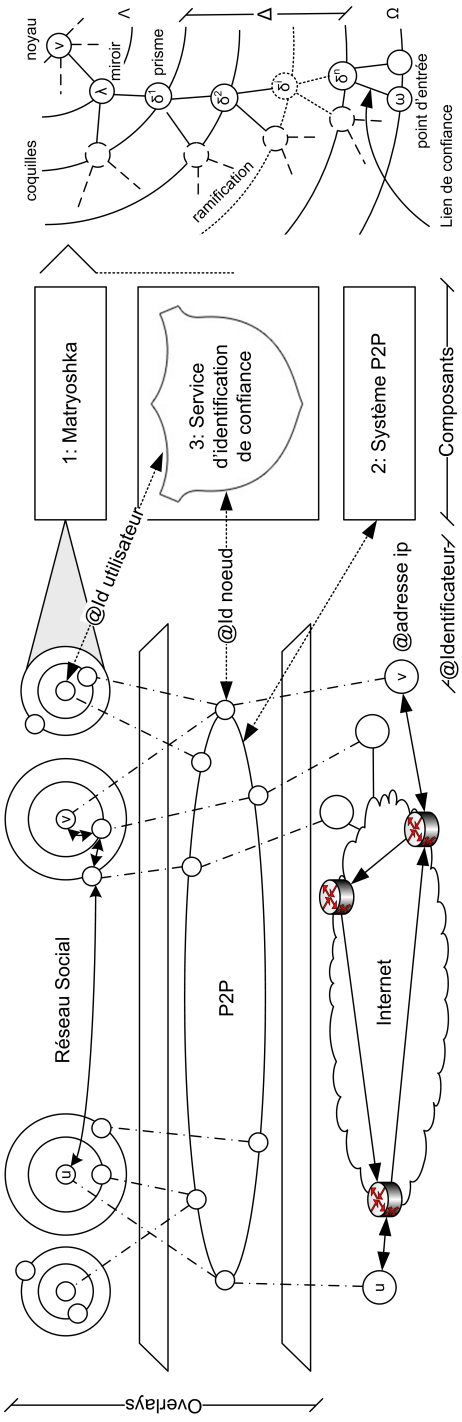

Figure A.7 – Les recouvrements de Safebook (à gauche), les composants principaux (au centre) et la Matryoshka (à droite).

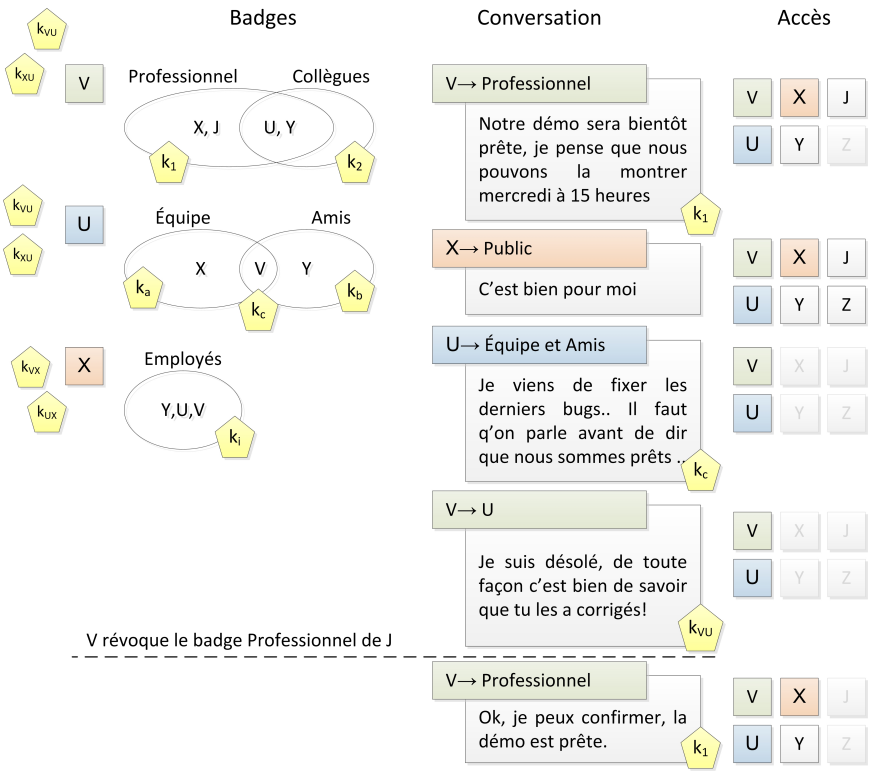

Figure A.8 – Un exemple de communication entre les utilisateurs avec des différents politiques d'access aux donnés partagés.

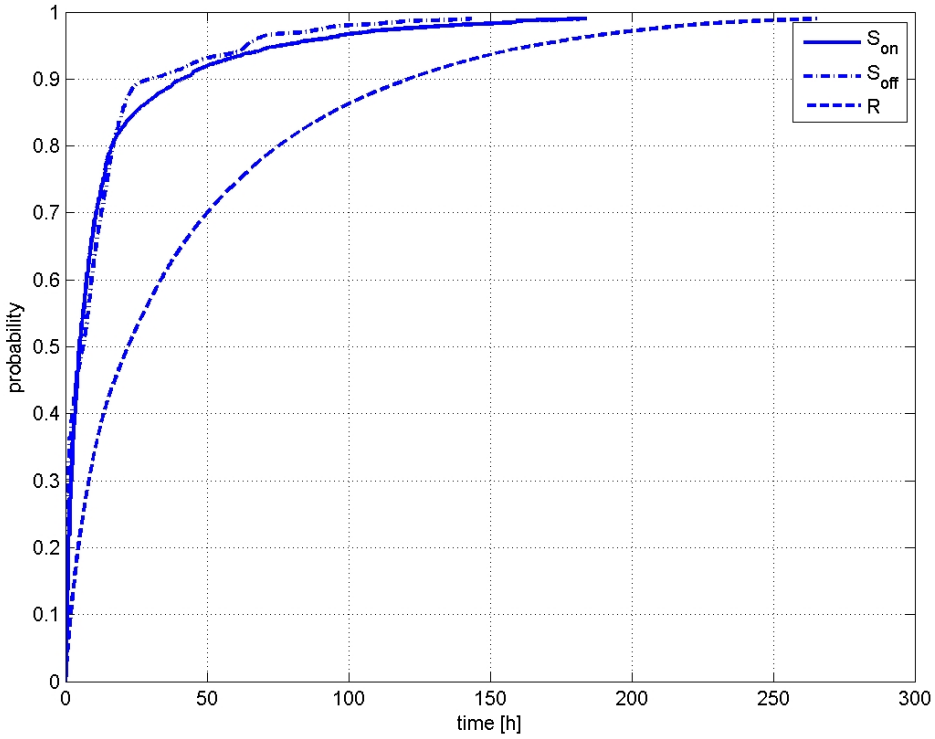

Figure A.9 – Les distributions des temps en ligne, hors ligne et correspondantes à la vie résiduelle provenant de l'ensemble de données Skype.

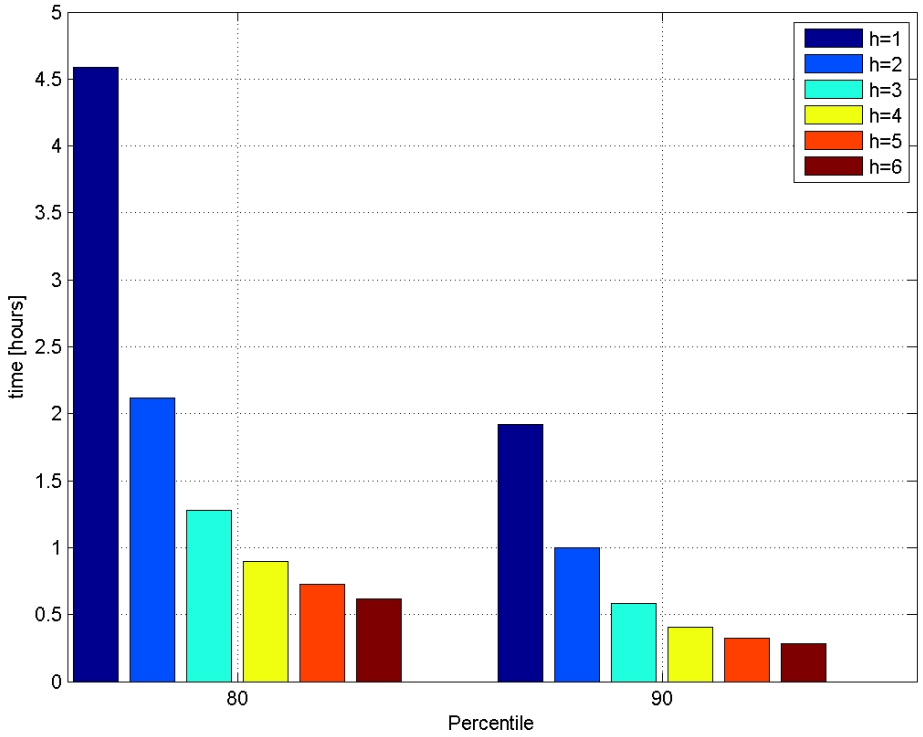

Figure A.10 – Durée de vie résiduelle de la chaîne par rapport à sa longueur h.

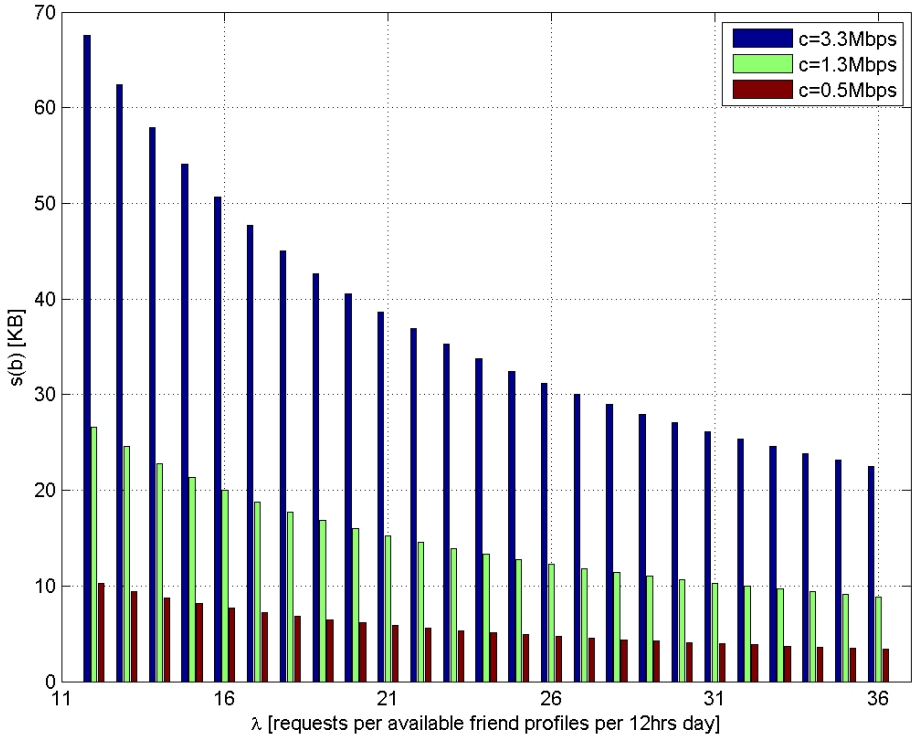

Figure A.11 – Taille maximale des données récupérées à chaque demande avec bande passante c en upload et différents taux de demandes λ.

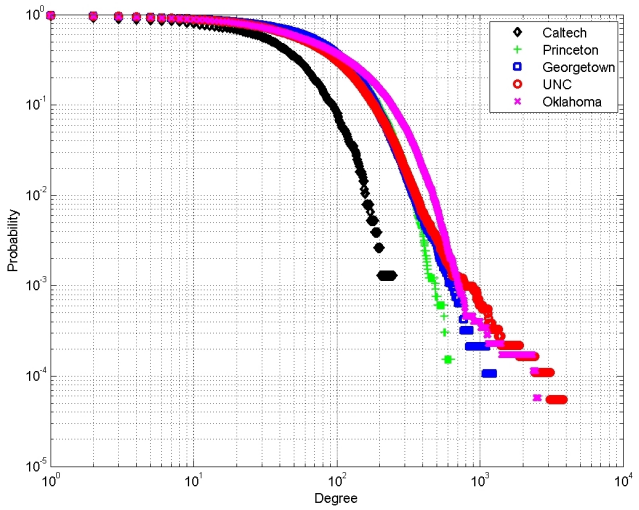

Figure A.12 – Log-log plot de la distribution cumulative complementaire du degré dans des reseaux sociaux réelles.

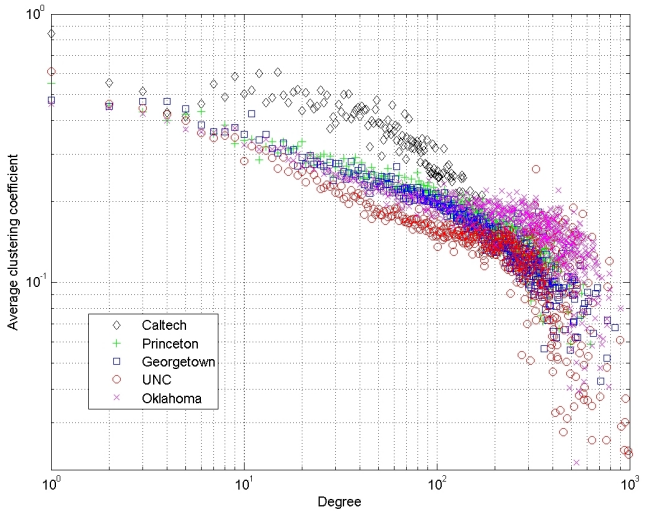

Figure A.13 – Coefficient de clustering moyen dans des reseaux sociaux reelles par rapport au degré des connections.

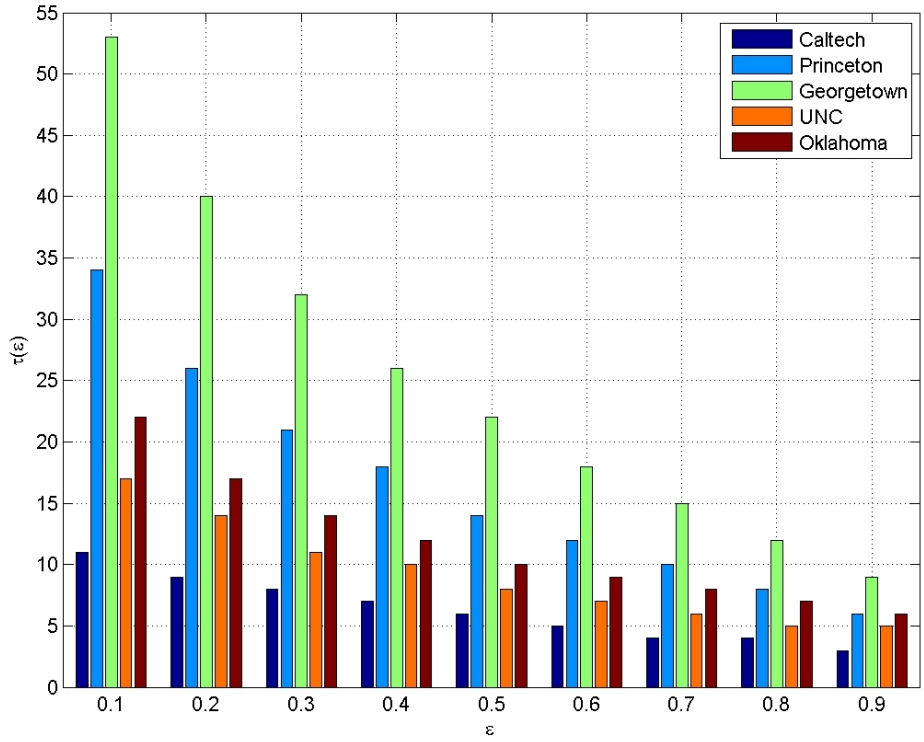

Figure A.14 – Temps de mélange (en pas) dans des réseaux sociaux réels.

