Dipartimento di Informatica e
Scienze dell'Informazione

On the Privacy and Security of Internet Communications: an Attempt to Fix Some of the Current Conceptual and Practical Issues

by

Paolo Gasti

Theses Series **DISI-TH-2010-08**

DISI, Università di Genova
v. Dodecaneso 35, 16146 Genova, Italy `http://www.disi.unige.it/`

Università degli Studi di Genova

Dipartimento di Informatica e
Scienze dell'Informazione

Dottorato di Ricerca in Informatica

Ph.D. Thesis in Computer Science

On the Privacy and Security of Internet Communications: an Attempt to Fix Some of the Current Conceptual and Practical Issues

by

Paolo Gasti

February, 2010

Dottorato di Ricerca in Informatica
Dipartimento di Informatica e Scienze dell'Informazione
Università degli Studi di Genova

DISI, Univ. di Genova
via Dodecaneso 35
I-16146 Genova, Italy
http://www.disi.unige.it/

Ph.D. Thesis in Computer Science (S.S.D. INF/01)

Submitted by Paolo Gasti
Department of Computer Science (DISI)
University of Genoa
gasti@disi.unige.it

Date of submission: February 2010

Title: On the Privacy and Security of Internet Communications: an Attempt to Fix Some of the Current Conceptual and Practical Issues

Advisor: Prof. Giovanni Chiola
Department of Computer Science (DISI)
University of Genoa
chiola@disi.unige.it

Ext. Reviewers:

Prof. Giuseppe Ateniese
Information Security Institute
The Johns Hopkins University
ateniese@isi.jhu.edu

Prof. Luigi V. Mancini
Dipartimento di Informatica
Università di Roma "La Sapienza"
lv.mancini@di.uniroma1.it

Abstract

Today's technical and legal landscape present formidable challenges to privacy. The disclosure of private data has become commonplace due to carelessness, theft or legal actions. Increasing reliance on network services causes sensitive data to be cached, copied, and archived by third parties, often without users' knowledge or control. Cloud computing and ubiquitous computing are becoming an integral part of everyday life. The social consequences of this development are profound, and the technological challenges posed by these paradigms are complex and interesting. While the lack of privacy has, in principle, always characterized Internet communications, never before Internet communication has revealed so much about the personal tastes of its users. This is because never before Internet has contained so much personal user data.

This thesis is divided into four parts. The first part introduces different aspects of privacy that are dealt in this thesis, detailing the current privacy and anonymity landscape.

The second part of this dissertation focuses on theoretical aspects, introducing new cryptographic algorithms and providing a formal analysis of their security. More specifically it presents the first anonymous identity-based encryption scheme based on the quadratic residuosity problem, which can be used as an anonymous encryption tool or as the fundamental block for a keyword searchable encryption scheme. This thesis also introduce the first sender and receiver public key deniable encryption scheme, which allows communicating parties to deny the existence of an encrypted message even in presence of an adversary who can coerce them into giving up their secret information.

The third part of this thesis analyzes several network-related aspects of anonymity and privacy, and provides practical solutions. First it introduces a new pseudonymous authentication and authorization systems, which solves several problems related to digital certificates. Then it focuses on cloud storage presenting DenFS, the first shared deniable file system. Finally it illustrates an anonymous peer-to-peer file sharing system based on network coding, for which it provides efficient strategy for packet verification.

The last part concludes this work and discusses some of the open problems.

To my wife Miharu, for teaching me every day
what is really important in life

Every man should know that his conversations, his correspondence, and his personal life are private. (Lyndon B. Johnson)

Acknowledgements

I would like to thank my advisor, Professor Giovanni Chiola, for his help and support. His invaluable encouragement and trust allowed me to complete my Ph.D. and my research activity.

I would like to make a very special thanks to Professor Giuseppe Ateniese, who as been my mentor and my source of inspiration, but most of all a very good and truly supporting friend.

I am in great debt with Professor Marina Ribaudo for her help, support and friendship when I most needed it.