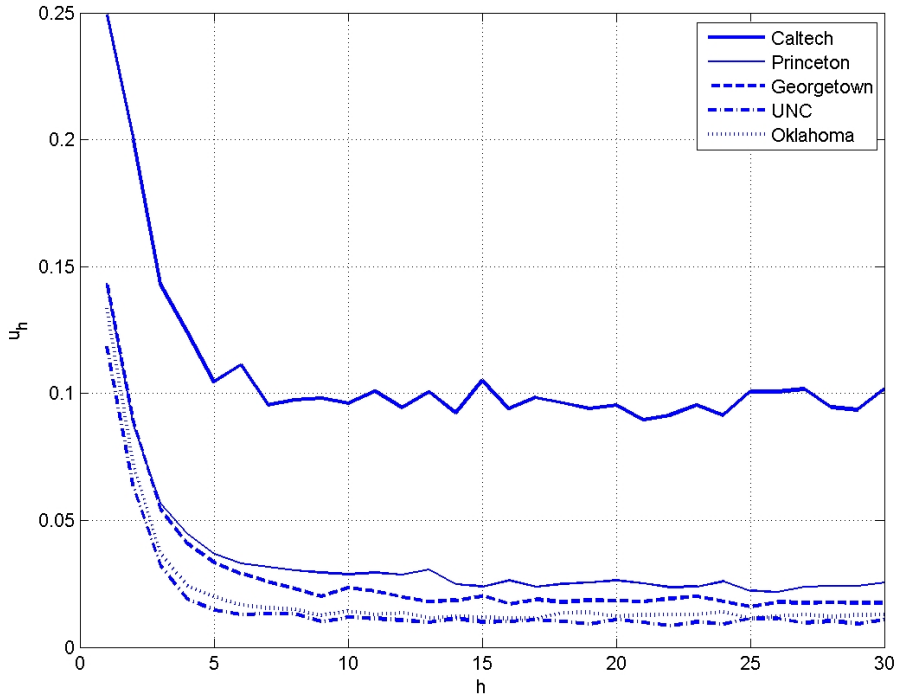

Figure A.15 – Ratio des amis communs entre les deux noeuds $\mathcal{V}$ et θ^h à une distance sociale h dans le reseau social.

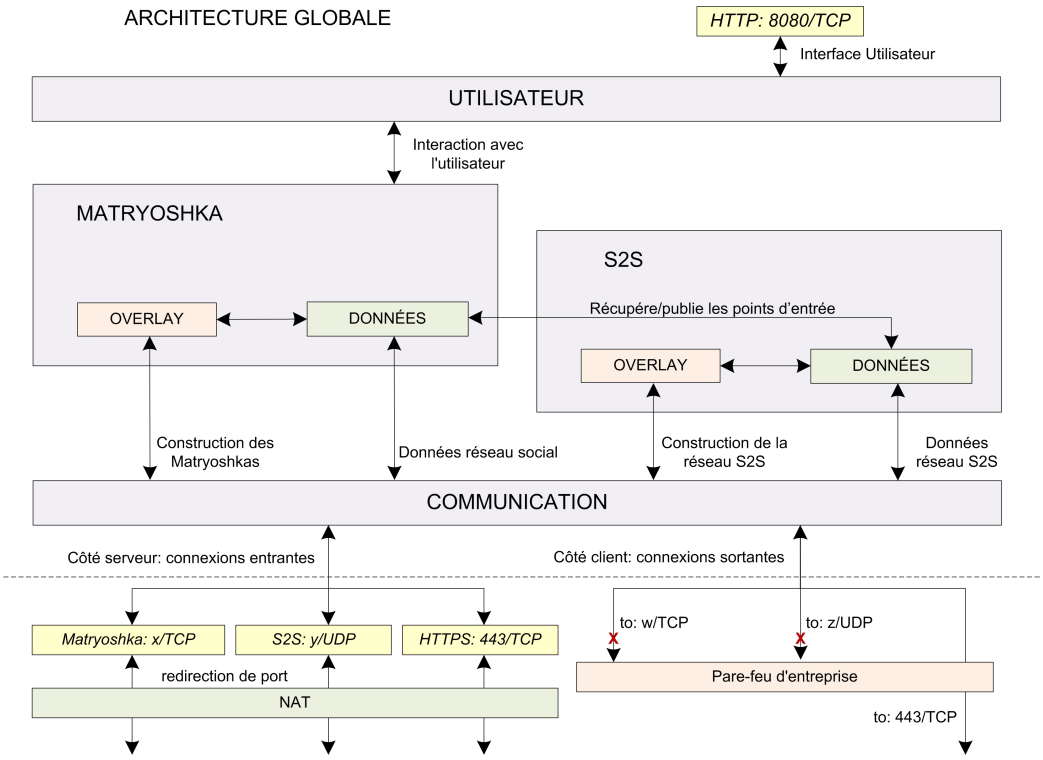

Figure A.16 – Architecture globale de Safebook.

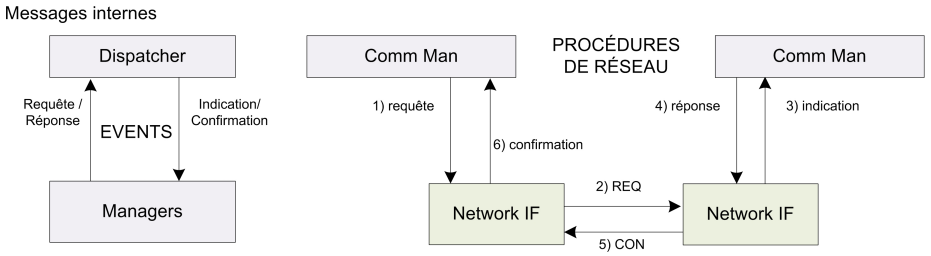

Figure A.17 – L'échange interne (à gauche) et externe (à droite) de messages en Safebook.

Figure A.18 – l'interface graphique de Safebook : sur le coin supérieur gauche comment rejoindre Safebook ; en haut à droite de la page le profil dans la section podium ; au milieu à gauche le partage de photos dans le page de la galerie ; au milieu à droite l'affichage sur le mur dans la page square ; en bas à gauche l'annonce de l'amitié et en bas à droite la navigation dans la page des contacts.

Appendix B

Further Matryoshka security features

B.1 Matryoshka Verification Protocol

The Matryoshkas pose as infrastructure vital for the availability of the data published by their core, and for their core's reachability. Considering their distributed organization, failures may always remain undetected over a period of time. Even worse, it introduces uncontrolled dependencies to other parties, thus making the Matryoshka potentially vulnerable to misbehavior. The cores in consequence need a means to verify the integrity of their Matryoshkas. The Matryoshka verification protocol enables the core $\mathcal{V}$ both to verify the integrity of the trees, defined by the maximum number of shells and the spanning factor, through the Matryoshka, and the set of nodes that are registered as entrypoints. It thus allows for the detection of incidents in which another node for reasons of failure or misbehavior deviates from the protocol by either selecting much more than $Span$ neighbors for the next shell, or registered as entrypoint without authorization.
To get $\Omega_{\mathcal{V}}$, the set of authorized entrypoints to $\mathcal{V}$'s Matryoshka, $\mathcal{V}$ sends a $findOutShell$ message through the Matryoshka, generating different signed random numbers for each ith mirror node $\lambda_{\mathcal{V}} \in \Lambda_{\mathcal{V}}$: $Rnd_{S_{\mathcal{U}_{\mathcal{V}}},i}$. On reception of $findOutShell$, the authorized entrypoints create an $outShellMemb$ message, which contains $Rnd_{S_{\mathcal{U}_{\mathcal{V}}},i}$, a list of entrypoints retrieved from $\mathcal{V}$'s docks ($RetrEPlist$), and their node Id certificate. $outShellMemb$ is then encrypted for $\mathcal{V}$ in order to prevent any information about trust relationships on the trees through the Matryoshka being leaked, and sent back to $\mathcal{V}$. Collecting these responses, $\mathcal{V}$ can check the integrity of its Matryoshka by verifying that each tree leads to almost $Span^{TtlMatr}$ entrypoints. On detection of underpopulated trees or overbranching, $\mathcal{V}$ performs aimed Matryoshka updates. If one or a set $\Gamma_{\mathcal{V}}$ of third party nodes exist that advertise $\mathcal{V}$'s Matryoshka without being authorized nor part of it, they can be identified by $\mathcal{V}$, as they will be element of $RetrEPlist$ but not in the set of nodes that answered to the $findOutShell$ message:

$$\Gamma_{\mathcal{V}} = \bigcup_{\omega \in \Omega} RetrEPlist_{\omega} \setminus \bigcup_{\omega \in \Omega} NId_{\omega}$$

$\mathcal{V}$ in consequence sends an $intruders$ message to $\Theta_{\mathcal{V}}$ containing the signed set $\Gamma_{\mathcal{V}}$ of intruders. The authorized entrypoints of $\mathcal{V}$ forward this list of intruders to the docks $\mathcal{K}$, with which they are registered. These in consequence remove the intruders from their entrypoint tables and forward the information to the other registering nodes in the $RespArea$.

B.2 Specific vulnerabilities

Notwithstanding the fact that privacy, integrity and availability is assumed by Safebook, new vulnerabilities arise due to some exposures through the misuse of Safebook protocols by potentially malicious parties. This section presents vulnerabilities that are specific to Safebook protocols and the countermeasures that are implemented as part of Safebook.

B.2.1 Denial of Service and Traffic Analysis

No user can access the content that is exchanged between other parties. However, due to its cooperative nature in which all messages are forwarded by other participating nodes, Safebook is fundamentally vulnerable to black hole (a malicious node intercepting and dropping messages for a destination) and white hole (interception and monitoring for reasons of deriving statistical properties) attacks. For this purpose, a malicious node $\mathcal{M}$ would need to intrude the Matryoshka of the target $\mathcal{V}$. Since for $\mathcal{M}$ it is impossible to generate *pathReq* or *register* messages, and since $\mathcal{M}$ can not change its node Id in order to become a dock of $\mathcal{V}$, the only way of mounting this attack is by finding colluding docks of the target, or colluding users of the target's Matryoshka. The first approach leads to entrypoint table poisoning, the second implies misbehavior in some trust relationship.

EPT poisoning - Attacks External to the Matryoskha

Poisoning the entrypoint tables for a target node can only be mounted in collusion with one of the docks of the target's Matryoshka, $\mathcal{K}$.
Even a colluding $\mathcal{K}$ can not simply add a fake entry pointing to the attacking node $\mathcal{M}$ and advertise it to the *RespArea*, as the signature of $\mathcal{M}$ is needed. However, assuming $\mathcal{K}$'s EPT has $DhtKey_{\mathcal{V},1} = hash\,(Name_{\mathcal{V}})$, $\mathcal{K}$ could supply $\mathcal{M}$ with one of the signed random numbers it received by an authorized entrypoint of $\Theta_{\mathcal{V}}$. $\mathcal{M}$ in consequence could create, sign, and send a bogus *coreReg* message back to $\mathcal{K}$, or even to a dock $\mathcal{J}$ for a different registration key of $\mathcal{V}$. Since $\mathcal{M}$ provides a seemingly correct *coreReg* message, $\mathcal{J}$ (and $\mathcal{K}$) advertise the new EPT record to the respective *RespArea*. The complete set $DhtKey_{\mathcal{V}}$ is not stored anywhere in Safebook, other than at $\mathcal{V}$ itself. However, presuming $\mathcal{M}$ by external means got to know $DhtKey_{\mathcal{V}}$, it could, in collusion with $\mathcal{K}$, register its node Id as a valid entrypoint for all keys in $DhtKey_{\mathcal{V}}$ throughout the entire P2P system to increase chances to be on the forwarding path to $\mathcal{V}$ (cmp. fig.B.1). Even though this scenario describes

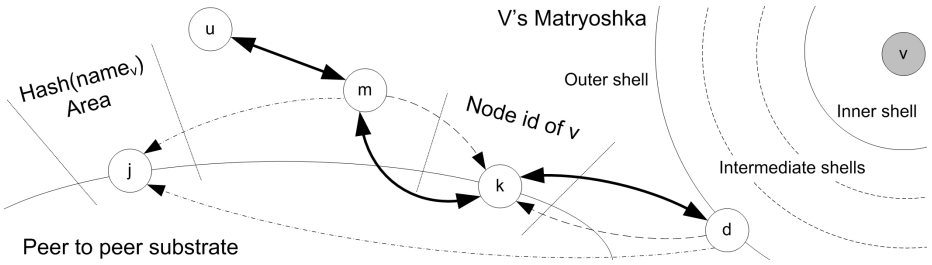

Figure B.1: A colluding intruder $\mathcal{M}$ in $\Theta_{\mathcal{V}}$.

a possible attack on Safebook, the consequences are neglectable for two reasons. Nodes requesting profile information are always supplied with the whole EPT and select arbitrary entrypoints from the list to pose the request to, thus limiting the chances for $\mathcal{M}$ to be selected. Additionally, the attack is detected by the Matryoshka verification protocol and the unauthorized entrypoints in consequence are removed from docks that are not colluding with $\mathcal{M}$.

Trusted Friend Misbehavior - Internal Attacks

A misbehaving prism in $\Theta_{\mathcal{V}}$ that mounts a black- or white hole attack is detected by the Matryoshka verification protocol, as well. As a matter of fact, if a node $\theta^j \in \Theta_{\mathcal{V}}$ placed in the jth shell of $\mathcal{V}$'s Matryoshka selects a number higher than $Span$ of next hops in order to attract more traffic (cmp. fig. B.2), its predecessor θ^{j-1} directly detects this misbehavior by simply checking the number of $outShellMemb$ messages received. If the number is too high, θ^j is obviously misbehaving. Its predecessor will purge it from the Matryoshka and it will not be selected a trusted next hop in the future. Even if the number is comparable to $Span$ and the predecessor is unable to detect the misbehavior, the exceeding leafs of the respective tree are identified as intruders and removed from $\Omega_{\mathcal{V}}$ as described in section B.1. In both the cases, the attack is detected and suppressed.

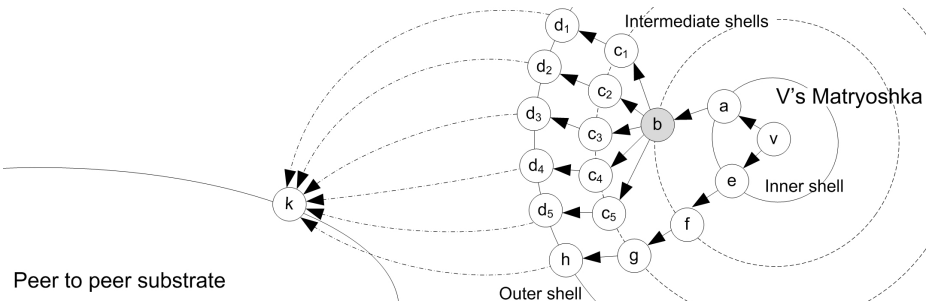

Figure B.2: A black hole $\mathcal{B}$ in $\Theta_{\mathcal{V}}$ (right).

Appendix C

Privacy Preserving Picture Sharing in Distributed OSNs

The problem of usage control, which refers to the control of the data after its publication, is becoming a very challenging problem due to the exponential growth of the number of users involved in content sharing. While the best solution and unfortunately the most expensive one to cope with this particular issue would be to provide a trusted hardware environment for each user, in this paper we address this problem in a confined environment, namely online social networks (OSN), and for the particular picture sharing application. In current OSNs, the owner of an uploaded picture is the only one who can control the access to this particular content and, unfortunately, other users whose faces appear in the same picture cannot set any rule. We propose a preliminary usage control mechanism targeting decentralized peer-to-peer online social networks where control is enforced thanks to the collaboration of a sufficient number of legitimate peers. In this solution, all faces in pictures are automatically obfuscated during their upload to the system and the enforcement of the obfuscation operation is guaranteed thanks to the underlying privacy preserving multi-hop routing protocol. The disclosure of each face depends on the rules the owner of the face sets when she is informed and malicious users can never publish this content in clear even if they

have access to it.

C.1 Introduction

Widespread adoption of Online Social Networks (OSNs) like Facebook[1], Twitter[2], LinkedIn[3], and recently Google Plus[4] is due to the ease of communication allowing users, even those with limited technical skills, to share a wide range of personal information with a theoretically unlimited number of partners. This advantage comes at the cost of increased security and privacy exposures, since they tend to disclose private personal information with little guard [89, 69, 126] and existing OSN applications seem to have an inherent economical interest in keeping this disclosure huge. Probably, such a strategy aims to attract other users and increase the OSN market value, that can reach, as in the case of Facebook in a deal of January 2011, up to 50 billion dollars[5].

Nevertheless, the overall protection offered by current centralized OSN on users personal private information is far from being satisfactory: unfortunately, all information cannot be protected and those that can be controlled are rarely protected by default [78]. Even with a very strong and specific access control policy, where the user who uploads or ***"posts"*** a content can, indeed, prevent unauthorized access, he unfortunately looses the control on it after its very first publication.

The problem of usage control which refers to the previously described issue is becoming a very challenging problem due to the exponential growth of the number of users involved in such content sharing applications. Recently, several peer-to-peer (P2P) based distributed online social networks (DOSN) have been proposed to preserve users' privacy ([57, 68, 42, 114, 76]). In all these solutions, users' data is not stored by a centralized OSN provider anymore. Such a DOSN can be considered as a good candidate for designing usage control mechanisms since it leverages on the collaboration of the users for any operation including data management and privacy protection. However, even in this distributed environment, usage control is difficult to achieve since a default mobile code protection mechanism does not exist and, hence, malicious users can manipulate the DOSN software running at their peer node. While the best solution to cope with such problems would be to provide a trusted

[1] http://www.facebook.com/
[2] http://twitter.com/
[3] http://www.linkedin.com/nhome/
[4] https://plus.google.com/
[5] http://www.bbc.co.uk/news/business-12106652

hardware environment for each user, it is unfortunately a very expensive alternative.

In this paper we propose to exploit the advantage of the underlying peer-to-peer architecture in DOSNs in order to enforce the control of the usage of some specific content, namely pictures. The proposed mechanism relies on the collaboration of nodes and control is enforced thanks to the forwarding of any packets towards several hops before reaching the final destination. The proposed usage control mechanism is designed over a recently proposed DOSN named as Safebook [57] which overcomes the problem of selfishness by leveraging on the real life social trust relationships among users. The underlying multi-hop forwarding solution can directly be used as a basis for the usage control mechanism. Differently from other P2P DOSNs, Safebook [57] overcomes the lack of cooperation among peers by leveraging on the real life social trust that is available as part of the very application. The untraceability of the communications during look-up and data retrieval operations is assured thanks to an additional feature of Safebook in that the messages between a requester node and a friend's node that serves the request always route through several hops in order to hide a user's social links that are reflected by the OSN graph. In a collaborative P2P DOSN such as Safebook, the multihop routing feature can provide the users with the usage control on their data.
In this paper we address the problem of usage control in Safebook focusing on pictures, as picture sharing is one of the most popular application offered by OSNs[6], and also one of the most exploited by malicious users: the privacy concerns derived by the misuse of stolen or accessed pictures are overblown [52], ranging from identity theft [39] to defamation. We propose, to the best of our knowledge, the first solution leveraging on peer collaboration guaranteeing data usage control on pictures in a P2P DOSN.

This paper is divided into five sections: section C.2 introduces the problem of usage control illustrated by picture sharing applications in online social networks. Section C.3 describes the proposed mechanism based on a multi-hop enforcement originating from the Safebook DOSN. Finally, the security and efficiency of this protocol are evaluated in section C.4.

[6] http://hbswk.hbs.edu/item/6156.html

C.2 Problem Statement

C.2.1 Usage control for picture sharing in online social networks

Usage control [93] becomes a mandatory requirement given the very large number of users sharing different types of content. The ideal goal of guaranteeing the control over any type of data in any type of platform seems very difficult to achieve. We address this problem in a confined environment which is online social networks and in the context of picture sharing since, as previously mentioned, this application is one of the most popular applications for OSN users.

Current picture sharing tools in online social networks allow users to upload any picture. Access rules to these pictures are defined by the owner of the picture that is the one who uploads it. This user has also some abilities to associate an area of the picture to a label: such a function, namely ***tag***, can be used to inform other users about their presence in the picture. This solution, recently patented by Facebook[7], seems unsatisfactory since users whose faces appear in pictures they do not own, are only informed about these pictures whenever they are "tagged"; they can further untag their faces if needed. Unfortunately, if users are not tagged in the picture, they will never be aware of these pictures. We assume that each person whose face appears in any picture should decide whether her face in that picture should be disclosed or not and therefore she should define the usage control policy regarding her own face.

C.2.2 Decentralized online social networks

As previously mentioned, online social networks severely suffer from the centralized control on users' data: potential misuse of private data by the OSN provider can be considered as a major threat in terms of privacy. In order to avoid such a centralized control by service providers over user data, some solutions [57, 68, 42, 114, 76] propose to design new applications based on a peer-to-peer architecture while leveraging real life social links to construct a network with trusted peers. The correct execution of any network/application operation depends on users' behavior. In order to achieve a good performance degree, these solutions define a threshold for the number of misbehaving users and analyze the trade-off between security and performance based on this degree: for example, in some solutions a

[7] http://www.redmondpie.com/facebook-awarded-patent-for-tagging-photos-and-digital-media/

packet must pass through a threshold number of nodes before reaching its destination in order to guarantee a certain security degree.

Such peer-to-peer online social networks can be considered as a good candidate for usage control mechanisms in picture sharing. In such an environment, a well behaving node would automatically obfuscate all faces in any picture it receives from other nodes. Therefore, given a threshold number of misbehaving or malicious nodes, in order to guarantee the correct execution of the usage control mechanism, the application can define a maximum number n_{max} of nodes a legitimate message has to pass through, before reaching its final destination. Among these n_{max} nodes at least one node should behave legitimately and apply the required protection operations. Additionally to the owner of the picture, only the owner of the face included in that picture should be able to have an initial access to the face in that picture. The further usage control rules for the dedicated face have to be defined by the corresponding user and the correct appliance of these rules should be verified at each node in the path towards the destination.

C.3 The proposed usage control mechanism

In this section, we describe a usage control mechanism enforced thanks to the cooperation among multiple users that perform multi-hop forwarding.

The idea of the proposed mechanism is to exploit the distributed nature of peer-to-peer online social networks and to leverage real-life social links to control the access to pictures: as opposed to centralized solutions, all operations are performed with the collaboration of multiple nodes. Thanks to this multi-hop enforcement in this distributed setting, cleartext pictures will only be accessible based on the rules defined by users whose faces figure in those pictures.

An interesting system that answers the previously described requirements is proposed in [57] as a distributed privacy preserving online social network named as *Safebook*. We briefly summarize its characteristics before presenting the main contributions of this paper.

C.3.1 Safebook: a P2P DOSN leveraging real life social trust

The main aim of Safebook is to avoid any centralized control over user data by service providers. Safebook relies on the cooperation among a number of independent parties that

are also the users of the OSN application. The correct execution of different services depends on the trust relationships among nodes which are by definition deduced from real-life social links. Tn order to prevent a malicious node from acting as a legitimate user and hence having access to the information related to its relations and data, Safebook defines for each user a particular structure named as ***Matryoshkas*** ensuring end-to-end confidentiality and providing distributed storage while preserving privacy. As illustrated in figure C.1,the Matryoshka of a user $\mathcal{V}$, namely the ***core***, is composed by several nodes organized in concentric shells. Nodes in the first shell are the real life friends of $\mathcal{V}$, and store her profile data to guarantee its availability. For this reason, nodes in the innermost shell are also called ***mirrors*** and serve requests if $\mathcal{V}$ is offline. If a requester $\mathcal{U}$ directly contacted one of $\mathcal{V}$'s mirrors, say $\mathcal{A}$, $\mathcal{U}$ would be able to infer the friendship relation between $\mathcal{V}$ and $\mathcal{A}$. To protect such an information, several multihop paths, ***chains*** of trusted friends, are built where every user's node selects among her own friends one or more next hops that are not yet part of the core's Matryoshka. $\mathcal{A}$ can then be seen as the root of a subtree with branching ***span***[8] whose leaves, namely the ***entrypoints***, lie in the outermost shell.

When a user $\mathcal{U}$ looks for $\mathcal{V}$'s data, her request is served by the entrypoints of $\mathcal{V}$'s Matryoshka and forwarded to the mirrors along these predefined path. The answer follows the same path in the opposite direction. To protect the user's privacy expressed by the links in the Matryoshka, not even the core of a Matryoshka knows its entire composition. Apart from the list of the entrypoints, that is publicly available, the core knows the composition of the first shell, and nothing about the intermediate ones. Nodes in the intermediate shells do not know one eachother and their understanding of the Matryoshka is limited to the previous and next hop of the path they belong to. Every user in Safebook can play different roles in different matryoshkas, but can be a core only for her own.

The list of the entrypoints is public, and is stored by the second component of Safebook, the ***P2P system***. By looking up for an hash value of a property of $\mathcal{V}$, such ash her full name, the P2P system provides the list of the entrypoints of $\mathcal{V}$'s Matryoshka. FigureC.2 resumes the data lookup process in Safebook.

Every user in Safebook has thus an identity in the Matryoshka overlay and in the P2P one. Nodes in the P2P overlay are arranged as in Kademlia[86], but all the requests are served recursively in order to obfuscate the peer identity of the real requester. To prevent malicious users from creating multiple identities, identifiers are granted and certified by

[8]for the sake of clarity, we will consider span=1 in the rest of the paper.

the last component of Safebook, the ***Trusted Identification Service***(TIS). The TIS is contacted only once during the user registration phase and does not impact the decentralized nature of Safebook's architecture since it is not involved in any data communication or data management operation.

We now present a new usage control mechanism taking Safebook as a basis.

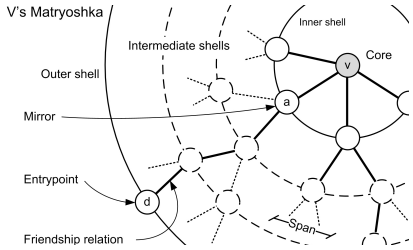

Figure C.1: The Matryoshka graph of a user $\mathcal{V}$, from [57]

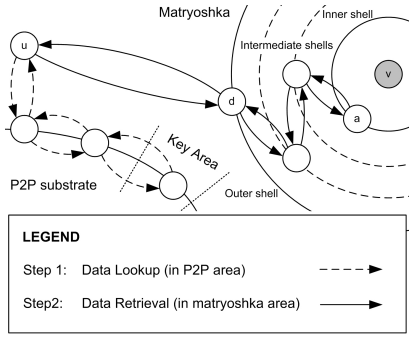

Figure C.2: Data lookup in Safebook, from [57]

C.3.2 Overview of the solution

In the particular environment of Safebook, a user can mainly play three different roles:

- she can **publish** a picture: in this case, she is represented as the core of her own Matryoshka and her friends will store this picture. She also has the option to tag some of her friends who might appear in the picture.
- She can act as a **forwarder** for some pictures: in this case, she belongs to the Matryoshka of either the owner of the picture or the owner of a face tagged in that picture.

- She can also request to **retrieve** some pictures which belong to one of her friends: in this case, she first needs to contact one of the entrypoints of the corresponding core in order to reach that particular user.

Dedicated tasks have been defined for each of these three roles. Indeed, some tasks are required at the stage of user registration. For example, before publishing a picture, the client has to perform some picture obfuscation operation. Similarly, as a forwarder, the node has to perform some verification operations in order to check whether the picture it is forwarding is correctly "protected" or not. All these tasks will be described in detail in the following section.

In the sequel of the paper, we use the following notation:

- $\mathcal{L}^{\mathcal{P}}$ denotes the regions in a picture $\mathcal{P}$ where a face appears;
- $\mathcal{F}^{\mathcal{P}}$ denotes the set of faces that appear in $\mathcal{P}$;
- $f_{\mathcal{V}}^{\mathcal{P}}$ denotes user $\mathcal{V}$'s face in $\mathcal{P}$;
- $\mathcal{f}_{\mathcal{V}}^{\mathcal{P}}$ denotes $\mathcal{V}$'s face features which are used for face detection algorithms;
- $\left\{\phi_{\mathcal{V}}^{+}, \phi_{\mathcal{V}}^{-}\right\}$ denotes user $\mathcal{V}$'s public and private keys, respectively;
- a message M signed using user $\mathcal{V}$'s private key $\phi_{\mathcal{V}}^{-}$ is denoted by $\{M\}_{S_{\phi_{\mathcal{V}}}}$;
- $E_K\{M\}$ denotes the encryption of a message M encrypted with the symmetric key K.

C.3.3 Solution description

User registration

Whenever a user $\mathcal{V}$ registers to Safebook and joins the network, she first generates a pair of public and private keys $\left\{\phi_{\mathcal{V}}^{-}, \phi_{\mathcal{V}}^{+}\right\}$ and sends the public key $\phi_{\mathcal{V}}^{+}$ and some samples of her face $f_{\mathcal{V}}$ to an off-line trusted third party, namely the ***Feature Certification Service***(FCS). The FCS generates a certificate for $\mathcal{V}$ denoted by $Cert\left(\mathcal{f}_{\mathcal{V}}, \phi_{\mathcal{V}}^{+}\right)$, which proves that the user with some face features $\mathcal{f}_{\mathcal{V}}$ owns the public key $\phi_{\mathcal{V}}^{+}$. This face feature certification phase is performed only once and the user does not need to contact the third party anymore.

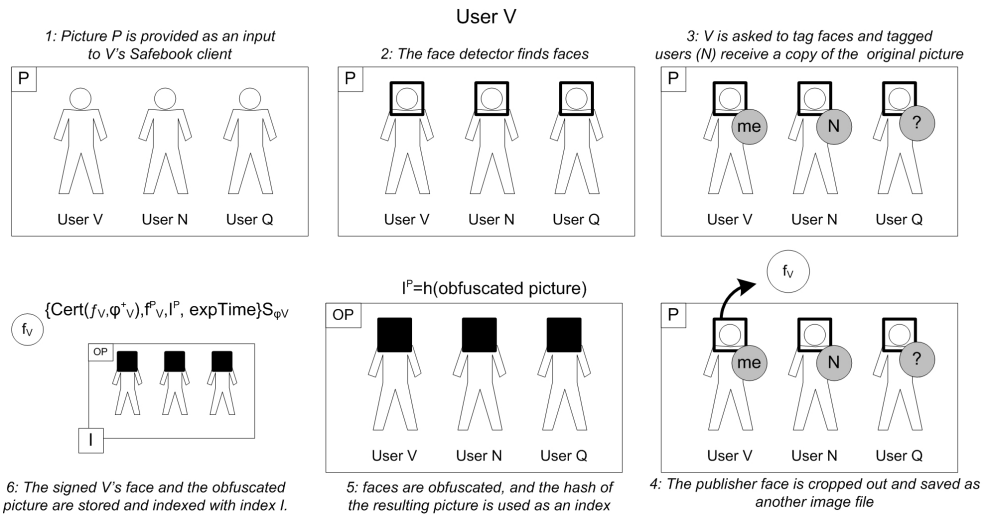

Figure C.3: Picture publication steps for $\mathcal{V}$, with $\mathcal{V}$'s face $f_{\mathcal{V}}$ made publicly available: 1- picture input; 2- face detection; 3- face tagging; 4- face extraction; 5- face obfuscation; 6- picture and publisher face publication.