I would like to thank all the people I worked with. A special thank goes to Marina Blanton, Giuseppe Ciaccio, Yu Chen, Luigi V. Mancini and Vishwas Patil for their invaluable wisdom and endless patience.

I am grateful with the Fulbright Program, which provided me with a grant that allowed me to perform some of my research activity at the Johns Hopkins University.

I thank the entire computer science department at the Johns Hopkins University for supporting me and allowing me to perform my research in their laboratories.

I also want to thank the Ph.D. students and collaborators of the University of Genoa and the students of the Johns Hopkins University for their help and support. Among them a special thanks goes to Alessio Merlo, Daniele Panozzo, Matteo Dell'Amico, Matthew Green, Seny Kamara and Lucas Ballard.

Finally I would like to thank my family for always believing in me.

Table of Contents

Introduction

In the information age, communication over the Internet is of paramount importance for the life and development of our society. Exponential increase in computing power and dramatic decrease in the physical size and price of personal computers [Moo65] have contributed to an unprecedented widespread adoption of digital medias and communication tools among the general public.

The result of this is that more data than ever before are made available in digital format, which is significant because digital information is easier and less expensive than non-digital data to access, manipulate and store, especially from disparate, geographically distant locations. More information is generated on the first place, because of the ease and very low cost of doing so and because of the high value of data in an increasingly information-based society.

Data often substitutes for what would previously have required a physical transaction or commodity. On-line shopping is an egregious example of this. Automatically recorded data – customer's previous transactions and the products in which a specific customer is interested just to cite some – makes the use of on-line services easier and more convenient: as an example, shops can automatically build a list of suggested products which a customer might be interested in buying. This is helpful for the customer, which can find interesting items with no effort at all, and interesting for the merchant, which has higher chances to sell its products. As a result, others know more about us than ever before. Information about health, credit history, marital status, education, employment, address, travel history and personal tastes of the majority of individuals is readily available – or easily inferable from available data – to whoever has interest and enough resources to gather it.

Email and instant messaging applications allow people to communicate quickly and inexpensively with correspondents from any part of the world. To date, emails and IMs are largely sent unencrypted, therefore all the intermediate nodes which are involved in the delivery of the messages can easily record and violate the privacy of the communication.

Due to its inherently participative nature, Internet allows users to share, cooperate and interact with each other. In this context, privacy decisions often involve attempting to

control information flow in order to balance competing interests i.e. the cost and benefits of sharing or hiding personal information. Once user's information has been published, it is completely out of their control. In recent years on-line social networking has moved from niche phenomenon to mass adoption. The rapid increase in participation has been accompanied by a progressive diversification and sophistication of purposes and usage patterns across a multitude of different sites. Most on-line social network platforms share several core features. Among them, the most important one is the availability of a personal "profile": a representation of the user to others, with the intent of contacting or being contacted by other members of the social network for several different reasons: meet new friends or dates, find new jobs an receive or provide recommendations, just to cite some [GA05]. While social networking platforms share the basic purpose of on-line interaction and communication, specific goals and patterns of usage may vary across different services. Even if the visibility of information is usually somehow limited, participants are typically willing to disclose as much information as possible in order to gain more visibility on the social platform. However, even when some cautions are adopted, it is well known that private information about individuals, such as political or religious affiliation can be evinced from the information leakage derived from the correlation of all the information available about a specific member of the network. The privacy risks relative to social networks are therefore very serious.

With more reliable, affordable broadband access, the Internet no longer functions solely as a communications network: it has become a platform for computing. Rather than running software on your own computer or server, Internet users reach to the "cloud" to combine software applications, data storage, and massive computing power. There is no unanimously accepted definition of cloud computing. In general terms, its the idea that user's applications run somewhere in the "cloud", i.e. on the service provider's equipment, accessed via the Internet. The term "cloud" refers generally to any computer network or system through which personal information is transmitted, processed, and stored, and over which individuals have little direct knowledge, involvement, or control. Its easier to understand the concept of cloud computing by providing examples. Google operates several well-known cloud computing services. It offers its users applications such as e-mail, word processing, spreadsheets and storage, and hosts them "in the cloud". So, for example, users can create documents without maintaining any word processing software on their computers. People can use Googles software "in the cloud". All they need is an Internet-capable device; it doesnt even need to be a computer.