Picture publication

To make her picture $\mathcal{P}$ available in the network, the publisher $\mathcal{V}$ has to perform the following main tasks:

- *Picture insertion and face detection*: To publish her picture $\mathcal{P}$, the user $\mathcal{V}$ provides $\mathcal{P}$ to her Safebook client. One of the main components the system consists of the `face detection` mechanism. The face detector aims at finding the presence of faces in the input picture $\mathcal{P}$ and, if this is the case, it returns their location $\mathcal{L}^{\mathcal{P}}$.

  Faces can vary in size, shape and color, and must be detected regardless of their position, orientation and light conditions. Existing face detection algorithms such as in [122] can directly be used in the proposed system. Their robustness directly affects the privacy achieved by this scheme. We assume that face detection algorithms are secure enough. Their design is out of the scope of this paper.

- *Picture tagging*: When the face detector derives $\mathcal{L}^{\mathcal{P}}$, the client uses the second component, namely the `face extractor`, which is in charge of copying every face $f_i^{\mathcal{P}} \in \mathcal{F}^{\mathcal{P}}$ detected in $l_i^{\mathcal{P}} \in \mathcal{L}^{\mathcal{P}}$ in a separate file.

  After this extraction task, the publisher $\mathcal{V}$ is asked to tag each face, i.e. to associate

every $f_i^{\mathcal{P}}$ to a profile in $\mathcal{V}$'s contact list. If a face $f_{\mathcal{N}}^{\mathcal{P}}$ is tagged with the profile of its owner $\mathcal{N}$, $\mathcal{N}$ receives a copy of the original picture $\mathcal{P}$ [9] and can decide to publish it again on her own profile. Furthermore, the publisher also decides whether her own face can be made available for the network or not.

- *Picture publication*: Once all known faces $f^{\mathcal{P}} \in \mathcal{F}^{\mathcal{P}}$ are tagged, the client can execute its last component which is the `face obfuscator`: The face obfuscator transforms the face location areas $\mathcal{L}^{\mathcal{P}}$ to uninterpretable areas using any human or computer vision algorithm, thus generating an obfuscated picture $\mathcal{OP}$. In our solution, the face obfuscator simply replaces every pixel in $\mathcal{L}^{\mathcal{P}}$ with a black one. The obfuscated picture can thus be seen as the original one with black shapes hiding every detected face. From the resulting obfuscated picture $\mathcal{OP}$, an unambiguous picture identifier I is computed as $I^{\mathcal{P}} = h(\mathcal{OP})$, where $h(\cdot)$ denotes a cryptographic hash function. $\mathcal{V}$'s face $f_{\mathcal{V}}^{\mathcal{P}}$ is then signed together with the certificate $Cert\left(f_{\mathcal{V}}, \phi_{\mathcal{V}}^{+}\right)$, the identifier $I^{\mathcal{P}}$, and an expiration time $expTime$.
  Finally, $\left\{Cert\left(f_{\mathcal{V}}, \phi_{\mathcal{V}}^{+}\right), I^{\mathcal{P}}, f_{\mathcal{V}}^{\mathcal{P}}, expTime\right\}_{S_{\phi_{\mathcal{V}}}}$ and $\mathcal{OP}$ are published. [10]

- *Picture advertisement*: Once advertised by $\mathcal{V}$ about the presence of $\mathcal{P}$, a user $\mathcal{N}$ can control the disclosure of her face $f_{\mathcal{N}}^{\mathcal{P}}$ in that picture.
  $\mathcal{N}$ may decide to make $f_{\mathcal{N}}^{\mathcal{P}}$ publicly available, and publish $\mathcal{OP}$ together with the following signed message:

$$\left\{Cert\left(f_{\mathcal{N}}, \phi_{\mathcal{N}}^{+}\right), I^{\mathcal{P}}, f_{\mathcal{N}}^{\mathcal{P}}, expTime\right\}_{S_{\phi_{\mathcal{N}}}}$$

  If $\mathcal{N}$ wishes to disclose this picture to a subset of its contact list only, she can encrypt the corresponding message with a symmetric key K previously distributed to the dedicated users. In this case, $\mathcal{N}$ will publish $\mathcal{OP}$ together with $E_K\left\{f_{\mathcal{N}}^{\mathcal{P}}\right\}, I^{\mathcal{P}}$.

Picture publication and advertisement actions are illustrated in figures C.3 and C.4, respectively.

[9] This doesn't violate $\mathcal{V}$'s privacy, since $\mathcal{V}$ aims at making $\mathcal{P}$ publicly available.
[10] In Safebook, this phase corresponds to the storage of the picture at $\mathcal{V}$'s friend nodes.

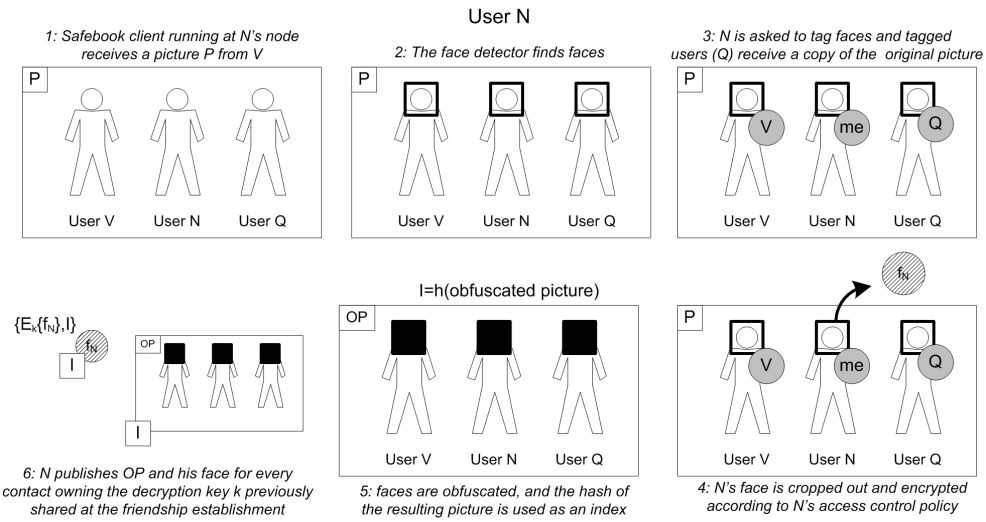

Figure C.4: Public picture advertisement: 1- N is informed about P; 2- face detection; 3- face tagging; 4- publisher face extraction; 5- face obfuscation; 6- picture and N's face publication according to N's access control policy on her face.

Forwarding pictures

Every intermediate node $\mathcal{T}$ storing or forwarding an obfuscated picture $\mathcal{OP}$ runs by default the face detector and obfuscator components on $\mathcal{OP}$. These tasks ensure the required privacy property in case some clients are manipulated by malicious nodes.

When storing or forwarding a user $\mathcal{V}$'s publicly available face, a legitimate node $\mathcal{T}$ first checks the validity of the signature $S_{\phi_{\mathcal{V}}}$, the expiration time, and the relation between the face features $f_{\mathcal{V}}$ in the certificate and the ones extracted from $f_{\mathcal{V}}^{\mathcal{P}}$. In case of verification failure, $\mathcal{V}$'s face is dropped.

Picture retrieval

To retrieve $\mathcal{V}$'s pictures, a user $\mathcal{U}$ who is not included in $\mathcal{V}$'s contact list sends a picture request *pctReq* message to $\mathcal{V}$ and receives a set of identifiers $I^{\mathcal{P}_j}$ related to $\mathcal{V}$'s publicly available pictures $\mathcal{P}_j$. $\mathcal{U}$ then asks for the identifiers she is interested in. For every identifier $I^{\mathcal{P}_j}$ $\mathcal{U}$ retrieves an obfuscated picture $\mathcal{OP}_j$

and the message

$$\left\{Cert\left(f_{\mathcal{V}}, \phi_{\mathcal{V}}^{+}\right), I^{\mathcal{P}_j}, f_{\mathcal{V}}^{\mathcal{P}_j}, expTime_j\right\}_{S_{\phi_{\mathcal{V}}}}$$

containing $\mathcal{V}$'s publicly available face in that picture. When interacting with her friend $\mathcal{N}$, $\mathcal{U}$ sends him a picture request containing some secret s, and receives a list of picture identifiers $I^{\mathcal{P}_j}$ associated to pictures of $\mathcal{N}$, which are either publicly available, or available to those contacts knowing the secret s, only. $\mathcal{U}$ then detects a match in $I^{\mathcal{P}}$ between the identifiers retrieved from $\mathcal{V}$ and $\mathcal{N}$, and, as she previously received $\mathcal{OP}$ from $\mathcal{V}$, she now asks for the missing information $f^{\mathcal{P}}_{\mathcal{N}}$. At the reception of $piRes = \left\{E_K\left\{f^{\mathcal{P}}_{\mathcal{N}}\right\}, I^{\mathcal{P}}\right\}$, $\mathcal{U}$ can retrieve $f^{\mathcal{P}}_{\mathcal{N}}$ since she already owns the appropriate decryption key K shared at the friendship establishment with $\mathcal{N}$.

C.4 Evaluation

In this section, we evaluate the proposed mechanism with respect to different security issues such as eavesdropping, unauthorized access or collusion attacks. The impact of these attacks is evaluated based on existing social graphs: in September 2005, Facebook published anonymous social graphs of 5 universities in the United States[11]: California Institute of Technology (Caltech), Princeton University (Princeton), Georgetown University (Georgetown), University of North Carolina (UNC), Oklahoma University (Oklahoma). Each graph is represented by an adjacency matrix A whose non diagonal elements a_{ij} are set to one if user $\nu_i \in V$ is a friend of user $\nu_j \in V$, or zero otherwise. As each adjacency matrix is symmetric, the represented social graph is undirected.

Before presenting the evaluation results, we briefly discuss about the feasibility of the proposed usage control mechanism. **Feasibility** The feasibility of the proposed usage control mechanism depends on the robustness and speed of the face detection and verification procedures and on the feasibility of the DOSN at its basis, in our case Safebook.

Face detection [116] and face recognition [125] procedures nowadays run in real time in common personal computers. Most of them [66, 85] make an intensive use of Scale Invariant Feature Transform (SIFT) [84], a well known techique used to extract view-invariant representations of 2D objects. Recognition rates of these solutions raise up to 95% in well known databases such as the Olivetti Research Lab (ORL) one [66, 85]. Other techniques can also be used to improve the recognition rate [115] at the expenses of a bigger face feature descriptor.

[11] http://people.maths.ox.ac.uk/ porterm/data/facebook5.zip

The proposed usage control scheme does not put any constraints on the face detection and recognition architecture: when the adopted face descriptor is bigger than the face image itself, a reference face image $r_{\mathcal{V}}$ rather than the feature descriptor $f_{\mathcal{V}}$ itself can be certified by the FCS. This change does not have a concrete impact on the time necessary to compare the reference face in the certificate with that one a user wants to make publicly available.

The feasibility of Safebook has been presented in [53]. The study discusses an inherent tradeoff between privacy and performance: on one hand, the number of shells in a user's Matryoshka should be defined as large as possible to enforce privacy in terms of communication obfuscation and protection of the friendship links, but small enough to offer a better performance in terms of delays and reachability; on the other hand, increasing the number of shells after a certain trheshold does not increase the privacy anymore. Such a threshold depends on the social network graph itself [54], more precisely on the number of hops after which a random walk on the social network graph approximates with a pre-defined error its steady state distribution [90]. In this case, the endpont and the startpoint of the random walk are uncorrelated.

Based on the results of the study in [53], we conducted our experiments by setting a number of shells as high as 4. We assumed the number of online nodes as high as 30%. For every social network dump, we measured the average number of user's friends d, the average number of chains p a user managed to build, and the average number of Matryoshka q a user is part of. Figure C.5 shows p is always about one fourth of d and both p an q increase linearly with d by a factor of around 1.2.

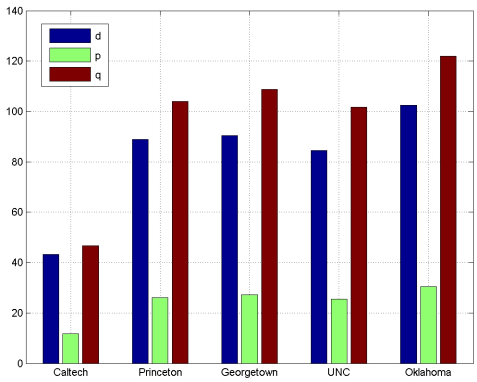

Figure C.5: Average outdegree, average number of 4-hops chains, average number of served Matryoshka for different social network graphs when 30% of nodes are on-line.

Unauthorized picture broadcast Even if malicious users manipulate the underlying software, broadcasting cleartext faces is prevented thanks to the collaborative multi-hop enforcement scheme. Indeed, it is assumed that there is at least one legitimate node which will execute the correct verification operations and the corresponding transformations in order to protect forwarded packets. Nevertheless, Safebook allows the forwarding of encrypted information to a subset of friends; a malicious member $\mathcal{V}$ may exploit this possibility to further send packets to all of its friends. However, $\mathcal{V}$ may need to set-up a virtual server and establish friendship relationships with all users to provide a picture $\mathcal{P}$ to all the users. This kind of attacks can be prevented by setting a maximum proper rate on friendship requests. The malicious node would need to design some server advertisement mechanisms using additional out of band information exchange (outside the Safebook network). Furthermore, a malicious node $\mathcal{V}$ may also collude with one of her contacts $\mathcal{C}_1$, asking him to manipulate her Safebook client, in order to encrypt and republish an unauthorized picture $\mathcal{P}$ at her own profile. If recursively repeated, this hop-by-hop collusion through the social network graph may disclose $\mathcal{P}$ to all the contacts of every colluder $\mathcal{C}_n$. This attack again requires the manipulation of the OSN client itself and the impact of such an attack would only be important if the number of malicious users is very large. Fortunately, a massive scale collusion attack may end on the creation of an environment where the adversary may in turn be a victim for her own private data. The impact of collusion attacks is also analyzed further in this section.

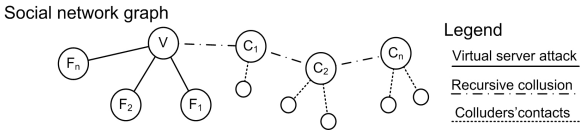

Figure C.6: Unauthorized picture broadcast by friendship relations establishment between a malicious $\mathcal{V}$ and any users $\mathcal{F}_i$, or by recursive collusion with nodes $\mathcal{C}_i$ not necessarily belonging to $\mathcal{V}$'s contact list.

Protection against collusion As previously mentioned, the enforcement of the control on the usage of a given picture is based on the collaboration of users and the correct execution of the previously described operations. However, some users can still be malicious and avoid obfuscating some public pictures. To evaluate the impact of misbehavior and collusion, we have simulated the process of Matryoshka creation in which the chains leading from the mirrors to the corresponding entrypoints are built. We assume 30% of the nodes is online,

and 10% of the nodes misbehave. We also assume that misbehaving nodes are always online. In case of absence of collusion, misbehaving nodes are randomly selected. In case of collusion, misbehaving nodes are selected between and in the friendlists of all the nodes with higher weight $w_{\mathcal{V}}$ defined as:

$$w_{\mathcal{V}} = d_{\mathcal{V}} * cc_{\mathcal{V}}$$

where $d_{\mathcal{V}}$ represents the degree of node $\mathcal{V}$, i.e. the number of $\mathcal{V}$'s contacts, and $cc_{\mathcal{V}}$ its clustering coefficient, i.e. the ratio between the number of existing links between $\mathcal{V}$'s contacts divided by the number of possible links that could exist. We define as ***compromised chain*** a chain entirely composed by misbehaving nodes.

Figure C.7 shows the number of compromised chains in case of absence of collusion, m_u, and in case of collusion, m_c. One can see that in case of collusion, this number drastically increases by a factor ranging between roughly 4 and 32. When assuming the fraction of misbehaving nodes as high as 25% (see figure C.8), almost all the chains get compromised.

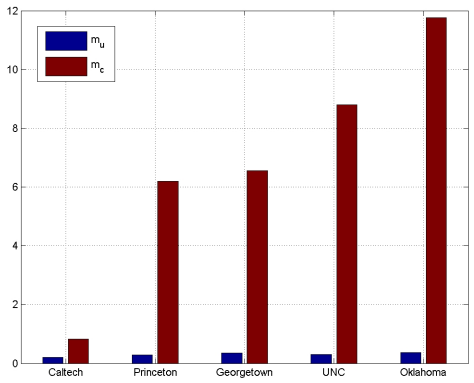

Figure C.7: Average number of compromised chains when 10% of nodes misbehave for different social network graphs.

Data confidentiality and Anonymity Given an obfuscated picture $\mathcal{OP}$, it should be impossible to retrieve any information about users whose depicted faces are not made publicly available. Since by the very design of the Safebook client, there is no way to query the OSN for the identity of the users whose faces are missing, it is, indeed, not possible for an adversary to extract any useful information from an obfuscated picture. Only friends of a user $\mathcal{N}$ can discover this information and retrieve $f^{\mathcal{P}}_{\mathcal{N}}$. Whenever $\mathcal{N}$'s friend $\mathcal{U}$ receives the list of identifiers of the pictures she is allowed to access, she checks the list of picture

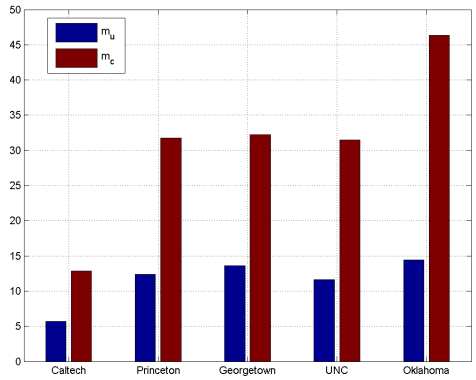

Figure C.8: Average number of compromised chains when 25% of nodes misbehave for different social network graphs.

identifiers in her cache and may find a match for $I^{\mathcal{P}}$. In this case, $\mathcal{U}$ can ask and obtain $f^{\mathcal{P}}_{\mathcal{N}}$, encrypted with a key K she received from $\mathcal{N}$ previously.

Invalid tagging and picture republishing A malicious user $\mathcal{V}$ may associate $\mathcal{N}$'s face $f^{\mathcal{P}}_{\mathcal{N}}$ with her own profile while tagging a picture $\mathcal{P}$. Nevertheless, $\mathcal{V}$ will not manage to make $f^{\mathcal{P}}_{\mathcal{N}}$ publicly available, unless the features of $f^{\mathcal{P}}_{\mathcal{N}}$ are similar enough to that ones in $Cert\left(f_{\mathcal{V}}, \phi^{+}_{\mathcal{V}}\right)$. However, according to the Face Recognition Vendor Test (FRVT) of 2006 [96] the false rejection rate for a false acceptance rate of 0.001 is 0.01 for state-of-the-art face recognition algorithms. A picture $\mathcal{P}$ can be accessed and republished by a third node $\mathcal{Y}$ that does not appear on it. $\mathcal{Y}$ can in fact store in her profile the obfuscated $\mathcal{OP}$ and any publicly available face

$$\left\{Cert\left(f_{\mathcal{X}}, \phi^{+}_{\mathcal{X}}\right), I^{\mathcal{P}}, f^{\mathcal{P}}_{\mathcal{X}}, expTime\right\}_{S_{\phi_{\mathcal{X}}}}$$

for that picture.

Limitations

In order for the intermediate nodes to verify whether the rules are followed or not, the picture should of course not be encrypted (even if there is obfuscation). This security mechanism cannot be implemented over encrypted messages. However, if a malicious user would want to encrypt the picture in order to circumvent the usage control mechanism, only nodes with corresponding decryption keys can have access to these pictures. Such an attack would require an important communication overhead and its impact on the security would

not be that important.

Figure C.9 summarizes the characteristics of the proposed solution.

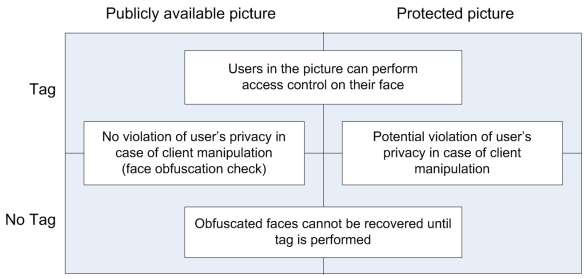

Figure C.9: Spread of information vs usage control: in case of unprotected picture publication, automatic face obfuscation is guaranteed by peer collaboration even when there is software manipulation; in case of encrypted publication, the software manipulation may violate user's privacy, but with limited impact within the DOSN.

C.5 Conclusion and Future Work

As it is not feasible to design a perfect usage control mechanism to control the management of any type of data in any environment, we proposed a preliminary solution dedicated to picture sharing tools widely used in the context of online social networks. Although it might be feasible to design such a mechanism in a centralized environment, current OSN providers are not yet interested in such protection mechanisms. On the contrary, decentralized, P2P based online social networks rely on the collaboration of users for any operation including data management and security. The proposed usage control mechanism takes advantage of this inherent cooperation between users and ensures the enforcement on the control of the pictures thanks to a dedicated multihop picture forwarding protocol. The message has to follow a dedicated path of sufficient intermediate nodes which perform the dedicated tasks defined in the usage control policy before reaching its final destination. Thanks to this multihop enforcement mechanism, users whose face appears in a given picture will be able to control its usage in the very beginning stage of its publication. Nevertheless, the protection of the picture and the enforcement of this control is only efficient in the confined environment of the DOSN and when pictures are not encrypted; however, the impact of attacks launched outside this environment or aiming at encrypting the message is very limited within the DOSN.

In the future work, we plan to evaluate the scalability and the performance impact of the proposed solution, and integrate its features in the current Safebook prototype[12].

[12] http://www.safebook.us/home.php?content=prototype

Appendix D

PRICE: PRivacy preserving Incentives for Cooperation Enforcement

Many incentive mechanisms have been proposed to foster cooperation among nodes in Peer-to-Peer (P2P) networks. Unfortunately, most of existing solutions rely on the existence of an online centralized authority that is in charge of a fair distribution and transaction of credits (incentives) between peers. Such centralized mechanisms mainly suffer from privacy leakage and single point of failure problems. To cope with these problems, we propose to take advantage of the distributed nature of P2P networks in order for the peers to take care of credit-based operations. Cheating and other DoS attacks are prevented thanks to a threshold security mechanism where the operation should be approved by a predefined certain number of peers. The main novelty of the proposed mechanism is the fact that a "credit" is assigned to some peers using distributed hash tables, hence, peers can follow and control the history of operations with respect to this credit, only. Thanks to this new approach, a malicious node cannot easily keep track of all operations originating from a single node and the impact of cheating or similar attacks would be strongly reduced.

D.1 Introduction

Peer-to-Peer systems are nowadays broadly adopted to provide several services such as file sharing [86, 98], data storage [71, 120], secure communication [103, 49], social networking [57, 42, 87]. The correct execution and the availability of such services strongly depend on the collaboration among peers. Several studies [32, 108, 82] pointed out that, unfortunately, peers often engage in ***free riding***, i.e. they try to consume as more resources as possible and on the contrary, contribute with as few resources as possible. This kind of selfish behavior has a strongly negative impact on the overall performance of the system and may even lead to its failure.

Several cooperation enforcement solutions [41, 88, 44, 127, 113, 117, 37] have been proposed to foster cooperation among peers in P2P or Mobile Ad-hoc Networks (MANETS). Most of them rely on credit-based mechanisms whereby nodes receive a reward whenever they cooperate for the execution of the requested action. These credit-based incentive mechanisms often rely on the existence of a centralized authority that ensures a fair distribution of the credits and also acts as a mediator in case of litigation during transactions. The adoption of such a centralized authority raises serious security and privacy concerns: Indeed, this online trusted authority has a direct access on the history of actions of any peer since it is in charge of distributing rewards corresponding to each granted service. Therefore, as in all existing centralized services, current credit-based incentive mechanisms suffer from privacy problems such as traceability or monitoring [117, 37].

In this paper, we propose PRICE, an incentive mechanism that can be adopted as a built-in service for any DHT-based P2P system. PRICE takes advantage of the distributed nature of the P2P network itself to manage credits and ensures the correctness and the security of transactions. The management of credits, defined as "coins" in PRICE, is distributed among peers in the network based on the use of the inherent P2P functionality, that is, a distributed hash table (DHT). A coin transaction only succeeds if a quorum among a predefined number of peers agrees on it. Although the task of credit management is distributed among several peers and therefore it can decrease the privacy of the system, PRICE offers an original approach by assigning the management of each single coin to a different set of peers instead of the account of a given peer. Therefore, on the one hand, no entity in the system is able to discover the total amount of credits a user holds; hence, as opposed to centralized solutions, a user's history of actions cannot be traced by any node; on the other

hand, even if there is a privacy leakage with respect to a single transaction, this will not a have an impact on the privacy of the user's overall actions. The association between the credit involved in the transaction and the peers that are responsible for the transaction itself is based on the pseudorandomness of the security functions used for the generation of the coins.

The rest of this paper is organized as follows: section D.2 introduces the main security and privacy challenges of an incentive mechanism. Section D.3 proposes an overview of PRICE which is, then, formally described in section D.4. The evaluation of PRICE is discussed in section D.5

D.2 Problem Statement

D.2.1 Cooperation enforcement in P2P networks

The correct execution of many P2P services relies on the collaboration of nodes involved in the network. Cooperation enforcement mechanisms would encourage nodes to perform a fair share of basic operations. Inducing cooperation between nodes can be based either on some reputation or rewarding mechanisms. Reputation mechanisms [41, 88, 105, 61] ensure that each node accepts to cooperate with its neighbors based on the past behavior of the latters. On the other hand, credit based schemes [43, 127, 117, 37] provide node collaboration by rewarding cooperating nodes with a certain amount of credits that they further can use for their own benefit. Credits can be in the form of E-cash [47, 48] or a tradable good/service such as future cell phone call time.

D.2.2 Credit-based incentive mechanisms

Existing rewarding mechanisms encourage nodes to cooperate in performing the required operations (forwarding, data storage, etc.). These solutions consist of virtual currencies that nodes receive whenever they cooperate. Unfortunately, because such solutions suffer from lack of fairness, they require the existence of a centralized online trusted third party mainly for credit management. Indeed, for example, in [127], the rewarding mechanism named as Sprite requires an immediate reachability of the TTP defined as Credit Clearance Service (CCS). Such mechanisms also suffer from the single point of failure problem as nodes must contact the CCS whenever they forward the message in order to receive their rewards.

Furthermore, this centralized entity has full control over these rewards and keeps track of any node's actions.
Distributed credit-based incentive mechanisms such as Karma [117] solve the single point of failure problem, since a set of peers in the DHT, namely the bank, stores a user's account. Still, this set of notes can trace the user's actions.

D.2.3 Security and Privacy Challenges

As for any credit-based mechanism, a credit based incentive mechanism should prevent nodes from cheating. Therefore the proposed mechanism should exhibit the following security properties:

- ***unforgeability***: a valid credit cannot be forged by any user;
- ***no double spending***: credits resulting from duplication or copying of valid credits should be prevented or immediately detected;
- ***communication confidentiality***: any action taken under the incentive mechanism should not leak information regarding the underlying service application;
- ***transaction untraceability***: a selfish or malicious user should not be able to monitor any legitimate's users account.

D.3 Solution Overview

In order to cope with the previously described security and privacy challenges we propose PRICE, a credit-based incentive mechanism whereby, as opposed to existing centralized solutions, the management of the credits, defined as ***coins***, is distributed among several peers in the network. While this distributed mechanism allows a better robustness of the system and prevents the problem of single point of failure, the privacy challenge becomes even more important since many peers can be aware of others' activities. The proposed management of coins hence prevents such a possible leakage by assigning different sets of peers for each coin rather than defining one responsible per node's account (activities). The peer assignment follows the inherent nature of P2P by taking advantage of distributed hash tables (DHT). In the following sections, the proposed mechanism is summarized and illustrated with a scenario.

D.3.1 Environment

As previously mentioned, the correct execution of PRICE relies on the use of a distributed hash table (DHT) based P2P network where every peer node is also considered as the application user. A peer is assigned to a unique identity, the ***Peer Identifier***, and the assignment of coins to peers is managed by the DHT: in addition to P2P services such as data storage or data retrieval, peers also participate on the management of coins. To prevent DoS attacks including Sybils [62] or eclipse [109] the mechanism defines an off-line ***Trusted Identification Service*** (TIS) which mainly computes the Peer Identifier and ensures that this value is unique and is assigned to its corresponding peer by generating a cryptographic certificate over the identifier. Any attack due to the multiple identities creation or identity manipulation is thus unfeasible in PRICE. In order to ensure the security of the rewarding mechanism, coins are generated and signed by a trusted entity named as ***Coin Generator*** (CGEN) whose unique role is to ensure the correctness and validity of the coin.

D.3.2 Scenario

In the following, we present a scenario in which two users Alice and Bob take part in a P2P network offering data storage services and use PRICE to manage their transactions. In the P2P network, let Alice be a user interested in storing her file using Bob's resources. Whenever Alice sends her request to Bob, she grants him with a ***coin*** for this additional service. This transfer should of course be considered as valid and Bob should be able to verify that he is the new owner of the coin. Therefore there is a strong need for defining a third entity or a witness to validate such a transfer. In the proposed mechanism, a set of peers is assigned for this role and they are defined as ***notaries***. The track of each coin is kept by a different set of notaries. Therefore, whenever Alice would like to grant a coin to Bob, she contacts one of the notaries corresponding to this specific coin, namely the ***caretaker notary***, and informs it about the new ownership of the coin. With the agreement of the other notaries, the caretaker then sends a proof of this transfer to Bob. Even if a malicious node succeeds in discovering current transfer of this coin, it will not be able to trace all actions taken by Alice or Bob since the management of each coin is assigned to different notaries.

Based on this scenario which is illustrated in figure D.1, we identify three main steps among the proposed incentive mechanism:

- ***account creation***, whereby a newcomer receives his peer identifier (PI) from the TIS and an initial number of coins from the CGEN;
- ***payment order***, whereby the newcomer requests to grant a coin to a beneficiary by sending a PAY message to the caretaker notary;
- ***payment notification***, whereby the caretaker notary collects the agreement of a sufficient number of notaries, and informs the payer and the beneficiary about the success of the transaction.