One of the advantages of cloud computing is that single users and small/medium sized businesses can instantly enjoy the benefits of the enormous infrastructure without having to implement and administer it directly. As the need for resources increases, additional service can be added as and when needed from the cloud computing vendor without having to pay for additional hardware.

However, when users store their data with programs hosted on someone else's hardware, they lose a degree of control over their sensitive information. The responsibility for protecting that information from attackers and internal data breaches then falls into the hands of the hosting company rather than the individual user. Law enforcement representatives trying to subpoena information could approach the cloud computing provider without informing the owner of the data. Some companies could even willingly share sensitive data with marketing firms. So there is a privacy risk in putting sensitive data in someone else's hands.

There are several different tools that users can adopt to protect their privacy. Encryption, when correctly instantiated, guarantees that no third party can violate user's privacy. Some of the threats, however, cannot be solved by simply adopting strong encryption. Anonymity and pseudonymity can be valuable for protecting users privacy. The term anonymity refers to the property of an entity of not being identifiable within the set from which it comes. If it is not identifiable, then the entity is said to be "anonymous". Anonymity is not an absolute. That is, the degree of anonymity [RR99] one enjoys may vary. To enable anonymity of a subject, there always has to be an appropriate set of subjects with potentially the same attributes. Pseudonyms are identifiers of subjects. The subject that may be identified by the pseudonym is the holder of the pseudonym [PH09]. Pseudonymity allows the holder of a pseudonym to build a reputation thanks to the linkability of each action performed using the same pseudonym.

This dissertation touches several different aspects of privacy, and provide several contributions to the field. Additionally, it introduces a new inter-domain authorization and authentication scheme, based on standard tools. Access control is a fundamental building block for the protection of the confidentiality of sensitive information.

This thesis deals with both theoretical and practical aspects of privacy-preserving tools. Practicality and deployability have been considered very important while designing our solutions. That is, all the proposed solutions are based on standard tools, and minimal requirement conditions are placed on entities in order to allow them to maintain their privacy during their interaction under reasonable assumptions.

Outline of this Thesis

The thesis is a collection of our work in the field of privacy, anonymity, and access control addressing several aspects of these topics.

This thesis is divided into four parts. The first part, covered by chapter 1, introduces different aspects of privacy and access control and the current state of the art. The second part focuses on some theoretical aspects of privacy. Chapters 2 and 3 provide novel contributions regarding anonymous identity-based encryption and deniable encryption. The third part of this dissertation is devoted to practical aspects and tools for preserving privacy. Chapters 4, 5, 6 and 7 provide novel constructions for privacy preserving digital certificates, inter-domain authentication and authorization, cloud-based deniable storage and anonymous peer-to-peer networks.

More specifically, this thesis is structured as follows:

Chapter 1 gives a background for the topics that will be treated in this thesis. Public-key infrastructures, identity-based encryption, inter-domain authentication and authorization, key-privacy, hybrid encryption, deniable encryption and anonymous peer-to-peer networks are introduced. The state of the art of these topics is briefly reviewed and discussed.

Chapter 2 introduces UAnonIBE, the first IBE providing universal anonymity (thus key-privacy) and secure under the standard quadratic residuosity assumption. The efficiency and ciphertext expansion of this scheme are comparable to those of Cocks IBE. We show that Cocks IBE and our anonymous variants are suitable in practice whenever hybrid encryption (KEM-DEM paradigm) is employed. Chapter 2 shows that UAnonIBE is a valid alternatives to the decidedly more expensive schemes introduced recently, (which, in addition, are anonymous but not universally anonymous). Incidentally, the results in chapter 2 can be used to simplify existing PEKS constructions and base their security on the standard quadratic residuosity assumption, which was left as an open problem in the area.