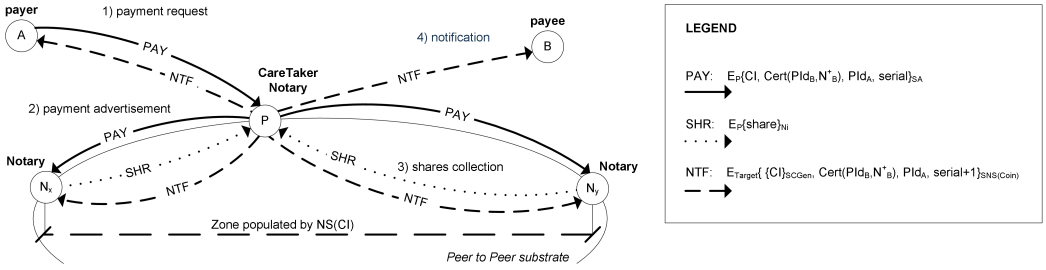

Figure D.1: Payment scheme.

D.4 Description

In this section, we first define the security properties of a coin and introduce the main components of PRICE which are the Coin Generator, the DHT based P2P substrate, and the Trusted Identification Service. We then formally describe the three steps of the proposed incentive mechanism, namely, **the account creation**, **the payment order** and the **payment notification**.

D.4.1 Preliminaries

The P2P substrate and the Trusted Identification Service

In the DHT, every user is associated to a peer node by a unique Peer Identifier PI which is computed by the TIS. By granting a certificate together with every identifier, the TIS protects the PRICE mechanism from different DoS attacks such as Sybil [62], impersonation, or eclipse[109]. Following the very definition of a DHT, a PI is defined as a number over a "key space" in order to facilitate the functions of data lookup.

The rewarding mechanism and the Coin Generator

PRICE relies on a specific implementation of rewards which are named as ***coins***, generated by a trusted entity, the ***Coin Generator***, and are defined by a tuple with the following parameters:

- a ***Coin Identifier*** CI, which is a pseudo-random number generated by the Coin Generator over the keyspace K and will be used as the input of a **coin lookup** operation in the P2P network;
- the signature of CI computed by the Coin Generator with its secret key as a proof of the validity of the coin.

Thanks to the security of the pseudo-random generator used by the CGEN, a coin c is unique. The signature of the CGEN provides the protection against forging attacks.

Table D.1 summarizes the notation used for the description of PRICE.

D.4.2 Account creation

Whenever a new user enters the system, it first needs to receive its peer identifier and its set of initial coins. Therefore, the account for a new user $\mathcal{A}$ is created in three separate steps: 1) identity creation and authentication, where $\mathcal{A}$ obtains its identifier, 2) P2P substrate join, where $\mathcal{A}$ takes its place in the DHT, and 3) welcome coin attribution, where $\mathcal{A}$ is granted with a predefined number of coins by the CGEN.

Identity creation

In order to get its peer identifier, $\mathcal{A}$ generates an asymmetric keypair $\mathcal{K}_{\mathcal{A}} = \left\{K_{\mathcal{A}}^{-}, K_{\mathcal{A}}^{+}\right\}$ and sends an out of band request to the TIS. This request contains $\mathcal{A}$'s public key $K_{\mathcal{A}}^{+}$, together with his claimed identity $ID_{\mathcal{A}}$. Once this request is received, the TIS generates $\mathcal{A}$'s peer identifier as $PI_{\mathcal{A}} = h_{MK}(ID_{\mathcal{A}})$, where $h_{MK}(\cdot)$ is a keyed hash function whose master secret MK is known only by the TIS and nobody else. The TIS sends back to $\mathcal{A}$ the certificate $Cert(PI_{\mathcal{A}}, K_{\mathcal{A}}^{+})_{S_{TIS}}$ and informs the coin generator a new user has joined the system.

Welcome coins attribution

As the CGEN receives a message from the TIS stating a new user $\mathcal{A}$ has arrived, it generates a new set of coins $\{c_i\}$, and provides $\mathcal{A}$ with this set by sending him a PAY message for every

Table D.1: Notation

$\mathcal{A}$	node $\mathcal{A}$
$PI_{\mathcal{A}}$	peer identifier of $\mathcal{A}$
$K^-_{\mathcal{A}}, K^+_{\mathcal{A}}$	private and public keys of $\mathcal{A}$
$\{\cdot\}\, S_{\mathcal{A}}$	signature generated with the private key of $\mathcal{A}$
$Cert(I, K^+)$	certificate associating an identifier I to a public key K^+
MK	master key
$h_{MK}(\cdot)$	keyed hash function with master secret MK
$E_{\mathcal{B}}\{M\}_{S_{\mathcal{A}}}$	message M signed by $\mathcal{A}$ and encrypted for $\mathcal{B}$
c	coin c
CI_c	coin identifier of c
$\mathcal{CR}(CI_c)$	coin registry of c
$\mathcal{NS}(CI_c)$	notary set of c
K	DHT keyspace
N	set of all the peer nodes in the DHT
R	set of all the resources stored in the DHT
C	set of all the coins in the DHT
$id_x(x)$	map of x to an identifier in K
$\rho(x)$	responsibility function mapping x to a set $\{PI\}$
q	number of coins every notary is responsible for
w	number of welcome coins granted to a newcomer

coin signed by the CGEN itself. $\mathcal{A}$ can collect CGEN's coins by sending these messages to the DHT. The signature of the CGEN prevents a malicious user from modifying the PAY messages and steal the welcome coins by changing the beneficiary.
Once $\mathcal{A}$ has successfully received its coins, it can join the P2P system and actively participate to any application or service offered by the P2P network and use PRICE for transactions accordingly.

P2P substrate join

On reception of the certificate, $\mathcal{A}$ joins the P2P substrate, and contacts other peers to advertise its presence and populate its routing table following usual P2P protocols. It also finds out the identities of the notaries corresponding to each of its coins using a map function ρ which, as an input of the identity of the coin c, outputs the set of peer identities responsible for its management, that are, the notaries. Upon reception of the initial coin, a caretaker notary, adds in its current coin registry the following information:

- the coin identifier,
- the signature of the CGEN,
- $\mathcal{A}$'s certificate,
- $\mathcal{A}$'s peer identifier which further will be replaced by the previous owner of the coin at each transaction of this coin,
- a serial number which is used to synchronize notaries and prevent replay attacks,
- a group signature generated by a subset of dedicated notaries.

D.4.3 Payment order

In order for $\mathcal{A}$ to transfer a coin c to $\mathcal{B}$, $\mathcal{A}$ has to indicate to the P2P system the new owner of the actual coin. This action takes place in two steps: 1) notary lookup, 2) payment request.

Notary lookup

In this step, $\mathcal{A}$ performs a lookup in the DHT using the coin identifier CI_c as a lookup key. In O(log (n)) steps, $\mathcal{A}$ reaches a node $\mathcal{P}$ in the notary set of the coin. $\mathcal{P}$ will act as the

caretaker of the transaction $\mathcal{A}$ is going to make.

Payment request

In order for $\mathcal{A}$ to grant a coin c to $\mathcal{B}$, $\mathcal{A}$ sends a signed payment message PAY to $\mathcal{P}$ containing the signed coin identifier CI_c proving c is a valid coin, the certificate of the new owner $\mathcal{B}$ proving $\mathcal{B}$ is a valid node in the system, $\mathcal{B}$'s IP address, and a serial number SN used to avoid replay attacks. This message is encrypted with $\mathcal{P}$'s public key to prevent eavesdropper from tracing the transaction. In case $\mathcal{A}$ is not the current owner of the coin or there is a mismatch between the serial number in the PAY message and that one in the coin registry, $\mathcal{P}$ simply discards the message, otherwise $\mathcal{P}$ forwards it to its neighborhood in the notary set $\mathcal{NS}(CI_c)$.
Please note that $\mathcal{P}$ is responsible for more than a single coin in the system and can receive several PAY messages for several coins from different users at the same time.

D.4.4 Payment notification

In order for the payment to succeed, a predefined quorum among the notaries of c has to agree on the update (or creation) of the entry associated to c in the coin registry performed by the caretaker $\mathcal{P}$. Once this agreement is met, the caretaker can notify the payer, the beneficiary and the notaries about the success of the transaction. These actions take place in two steps: 1) coin registry update, 2) payment confirmation.

Coin registry update

Every node in the DHT stores a coin registry $\mathcal{CR}$ keeping the association between every coin identifier it is responsible for and the peer identifier of the current owner of that coin. An entry in the coin registry has the form:

$$\mathcal{CR}(CI_c) = \left\{CIc, PI_{\mathcal{Z}}, Cert(PI_{\mathcal{A}}, K^+_{\mathcal{A}}), SN\right\}_{S_{\mathcal{NS}(CI_c)}}$$

where $Cert(PI_{\mathcal{A}}, K^+_{\mathcal{A}})$ identifies the current owner of the coin and is used to verify the integrity of PAY messages, $PI_{\mathcal{Z}}$ is the peer identifier of the previous owner of c, SN is a serial number used to refresh the coin registry of the nodes coming back online in the DHT and to avoid replay attacks, and $S_{\mathcal{NS}(CI_c)}$ is the group signature generated by a sufficient number of nodes in the notary set.

When a notary $\mathcal{N}_j$ receives a forwarded PAY message from $\mathcal{P}$, $\mathcal{N}_j$ checks the integrity of PAY, and computes on a temporary updated version of $\mathcal{CR}(CI_c)$ its own share $Share_j$ to be sent back to $\mathcal{P}$ through an SHR message. If a predefined quorum among a representative group of $\mathcal{NS}(CI_c)$ is reached, $\mathcal{P}$ computes the group signature $S_{\mathcal{NS}(CI_c)}$, updates $\mathcal{CR}(CI_c)$ and advertises it along the notary set.

Payment confirmation

In case the group signature $S_{\mathcal{NS}(CI_c)}$ is generated, the transaction succeeds and $\mathcal{P}$ sends back both $\mathcal{A}$ and $\mathcal{B}$ a notification message NTF containing the updated coin registry entry $\mathcal{CR}(CI_c)$. The group signature in this entry proofs the correctness of the transaction and prevents a malicious notary from arbitrarily modifying its content.

D.5 Evaluation

In this section we evaluate the feasibility of PRICE with respect to the security and privacy challenges defined in section D.2.

We assume the DHT as follows:

$$DHT = \langle K, N, R, C, id_n(\cdot), id_r(\cdot), id_c(\cdot), \rho(\cdot) \rangle$$

K is the DHT keyspace, N, R and C correspond to the set of nodes, the set of resources and the set of coins, respectively. $id_n : N \to K$, $id_r : R \to K$ and $id_c : C \to K$, denote the functions respectively associating a node, a resource, a coin to their identifier. Finally, as previously defined $\rho : K \to \{N\}$ denotes the mapping function which outputs the set peers responsible given a resource. In particular, this responsibility function determines the notary set of a coin: $\rho : id_c \to \{\mathcal{NS}(CI_c)\}$. We will call ***k-bit zone*** the subset of the id space containing all the peers whose id agrees in the high order k bits.

D.5.1 Security

Coin integrity/unforgeability The integrity or unforgeability of a coin c is guaranteed thanks to the signature S_{CGEN} of the Coin GENerator authority that generated that coin. Such a signature cannot be computed by anybody else, as the private key of the CGEN is never disclosed.

Transaction integrity The integrity of a transaction involving a coin c is represented by the integrity of the record $\mathcal{CR}(c)$ in the coin registry, and is guaranteed by the group signature $S_{\mathcal{NS}(CI_c)}$. Therefore PRICE prevents the double spending of coins. Computing such a signature requires the collusion of a sufficient number of notaries and can therefore be mitigated by increasing the minimum notaries quorum at the expense of higher computation cost and communication overload.
Moreover, to keep fresh versions of the coin registry, a serial number in every entry of the coin registry helps a notary to come back online to update his registry from the other notaries.

Identifiers integrity In PRICE, peers receive their peer identifier PI from the TIS as an output of a one-way function $h_{MK}(\cdot)$ over their real identity ID. Since the secret MK used in the keyed hash function $h_{MK}(\cdot)$ is known by the TIS only, identifiers cannot be arbitrarily computed or guessed by any user. Moreover, the account creation procedure can be repeated several times but the result always leads to the same identifier. Therefore, even though certificates can be re-issued, peer identifiers never change. This prevents any malicious user from stealing a legitimate user's identity, or from creating different identities, namely Sybils, and launch Denial of Service attacks.

D.5.2 Privacy

Data confidentiality In PRICE, PAY and NTF messages are encrypted with the recipient's public key found in its certificate signed by the TIS. The user's private and public keys are computed by the user himself at the act of the account creation, and the private key is never disclosed. In case the private key is stolen, the certificate can be re-issued.

Anonymity As the TIS and the CGEN are separate entities, nobody can link a coin identifier to a real user's identity. In fact, the TIS is the only party being able to link a peer identifier to a real identity, but it does not hold any information about that user's coins. On the other hand, the CGEN distributes welcome coins to new peers, but it does not manage identity information. In case the TIS and the CGEN services are merged, no information rather than the *initial* association between coins and users can be derived. In fact, both the TIS and the CGEN are off-line services contacted only once by each legitimate user and do not play any role neither in communication nor in data management. Perhaps, they can be built in a distributed fashion.
In the DHT, a caretaker $\mathcal{P}$ can link a coin identifier CI_c to the owner's peer identifier $PI_{\mathcal{A}}$

for all the coins $\mathcal{P}$ is responsible for. Anyway, this does not reveal the owner $\mathcal{A}$'s real identity to $\mathcal{P}$, as no information about $PI_{\mathcal{A}}$ can be retrieved from $\mathcal{A}$'s certificate $Cert(PI_{\mathcal{A}}, K^{+}_{\mathcal{A}})_{S_{TIS}}$.

Transaction untraceability In PRICE, the number of coins held by a user $\mathcal{A}$ and the history of all the transactions $\mathcal{A}$ did in the system is known by $\mathcal{A}$ and no one else. A single coin transaction can be traced by the notary set of that coin. However, this does not reveal anything about the other transactions of the same actor $\mathcal{A}$. Moreover, due to the security of the pseudo-random function used by the CGEN to generate a coin c, the association mapped by $\rho(\cdot)$ between the coin registry entry $\mathcal{CR}(CI_c)$ and a notary $\mathcal{N}_j \in \mathcal{NS}(CI_c)$ responsible for it is also random.

D.5.3 Performance

In this section, we provide and evaluation of the performance of PRICE in terms of latency, storage and bandwidth consumption. In the following, we will consider Kad [86] as the underlying P2P overlay.

Latency The total transaction time T for a coin c can be seen as the sum of the time T_L required to the payer $\mathcal{A}$ for looking up for a coin identifier, the time T_R for transferring the PAY message, the time T_F required for the caretaker $\mathcal{P}$ to forward PAY along the notary set, the time T_S to collect the shares in order to compute the group signature and, finally, the time T_C required for confirming the payment:

$$T = T_L + T_R + T_F + T_S + T_C \tag{D.1}$$

Considering T_R as negligible, we can define T as the sum of the time required for a successful lookup T_L and three one-hop Round Trip Time T_{RTT} in the DHT.

T_{RTT} and T_L are defined as random variables and are set to the values originated from real measurements on Kad conducted in [110]. T is then evaluated with Monte Carlo techniques based on these measurements. A set of 10000 samples t_i is computed as follows: we generate 4 uniform random variates between 0 and 1, namely y_l, y_{r1}, y_{r2}, y_{r3} and sum the inverse $F^{-1}_{T_{RTT}}$ and $F^{-1}_{T_{RTT}}$ of the cumulative distributions $F_{T_{RTT}}$, F_{T_L} at those points.

The results are shown in figure D.2, and table D.2 summarizes the main statistics. As one can see, even if 50% of transactions require less than 7.5 seconds, still 10% of them succeed in more than 12.3 seconds. Decreasing the significant contribution of T_L by the use

of central indexing services may speed up the transaction time at the expense of a lower privacy protection.

Table D.2: Time statistics in seconds for the three main distributions in figure D.2

	Average	50th percentile	90th percentile
T_{RTT}	0.664	0.287	1.50
T_L	6.51	5.64	8.87
T	8.47	7.48	12.3

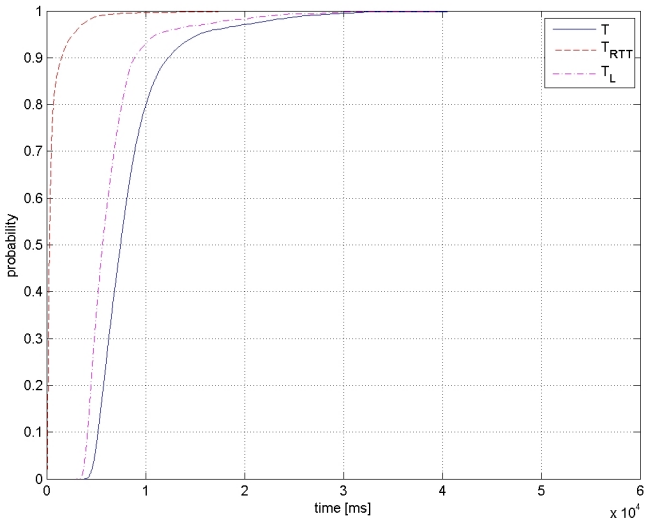

Figure D.2: Total transaction time evaluation: T_{RTT} and T_L from [110], T from Monte Carlo techniques (10000 samples).

Storage overhead An entry in the coin registry contains the coin id, the signature of the CGEN stating this coin is valid, the current owner's certificate, the old user's peer identifier, a serial number, and finally the notary set group signature validating the correct association between the coin and its current owner. Assuming a keyspace of 128 bits, a signature length of 512 bits, public keys of 512 bits and a 32 bits integers length, an entry requires 308 Bytes. The number of coins q every peer is responsible for, and as a consequence the size required to store the coin registry, strongly depends on the number of peers and the number of coins in a notary set. Assume the id space is divided into 2^k zones, and in each of them peers

and resources agree on the k high-order bits. Assume the responsibility function $\rho(CI_c, k)$ maps a coin c to the set of peers in the k-bit zone defined by k. In this case:

$$q = \frac{\|N\|}{2^k} w \tag{D.2}$$

where $\|N\|$ is the cardinality of the set N, i.e. the total number of peers in the system, and w is the number of welcome coins every peer receives at the very first join. Table D.3 shows the size every peer should allocate, on average, to store its coin registry in a network of 5.12 millions peers[1] and where each node initially receives 100 welcome coins. When k is set to 8, then 20,000 peers populate a zone, and can act as notaries for a maximum of 2 millions of coins. Their coin registry can then reach a maximum size of 587 MB. By increasing k to 16, the number of coins every peer is responsible for decreases to 7800, leading the size of a coin registry to 2.29 MB.

Table D.3: Coin registry size in MB for different values of k

k [bit]	8	10	12	14	16
$\mathcal{CR}$ [MB]	587.46	146.87	36.72	9.18	2.29

Communication Bandwidth Overhead In order to evaluate the communication overhead, we first evaluate the minimum number of peers t required to compute a group signature. This threshold number should be defined according to the underlying privacy and robustness challenges: t strongly depends on the ratio m of malicious users and the online probability p of nodes. t can therefore be computed as follows:

$$t = \frac{\|N\|}{2^k} * p * m + 1 \tag{D.3}$$

Figure D.3 outputs the t values with respect to different m and p values where k is set to 16. t varies between 2 and 21 where both m and p take values between 0.1 and 0.5.

Each of these t notaries receives the PAY message forwarded by the caretaker $\mathcal{P}$, furhter computes the share of the group signature and sends it back to $\mathcal{P}$. Assuming a keyspace of 128 bits, a signature length of 512 bits, public keys of 512 bits and a 32 bits integers length, a PAY message requires 246 Bytes, while s' size is 64 Bytes.

[1] Steiner et al.[111] observed between 12 and 20 thousand active peers in one 256-th of the entire KAD id space.

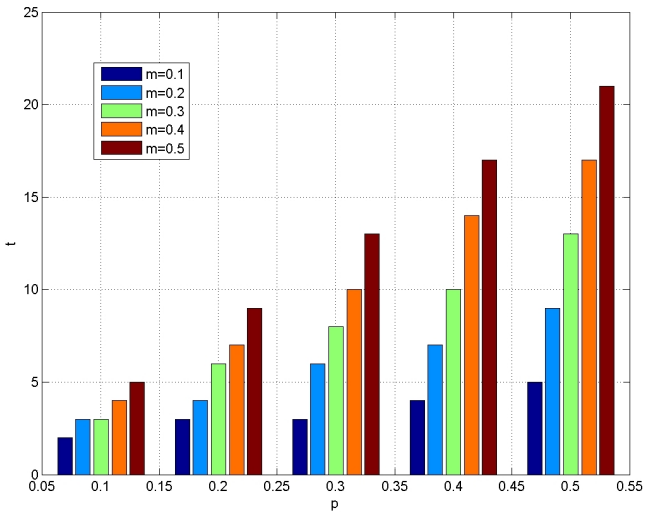

Figure D.3: Evaluation of the number of notaries t to be contacted for every transaction for different online- p and misbehaving- m probabilities.

Once computed the whole group signature $S_{\mathcal{NS}(CI_c)}$, $\mathcal{P}$ sends a NTF message containing the coin registry entry $\mathcal{CR}(CI_c)$, whose size, according to the previous assumptions, is 308 Bytes. Assuming transactions occur every hour with a frequency λ, figure D.4 shows that the bandwidth consumption is slightly less than 7Kbps when 100 transactions occurs every hour and 50 notaries have to agree on them.

D.6 Related Work

A huge literature proposed credit-based mechanisms to stimulate cooperation in networks with the presence of selfish nodes[2].

In MANETS, credit-based incentive mechanisms were designed for enforcing the cooperation among nodes for the specific operation of *forwarding*. To achieve fairness, [43] was relying on tamper proof hardware whereas [127] defined a centralized on-line trusted entity.

PRICE does not focus on the nature of a specific operation and does not require any tamper proof hardware nor centralized entities to manage credits.

In the P2P scenario, authors in [117] proposed a micropayment scheme where each peer is associated to a scalar value called *Karma*. A set of randomly chosen *bank set* nodes increase

[2]i.e. nodes trying to maximize the benefits they get from the network while minimizing their contribution to it.

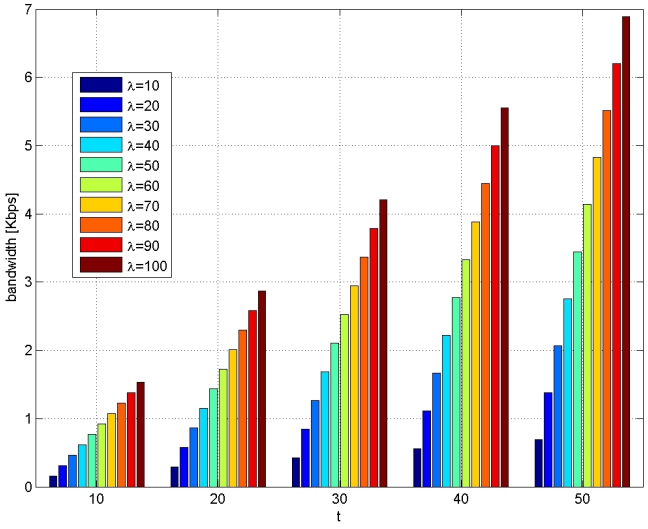

Figure D.4: Evaluation of the bandwidth consumption for different transaction rates λ (per hour) and notaries to be contacted t.

or decrease a peer's karma in case this peer contributes with- or consumes- resources. An atomic transaction scheme ensures fairness in the payment since the key to decrypt resources and certificate of receipt are provided simultaneously to the resource consumer and the provider respectively. When a file transfer occurs from a peer B to a peer A such a file is encrypted with a secret DES, then upon A's authorization, each member of A's bank set independently send a karma transfer request to all members of B's bank set, that in turn ask again A's bank set nodes for an acknowledgment. Once verified a majority quorum exists, B proceeds with the file transfer, and A provides B with a receipt. If B gets the receipt, A receives the key to decrypt the file.

In BitTorrent[3], a variant of "'tit-for-tat"' [50] mechanism encourages fairness in the exchange of file chunks. Such a mechanism aims at seeking pareto efficiency, meaning in this case that peers reciprocate uploading to peers which upload to them, aiming at having all the time several connections actively transferring data in both directions. In case of lack of reciprocity, a peer can temporarily refuse to upload a chunk to- , or *choke* a-, lazy peer. An optimistic unchoke mechanism, corresponding to always cooperating on the first move in prisoner's dilemma, solves the problem of discovering if current unused connections are better than the ones being used.

[3]http://www.bittorrent.com

Criticisms against the incentive mechanisms in BitTorrent assert that its effectiveness is largely due to the altruistic behavior of a small number of altruistic nodes [97] and solutions like in [113] have been proposed to improve the overall system performance.

In Swift [113], peers exchanging file chunks are denoted as *traders* and employ a default trading strategy that is either good for them and for the network itself. Free riders are the most penalized in case of insufficient upload capacity to satisfy demand. Authors consider three strategies for traders and classify them accordingly: *paranoid*, *one-time risk-taking*, and *perioding risk-taking*. Paranoid traders are reciprocative players waiting for the reception of a valid chunk before offering to send an equal amount back, one-time risk-takers can offer free chunks to a peer never encountered before to encourage trading with the chance of receiving nothing in return, while periodic risk-takers give out free chunks periodically. Authors show that peers taking risks receive the most benefit in return, and deviating from the proposed default strategy of periodic risk-taking provides little or no advantage. Swift has then been added to the official BitTorrent client and named as *TradeTorrent*[4].

Finally, authors in [37] drew inspiration from BitTorrent to propose a P2P content distribution system based on endorsed e-cash [45] to provide accountability while preserving privacy in P2P systems. In such an approach, users can exchange files if they know the correct hashes on those files. In endorsed e-cash, users withdraw e-coins from a central bank maintaining all participants'accounts and spend them for digital content with a fair exchange protocol. In case a user gets paid, he must deposit e-coins in the bank before spending them again. A Trusted Third Party (TTP), namely the *arbiter*, is responsible for resolving disputes. Authors modify the endorsed e-cash protocol in [45] to allow the arbiter for resolving conflicts by examining a much shorter amount of data. Sybyl node creation is discouraged thanks to a mechanism in which newcomers are invited by friends and receive an initial credit from them.

PRICE extends the security an privacy features offered in [117, 113, 37] by revisiting the concept of bank account. As a main novelty of PRICE, in fact, no entity in the system can derive the total amount of credit a node currently holds, as accounts are made available for coins rather than for users. As an important consequence, there is no entity an attacker can target to discover one or more victim's account, and derive, for instance, its participation in the network.

[4] http://mnl.cs.stonybrook.edu/project/tradetorrent/

D.7 Conclusion

PRICE is a new cooperation enforcement mechanism which relies on credit-based incentives and takes advantage of the underlying DHT based P2P network to cope with security and privacy challenges. The task of coin management is distributed among several peers and in order to ensure transaction untraceability, PRICE assigns each single coin to a different set of notaries. The assignment function on the inherent functionality of a P2P network which is the DHT and the randomness of each assignment is ensured thanks to the security of the pseudo-random function used to generate the coin. The number of notaries is defined based on the ratio of malicious nodes and the average online probability and can have a direct impact on the robustness and performance of the P2P network. The communication overhead increases when more notaries are solicited for computing the threshold signature.

Bibliography

[1] '419' scam - advance fee / fake lottery scam. http://www.419scam.org/.

[2] AllMyData Tahoe. http://allmydata.org.

[3] Average facebook user has 130 friends. http://breakingnewsworld.net/2011/11/average-facebook-user-has-130-friends/.

[4] Facebook fan pages need security upgrade, says victim. http://gadgetwise.blogs.nytimes.com/2010/03/18/fake-facebook-fan-pages/.

[5] Facebook hack reveals trend in targeting social networks. http://fraudwar.blogspot.com/2009/05/facebook-hack-reveals-trend-in.html.

[6] Facebook responds to massive phishing scheme. http://scitech.blogs.cnn.com/2010/03/19/facebook-responds-to-massive-phishing-scheme/.

[7] Facebook statement of rights and responsibilities. http://www.facebook.com/legal/terms.

[8] Facebook users at risk of 'rubber duck' identity attack. http://www.sophos.com/pressoffice/news/articles/2009/12/facebook.html.

[9] Facebook users targeted in massive spam run. http://www.pcworld.com/businesscenter/article/191847/facebook_users_targeted_in_massive_spam_run.html.

[10] Facebook users unwittingly spread koobface worm. http://content.usatoday.com/communities/technologylive/post/2009/12/koobface-compels-facebook-victims-to-help-spread-worm-/1.

[11] Gnu general public license. http://www.gnu.org/licenses/gpl-3.0.html.

[12] Goldman offering clients a chance to invest in facebook. http://dealbook.nytimes.com/2011/01/02/goldman-invests-in-facebook-at-50-billion-valuation/.

[13] Google announces fourth quarter and fiscal year 2011 results. http://investor.google.com/earnings/2011/Q4_google_earnings.html.

[14] Google+ policy. http://www.google.com/intl/en/+/policy/index.html.

[15] Google privacy policy. http://www.google.com/intl/en/policies/privacy/.

[16] Linkedin about us. http://press.linkedin.com/about.

[17] Linkedin user agreement. http://www.linkedin.com/static?key=user_agreement.

[18] Linkedin's $8b ipo - silicon valley, get ready for housing recovery. http://venturebeat.com/2011/05/19/linkedins-8b-ipo-silicon-valley-get-ready-for-housing-recovery/.

[19] Man bust for facebook insults. http://mybroadband.co.za/news/Internet/6580.html.

[20] Myspace censors user-generated content. http://www.civic.moveon.org/pdf/myspace/.

[21] Myspace user data for sale. http://www.pcworld.com/article/191716/myspace_user_data_for_sale.html.

[22] Nestle's facebook page: How a company can really screw up social media. http://blogs.bnet.com/businesstips/?p=6786.

[23] Rachel uchitel threatens lawsuit over facebook 'defamation'. http://www.thaindian.com/newsportal/sports/rachel-uchitel-threatens-lawsuit-over-facebook-defamation_100337445.html.

[24] Safebook prototype. http://www.safebook.eu/home.php?content=prototype.

[25] Social network ad revenues to reach $10 billion worldwide in 2013. http://www.emarketer.com/Article.aspx?R=1008625.

[26] Twitter blog: Monday morning madness. http://blog.twitter.com/2009/01/monday-morning-madness.html.

[27] Twitter is now valued at $8 billion. http://tech2.in.com/news/social-networking/twitter-is-now-valued-at-8-billion/250542.

[28] Twitter terms of service. https://twitter.com/tos?lang=en.

[29] Twitter warns of new phishing attack. http://www.pcworld.com/businesscenter/article/174607/twitter_warns_of_new_phishing_attack.html.

[30] Wuala. http://www.wua.la/en/home.html.

[31] Yahoo finance: statistics on google inc. (goog). http://finance.yahoo.com/q/ks?s=GOOG.

[32] Eytan Adar and Bernardo A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), October 2000.

[33] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[34] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 181–190, Banff, Alberta, Canada, 2007. ACM.

[35] Randolph Baden, Adam Bender, Daniel Starin, Neil Spring, and Bobby Bhattacharjee. Persona: An online social network with user-defined privacy. In *ACM SIGCOMM*, Barcelona, Spain, August 2009.

[36] Marco Balduzzi, Christian Platzer, Thorsten Holz, Engin Kirda, Davide Balzarotti, and Christopher Kruegel. Abusing Social Networks for Automated User Profiling. Research Report RR-10-233, EURECOM, 2010.