Chapter 3 brings deniable encryption to practical use by introducing the first efficient sender-and-receiver deniable *public-key* encryption scheme from standard tools alone. Deniable encryption allows the sender or receiver to open a ciphertext to a different value than what it encrypts. Several different instantiations of the deniable encryption scheme are proposed and analyzed. A formal treatment of the security of the schemes is also given. A small prototype demonstrates the practicality and the performances of the proposed scheme.

Chapter 4 introduces StemCerts, a new approach which allows the owner of a certificate to modify some parts of it without interacting with the certification authority.

StemCerts allows users to generate one-time and/or pseudonymous credentials The certification authority can decide which fields of the *slave* certificate can be altered, and which fields must keep the value that was agreed with the CA. As long as the owner of the certificate is the only entity who knows a specific secret associated with its certificate, it is also the only one who can renew and update already elapsed, one-time certificates. Thanks to the proposed approach, we provide digital certificates with the flexibility that is likely needed by current applications. Chapter 4 provides details for the implementation of StemCerts on a SSL-enabled web server and introduces a smart card-based implementation of StemCerts.

Chapter 5 presents a novel approach for inter-domain authorization based solely on standard digital certificates. We have argue that it is possible to realize flexible inter-domain authorizations within the X.509 specifications, which is the most widely deployed type of PKI across the industry. Chapter 5 shows how our X.509 extension helps collaborators maintain their functional autonomy. We also introduce the first X.509-compatible approach which allows rights amplification and degradation. The ability to quickly initiate/break collaborations while maintaining domain's functional autonomy is specially very useful for ephemeral collaborations. The treatment provided to permissions in chapter 5 can be easily extended to roles when the collaborating domains have RBAC as their underlying access control framework.

Chapter 6 introduces the first cloud-based deniable filesystem and shows, through experiments and performance evaluation, that it is a practical and usable tool for protecting users privacy. Deniable cloud storage allows users to collaborate and store data in such a way that if an adversary coerces a user into opening all encrypted data, the information still stays secret. We develop a prototype of a deniable file system DenFS which realizes this functionality.

Chapter 7 analyzes the integrity issue in anonymous P2P networks based on network coding. To this regard, it provides a novel mechanism for identifying univocally any block on the anonymous peer to peer network based on a homomorphic hash algorithm. It also studies an efficient integrity check based on Batch Verification for the case of PANGEA, an anonymous peer to peer file sharing network that we are currently developing.

Chapter 8 concludes this dissertation.

Chapter 1

Related Work

Digital Certificates

In 1978 Rivest, Shamir and Adleman [RSA78] proposed the first public-key encryption scheme. Using their scheme it was possible for a user to securely send a message to another user without the two having any pre-existing shared secret and without the receiver knowing that it was about to receive a message. This is of paramount importance and today is the basis for secure and private communication. However, real-world deployment of public-key encryption has shown to have significant problems. In particular the sender must be sure that the public key that it associates with the receiver really belongs to the intended receiver. This confidence is usually obtained through a public-key infrastructure (PKI), which is composed of one of more trusted third parties that can be relied upon to check a receiver's identity and vouch for the connection between that identity and a particular public key. In any real-word system, public key management is the most costly and inefficient part of any framework that makes use of public-key cryptography.

Digital certificates provide a basis that allows entities to trust each other. A digital certificate is a data structure that binds a public key to a subject. The binding is asserted by having a trusted CA digitally sign each certificate. The CA may base this assertion upon technical means (e.g., proof of possession through a challenge- response protocol), presentation of the private key, or on an assertion by the subject [FB02]. A certificate has a limited valid lifetime which is indicated in its signed contents. Because a certificate's signature and timeliness can be independently checked by a anyone, certificates can be distributed via untrusted communications and server systems, and can be cached in unsecured storage.