[37] Mira Belenkiy, Melissa Chase, C. Chris Erway, John Jannotti, Alptekin Küpçü, Anna Lysyanskaya, and Eric Rachlin. Making p2p accountable without losing privacy. In

Proceedings of the 2007 ACM workshop on Privacy in electronic society, WPES '07, pages 31–40, Alexandria, Virginia, USA, 2007. ACM.

[38] Ranjita Bhagwan, Kiran Tati, Yu-Chung Cheng, Stefan Savage, and Geoffrey M. Voelker. Total recall: system support for automated availability management. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI '04, pages 25–25, San Francisco, California, 2004. USENIX Association.

[39] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 551–560, Madrid, Spain, 2009. ACM.

[40] Joseph Bonneau and Sören Preibusch. The Privacy Jungle: On the Market for Privacy in Social Networks. *Proceedings of the Eighth Workshop on the Economics of Information Security*, June 2009.

[41] S. Buchegger and J-Y. Le Boudec. Nodes bearing grudges: Towards routing security, fairness and robustness in mobile ad hoc networks. In *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, PDP 2002, Canary Islands, Spain, 2002.

[42] Sonja Buchegger, Doris Schiöberg, Le Hung Vu, and Anwitaman Datta. PeerSoN: P2P Social Networking Early Experiences and Insights. In *Proceedings of the Second ACM Workshop on Social Network Systems 2009, colocated with Eurosys 2009*, SNS '09, Nürnberg, Germany, March 2009.

[43] Levente Buttyán and Jean-Pierre Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '00, pages 87–96, Boston, Massachusetts, 2000. IEEE Press.

[44] Levente Buttyán and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *Mobile Networks and Applications*, 8(5):579–592, October 2003.

[45] J. Camenisch, A. Lysyanskaya, and M. Meyerovich. Endorsed e-cash. In *IEEE Symposium on Security and Privacy*, SP '07, pages 101 –115, May 2007.

[46] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8), 2002.

[47] David Chaum. Blind signatures for untraceable payments. In Ronald Linn Rivest, A. Sherman, and D. Chaum, editors, *Advances in Cryptology*, CRYPTO '82, pages 199–203. Plenum Press, 1983.

[48] David Chaum. Blind signature system. In D. Chaum, editor, *Advances in Cryptology*, CRYPTO '83, page 153, New York, 1984. Plenum Press.

[49] Tom Chothia and Konstantinos Chatzikokolakis. A survey of anonymous peer-to-peer file-sharing. In *Proceedings of the 2005 international conference on Embedded and Ubiquitous Computing*, EUC '05, pages 744–755, Nagasaki, Japan, 2005. Springer-Verlag.

[50] Bram Cohen. Incentives Build Robustness in BitTorrent. In *1st Workshop on Economics of Peer-to-Peer Systems*, P2PECON '03, Berkeley, CA, USA, June 2003.

[51] comScore. It's a social world: Top 10 need-to-knows about social networking and where it's headed, December 2011. `http://www.comscore.com/Press_Events/Press_Releases/2011/12/Social_Networking_Leads_as_Top_Online_Activity_Globally`.

[52] Leucio Antonio Cutillo, Mark Manulis, and Thorsten Strufe. *Security and Privacy in Online Social Networks*. Springer, 2010.

[53] Leucio Antonio Cutillo, Refik Molva, and Melek Önen. Performance and privacy tradeoff in peer-to-peer on-line social networks. Technical Report RR10244, Eurecom, July 2010.

[54] Leucio Antonio Cutillo, Refik Molva, and Melek Önen. Analysis of privacy in online social networks from the graph theory perspective. In *GLOBECOM 2011, Selected Areas in Communications Symposium, Social Networks Track*, Houston, Texas, USA, December 2011.

[55] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Leveraging social links for trust and privacy in networks. In *INetSec 2009, Open Research Problems in Network Security*, Zurich, Switzerland, April 2009.

[56] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Privacy preserving social networking through decentralization. In *WONS 2009, 6th International Conference on Wireless On-demand Network Systems and Services*, Snowbird, Utah, USA, February 2009.

[57] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Safebook: a privacy preserving online social network leveraging on real-life trust. *IEEE Communications Magazine, Consumer Communications and Networking Series*, 47(12), December 2009.

[58] d. m. boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1), 2007.

[59] danah m. boyd. Facebook's privacy trainwreck. *Convergence: The International Journal of Research into New Media Technologies*, 14(1):13 – 20, 2008.

[60] Matteo Dell'Amico and Yves Roudier. A measurement of mixing time in social networks. In *STM 2009, 5th International Workshop on Security and Trust Management*, Saint Malo, France, September 2009.

[61] P. Dewan and P. Dasgupta. P2p reputation management using distributed identities and decentralized recommendation chains. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):1000 –1013, July 2010.

[62] John Douceur. The Sybil Attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, volume 2429 of *IPTPS '02*, pages 251–260, Cambridge, MA, USA, March 2002. Springer.

[63] Peter Druschel and Antony Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, HOTOS '01, pages 75–, Schoss Elmau, Germany, 2001. IEEE Computer Society.

[64] Dinei Florencio and Cormac Herley. A Large-Scale Study of Web Password Habits.

In *16th International Conference on World Wide Web (WWW 2007)*, pages 657–666. ACM, 2007.

[65] Borko Furht, editor. *Handbook of Social Network Technologies and Applications*. Springer, 2010. ISBN: 978-1-4419-7141-8.

[66] Cong Geng and Xudong Jiang. Face recognition using sift features. In *Proceedings of the 16th IEEE international conference on Image processing*, ICIP'09, pages 3277–3280, Cairo, Egypt, 2009. IEEE Press.

[67] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, STOC '85, pages 291–304, Providence, Rhode Island, USA, 1985. ACM.

[68] Kalman Graffi, Christian Groß, Dominik Stingl, Daniel Hartung, Aleksandra Kovacevic, and Ralf Steinmetz. Lifesocial.kom: A secure and p2p-based solution for online social networks. In *Proceedings of the IEEE Consumer Communications and Networking Conference*, CCNC 2011. IEEE Computer Society Press, January 2011.

[69] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, WPES '05, pages 71–80, Alexandria, VA, USA, 2005. ACM.

[70] Saikat Guha, Neil Daswani, and Ravi Jain. An experimental study of the skype peer-to-peer voip system. In *Proceedings of The 5th International Workshop on Peer-to-Peer Systems*, IPTPS '06, pages 1–6, Santa Barbara, CA, February 2006.

[71] Andreas Haeberlen, Alan Mislove, and Peter Druschel. Glacier: highly durable, decentralized storage despite massive correlated failures. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 143–158, Boston, MA, USA, 2005. USENIX Association.

[72] Neil Haller. The s/key one-time password system. In *In Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 151–157, San Diego, CA, USA, 1994.

[73] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, October 2007.

[74] Harvey Jones and José H. Soltren. Facebook: Threats to Privacy. *Project MAC: MIT Project on Mathematics and Computing*, December 2005.

[75] Lalana Kagal, Chris Hanson, and Daniel Weitzner. Using dependency tracking to provide explanations for policy management. pages 54–61, 2008.

[76] Dimitris N. Kalofonos, Zoe Antoniou, Franklin D. Reynolds, Max Van-Kleek, Jacob Strauss, and Paul Wisner. Mynet: A platform for secure p2p personal and social networking services. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications*, PERCOM '08, pages 135–146, Hong Kong, China, March 2008. IEEE Computer Society.

[77] Nicolas Kourtellis, Joshua Finnis, Paul Anderson, Jeremy Blackburn, Cristian Borcea, and Adriana Iamnitchi. Prometheus: user-controlled p2p social data management for socially-aware applications. In *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware*, Middleware '10, pages 212–231, Bangalore, India, 2010. Springer-Verlag.

[78] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, pages 7–12, Barcelona, Spain, 2009. ACM.

[79] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24:770–772, November 1981.

[80] Lucas Laursen. Fake facebook pages spin web of deceit. 458(7242):1089, 2009.

[81] Christof Leng, Wesley W. Terpstra, Bettina Kemme, Wilhelm Stannat, and Alejandro P. Buchmann. Maintaining replicas in unstructured p2p systems. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 19:1–19:12, Madrid, Spain, 2008. ACM.

[82] Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. Free riding in BitTorrent is cheap. In *Fifth Workshop on Hot Topics in Networks*, HotNets-V, Irvine, CA, US, nov 2006.

[83] László Lovász. Random walks on graphs: A survey. In D. Miklos, V. T. Sos, and

T. Szonyi, editors, *Combinatorics, Paul Erdös is Eighty*, volume 2, pages 353–398, Budapest, 1993. János Bolyai Mathematical Society.

[84] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[85] A. Majumdar and R. K. Ward. Discriminative SIFT Features for Face Recognition. In *Canadian Conference on Electrical and Computer Engineering*, pages 27–30, May 2009.

[86] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, Cambridge, MA, USA, 2002. Springer-Verlag.

[87] Anh-Minh Ngyuen Mehdi Mani and Noel Crespi. What's up: P2p spontaneous social networking. In *Proceedings of PERCOM 2009, IEEE International Conference on Pervasive Computing and Communications*, Galveston Tx, USA, March 2009.

[88] P. Michiardi and R. Molva. CORE: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of IFIP Communication and Multimedia Security Conference*, CMS 2002, Portoroz, SLOVENIA, 2002.

[89] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 251–260, New York, NY, USA, 2010. ACM.

[90] Michael Mitzenmacher and Eli Upfal. *Probability and Computing : Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, January 2005.

[91] Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 383–389, Melbourne, Australia, 2010. ACM.

[92] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 173–187, Oakland, California, USA, May 2009. IEEE Computer Society.

[93] Jaehong Park and Ravi Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the seventh ACM Symposium on Access Control Models and Technologies*, SACMAT '02, pages 57–64, Monterey, California, USA, 2002. ACM.

[94] Thomas Paul, Sonja Buchegger, and Thorsten Strufe. Decentralizing social networking services. In *International Tyrrhenian Workshop on Digital Communications*, ITWDC 2010, pages 1–10, Island of Ponza, Italy, September 2010.

[95] Thomas Paul, Sonja Buchegger, and Thorsten Strufe. Decentralized social networking services. In Luca Salgarelli, Giuseppe Bianchi, and Nicola Blefari-Melazzi, editors, *Trustworthy Internet*, pages 187–199. Springer Milan, 2011.

[96] P. Jonathon Phillips, W. Todd Scruggs, Alice J. O'Toole, Patrick J. Flynn, Kevin W. Bowyer, Cathy L. Schott, and Matthew Sharpe. Frvt 2006 and ice 2006 large-scale experimental results. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):831–846, May 2010.

[97] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bit torrent. In *Proceedings of the 4th USENIX conference on Networked Systems Design and Implementation*, NSDI '07, page 1, Cambridge, MA, USA, 2007. USENIX Association.

[98] Johan Pouwelse, Pawel Garbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In Miguel Castro and Robbert van Renesse, editors, *Peer-to-Peer Systems IV*, volume 3640 of *Lecture Notes in Computer Science*, pages 205–216. Springer Berlin / Heidelberg, 2005.

[99] David Recordon and Drummond Reed. Openid 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, DIM '06, pages 11–16, Alexandria, Virginia, USA, 2006. ACM.

[100] I.S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2), 1960.

[101] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[102] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. Opendht: a public dht service and its uses. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '05, pages 73–84, Philadelphia, Pennsylvania, USA, 2005. ACM.

[103] M Rogers and S Bhatti. How to Disappear Completely: A Survey of Private Peer-to-Peer Networks. In *Proc. International Workshop on Sustaining Privacy in Autonomous Collaborative Environments*, SPACE 2007, Moncton, New Brunswick, Canada, 2007.

[104] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In Rachid Guerraoui, editor, *Middleware 2001*, volume 2218 of *Lecture Notes in Computer Science*, pages 329–350. Springer Berlin Heidelberg, 2001.

[105] A. Satsiou and L. Tassiulas. Reputation-based resource allocation in p2p systems of rational users. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):466 –479, April 2010.

[106] Fabian Schneider, Anja Feldmann, Balachander Krishnamurthy, and Walter Willinger. Understanding Online Social Network Usage from a Network Perspective. In *ACM SIGCOMM conference on Internet measurement*, 2009.

[107] Amre Shakimov, Harold Lim, Ramón Cáceres, Landon Cox, Kevin Li, Dongtao Liu, and Alexander Varshavsky. Vis-à-vis: Privacy-preserving online social networking via virtual individual servers. In *Third International Conference on Communication Systems and Networks*, COMSNETS 2011, Bangalore, India, 2011.

[108] Jeffrey Shneidman and David Parkes. Rationality and self-interest in peer to peer networks. In M. Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II*, volume 2735 of *Lecture Notes in Computer Science*, pages 139–148. Springer Berlin / Heidelberg, 2003.

[109] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *Proceedings of the 25th IEEE International Conference on Computer Communications*, INFOCOM 2006, pages 1–12, April 2006.

[110] Moritz Steiner, Damiano Carra, and Ernst W. Biersack. Faster content access in kad. In *Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing*, P2P '08, pages 195–204, Aachen, GERMANY, September 2008. IEEE Computer Society.

[111] Moritz Steiner, Taoufik En Najjary, and Ernst W Biersack. A global view of KAD. In *ACM SIGCOMM Internet Measurement Conference*, IMC 2007, San Diego, USA, October 2007.

[112] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 189–202, Rio de Janeriro, Brazil, 2006. ACM.

[113] Karthik Tamilman, Vinay Pai, and Alexander Mohr. Swift: A system with incentives for trading. In *Proceedings of Second Workshop of Economics in Peer-to-Peer Systems*, June 2004.

[114] Amin Tootoonchian, Stefan Saroiu, Yashar Ganjali, and Alec Wolman. Lockr: better privacy for social networks. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 169–180, Rome, Italy, December 2009. ACM.

[115] Carmelo Velardo and Jean-Luc Dugelay. Face recognition with daisy descriptors. In *Proceedings of the 12th ACM workshop on Multimedia and security*, MM&Sec '10, pages 95–100, Roma, Italy, 2010. ACM.

[116] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[117] V. Vishnumurthy, S. Chandrakumar, and E. Sirer. KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. In *Workshop on the Economics of Peer-to-Peer Systems*, P2PEcon, Berkeley, CA, USA, 2003.

[118] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: Using Hard AI Problems for Security. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 294–311. Springer, 2003.

[119] Samuel D. Warren and Louis D. Brandeis. The right to privacy. *Harward Law Review*, 4(5):193–220, December 1890.

[120] Hakim Weatherspoon. *Design and evaluation of distributed wide-area on-line archival storage systems*. PhD thesis, Berkeley, CA, USA, 2006. AAI3254129.

[121] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A Practical Attack to De-Anonymize Social Network Users. In *IEEE Symposium on Security and Privacy*. IEEE CS, 2010. `http://www.iseclab.org/papers/sonda.pdf`.

[122] Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, January 2002.

[123] Zhongmei Yao, Derek Leonard, Xiaoming Wang, and Dmitri Loguinov. Modeling heterogeneous user churn and local resilience of unstructured p2p networks. In *Proceedings of the 14th IEEE International Conference on Network Protocols*, ICNP '06, pages 32–41, Santa Barbara, California, 2006. IEEE Computer Society.

[124] Ching Man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, and Tim Berners-Lee. Decentralization: The Future of Online Social Networking. In *W3C Workshop on the Future of Social Networking*, Barcelona, Spain, January 2009.

[125] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, December 2003.

[126] Elena Zheleva and Lise Getoor. To Join or Not to Join: The Illusion of Privacy in Social Networks with Mixed Public and Private User Profiles. In *WWW 2009*, pages 531–540. ACM, 2009.

[127] S. Zhong, J. Chen, and Y.R. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, INFOCOM '03, San Francisco, CA, USA, 2003.