Digital certificates, and in particular X.509 certificates, were not designed to take privacy into consideration. Scalable privacy support within such infrastructures requires the avail-

ability of a privacy architecture that supports the implementation of basic privacy principles. Research efforts involving privacy architectures are currently on-going within the European PRIME project [PRI05] as well as in the NSF-funded PORTIA project [POR05], just to cite a few. Such architectures include both privacy-enhancing authorization mechanisms allowing decisions to be based on the requesters' (certified) attributes rather than on their identity, and languages for requesting general (certified) statements from a party. Moreover, due to their nature, digital certificates are immutable: users cannot alter any of the fields of a certificate without invalidating the signature apposed by the CA.

Users personal information changes more frequently than what one would usually expect. According to [And08], about 25% of Cambridge phone book addresses change every year, and the turnover is probably higher for e-mail addresses. A project aimed at developing a directory of people who use encrypted e-mail, together with their keys, found that changes in e-mail addresses were the main reason for changed entries [ACL99]. On the contrary, the loss or theft of keys was never reported as a cause for such changes. It is therefore clear that, in order to suit general users, digital certificates must provide flexibility to accommodate user's needs over time.

StemCerts[CG06] allow a user to modify predefined parts of her certificate in a limited and controlled fashion, without any interaction with the Certification Authority after the certificate has been signed. For instance, the certificate owner might be allowed to modify the "name" field, and/or the "expiration date" field, and/or the "public key" field of a standard X.509 v3 data structure after it has been issued, while keeping the validity of the CA's signature, and still letting the CA – and only the CA – know which is the real identity of the user who is using a particular certificate. Only the legitimate owner can modify the information contained in the certificate.

From the technical point of view the result was first achieved by adopting Chameleon Hash functions instead of the usual cryptographic hash functions for the certificate signature. A Chameleon Hash function [KR00] is a *trapdoor* collision-resistant hash function: without knowledge of the trapdoor information, a Chameleon Hash function has the same characteristics of any cryptographic hash function, such as pre-image and collision-resistance. However, the value calculated using this kind of hash depends on a public key. There is a corresponding private key – the trapdoor – which gives the ability to efficiently calculate collisions and second pre-images. In [AdM04] Ateniese and De Medeiros propose several schemes which provide key exposure freeness. In particular, they described a scheme related to a twin Nyberg-Rueppel signature (signature introduced in [NPS01]). We used it in our first StemCert implementation, because it has all the required features for our scheme, basically applying the idea of sanitizable signature to X.509 digital certificates. A sanitizable signature allows authorized, semi-trusted censors to modify – in a limited and controlled fashion – parts of a signed message without interacting with the original signer. Sanitizable signatures were first introduced by Ateniese et al. in [ACdMT05].

Inter-domain Authentication and Authorization. Several proposals exist in literature to address collaboration in distributed environment. Most of these proposal are policy based approaches in which certificates are used as assertions and actual authorizations of a user are computed based on policy-based language. In authentication-cum-authorization approach [LN99] certificates play a role of identity authentication and in policy based approach they play a role of conveying assertions. In a dynamic distributed setup, policy based authorization mechanism provide a better solution over authentication-cum-authorization mechanism. Policy based authorization mechanisms [BFS98, BFIK99, Sec05, XAC05, Web, JBBG04, LMW02a, LMW$^+$02b] overcome the shortcomings like, for example, context-sensitive authorizations, dynamic rights amplification, suspension or degradation of rights. Certificates are prone to revocation in a dynamic setup if one does not carefully choose the "security assertion values" (permissions) to be embedded into the certificates. It is a common practice to insert only the information that is not going to change for a relatively longer time period, and dynamic information is captured and interpreted separately, using a policy language [BFS98, BFIK99, Sec05, XAC05, LMW02a]. The problem overlooked by existing approaches is to make a systematic distinction between dynamic and static information (permissions), which we feel is almost impossible or cannot be precisely captured *a priori* while issuing the certificates [PS04]. Through our approach, we put forward a mechanism that shields the authorization certificates from the need of revocation/re-issuance in synchronization with the dynamic state changes in a domain.

X.509 was originally conceived to authenticate the entries in X.500 directory structure. Later on, it was exploited to perform authentication-cum-authorization decisions over resources scattered across independent administrative domains. Efforts to embed authorizations of a subject into the certificate itself were made through certificate extensions [ITU05]. The obvious challenge in such an approach of embedding is to maintain the certificate's validity due to change in subject's authorization status. This challenge led to the need for separating authentication and authorization of a subject, and attribute certificates [FH02] were conceived. Attribute certificates provide the foundation upon which the Privilege Management Infrastructure (PMI) can be built. They don't contain any public key, but attributes that may specify group membership, role, security clearance or other authorization information corresponding to the attribute certificate holder. A subject may have multiple attribute certificates associated with each of its PKCs. There is no requirement that the same authority create both the public key certificate and attribute certificate(s) for a user. This also brought along the need for attribute authority (AA, similar to Certificate Authority – CA) and attribute revocation lists (ARLs). PERMIS [CO02], Akenti [TJM$^+$99], Argos [JD96], Shibboleth [Shi05], CAS/Globus [CAS04], WS-Security [Web], SALSA [KPF01], etc., are some of the existing inter-domain authorization mechanisms or frameworks that mainly rely on X.509 type of PKI. There also exist policy based approaches like PolicyMaker/KeyNote [BFS98, BFIK99] that use cryptographic security assertions to derive to an authorization decision. RBAC (Role-Based Access Control [FSG$^+$01]) is a *de*

facto standard in industry to perform authorizations over an organization's resources by its users. In [HMM+00, LN99, SYJS01], the authors propose a X.509 based approach to extend the framework of RBAC across domains. There also exist standards like SAML [Sec05], XACML [XAC05], RT/RTML [LMW02a, LMW+02b] meant for designing inter-operation interfaces for organizations that need to collaborate. Specification languages [Sec05, XAC05] and frameworks [Web] have gathered much relevance in work-flow and grid computing fields.

A deep analysis of these practices made us conclude that in most of the existing approaches for collaboration, cryptographic primitives are mainly used to perform authentication. The authorization related attributes are specified in XML-like language with a plausible integration of cryptographic primitives over such attributes to provide authenticity and non-repudiation properties for the credentials flowing across domains. Policy-based approaches may not be able to quickly gear up for collaborations as the participating domains may have different policy languages used in their setups. Domains' transition from post-collaboration to pre-collaboration state may not be smooth and quick. Therefore, it was interesting for us to investigate if we could design a mechanism purely within X.509 framework. We would like to quickly highlight that though policy based inter-domain access control mechanisms (SAML, XACML, RT/RTML, *et. al.*) are more expressive than our approach, it would be unfair to compare them with our mechanism as they fall in different categories.

Identity-based Encryption and Key Privacy

One way to avoid the need for a public key infrastructure is to use an identity-based encryption scheme. Identity-based encryption was introduced by Shamir in 1984 [Sha84]. An identity-based encryption (IBE) scheme allows a pair of users to communicate securely without exchanging any key and without keeping key directories. Clearly an entity is not expected to compute its own private key. IBE schemes assume the existence of a trusted key generation center, whose purpose is to initially setup the system and use their secret knowledge to compute private keys for the requesting entities.

Several partial and inefficient solutions were proposed after Shamir's initial challenge but it was only recently that Sakai et al. [SOK00], Boneh and Franklin [BF03b], and Cocks [Coc01] came up with very practical solutions.

The Boneh-Franklin work has been the most influential of all: it did not just introduce the first practical IBE scheme but, more importantly, it provided appropriate assumptions and definitions and showed how to pick the right curves, how to encode and map elements into points, etc.

Cocks' scheme is *per se* revolutionary: it is the first IBE that does not use pairings but rather it works in standard RSA groups and its security relies on the standard quadratic

residuosity assumption (within the random oracle model). Cocks IBE, however, encrypts the message bit by bit and thus it is considered very bandwidth consuming. On the other end, Cocks [Coc01] observes that its scheme can be used in practice to encrypt short session keys in which case the scheme becomes very attractive. We may add that the importance of relying on such a standard assumption should not be underestimated. In fact, this is what motivated the recent work of Boneh, Gentry, and Hamburg [BGH07] where a new space-efficient IBE scheme is introduced whose security is also based on the quadratic residuosity assumption: a t-bit message is encrypted into a $n + t + 1$-bit ciphertext, where n is the security parameter. Unfortunately, as they point out in their paper, the time efficiency of their scheme is far from ideal. The encryption time in all practical public-key and IBE systems, such as RSA, Cocks IBE [Coc01] and Boneh and Franklin IBE [BF03b], is cubic in the security parameter. The encryption time in the Boneh et al. scheme is quartic in the security parameter per message bit [BGH07]. For each bit in the plaintext, the encryption algorithm must find the solution $(x, y) \in (\mathbb{Z}_N)^2$ to an equation of the form $Rx^2 + Sy^2 = 1$ and the solution $(\alpha, \beta) \in (\mathbb{Z}_N)^2$ to an equation $u\alpha^2 + S\beta^2 = 1$. For this reason, the encryption algorithm is particularly inefficient and may take several seconds to complete even on a fast machine.

However, the scheme of Boneh et al. [BGH07] has an important advantage over the scheme of Cocks: it provides anonymity, i.e., nobody can tell who the intended recipient is by just looking at the ciphertext. Anonymity, or key-privacy, is a very important property that was first studied by Bellare et al. [BBDP01]. Recipient anonymity can be used, for example, to thwart traffic analysis, to enable searching on encrypted data [BCOP04], or to anonymously broadcast messages [ABC$^+$05].

A popular example of anonymous broadcast is the blind carbon copy feature available in every email client and provided by the SMTP protocol. As shown in [BBW06], popular software such as PGP and Outlook's S/MIME are vulnerable to attacks on recipient's privacy. For instance, if not specified otherwise, PGP completely exposes recipient identities, including blind-carbon-copy recipients, since the ciphertext reveals the key IDs of all the recipients. A key ID is essentially the hash of the key and it is used to univocally identify it. Once the key ID is known, it is possible to query a key server to retrieve the identity of the recipient.

PGP allows to omit the key ID replacing it with all zeros. Omitting those IDs increases the amount of work needed to decrypt a message. A message without IDs, encrypted to n recipients, contains n unidentified ciphertexts. To decrypt a message, on average every recipient must try to decrypt each ciphertext, performing on average $n/2$ decryption operations. Anyway, even omitting key IDs PGP does not achieve recipients privacy: when PGP generates ElGamal public keys, it does so in the group of integers modulo a random prime. Thus, different principals are very likely to have public keys in different groups, making PGP encryptions vulnerable to passive key privacy attacks.

Microsoft Outlook, like many S/MIME clients, is vulnerable to attacks on recipient privacy as also shown in [BBW06]. When encrypting a message using S/MIME, Outlook labels each message with the issuer and serial number of each recipient's public key certificate. Many certificate authorities provide an on line service which translates certificate serial numbers into the certificates containing identity information thus potentially compromising the privacy of BCC recipients. Moreover, if a recipient uses a self signed certificate Outlook includes its full name and email address in the message sent to all recipients.

A possible solution to the anonymity of the BCC users is to modify and instruct the SMTP server to remove the headers of users in the BCC field before sending the message out. However, this requires substantial changes to the SMTP software that would have to be adopted on a large scale. Anonymous IBE offers a viable alternative that could be deployed to eliminate all possible passive attacks on the ciphertext. A passive attacker could determine the identity from the encrypted message with only negligible probability.

Several IBE schemes provide anonymity, for instance the Boneh-Franklin scheme is anonymous. Other schemes that do not originally provide anonymity can be either properly modified [BW06] or adapted to work in the XDH setting [Sco02, BBS04, BGdMM05, ACdM05].

At this point, it is natural to ask whether it is possible to enhance Cocks IBE and come up with a variant that provides anonymity and that, unlike Boneh et al.'s scheme [BGH07], is as efficient as the original scheme of Cocks. The first attempt in this direction has been proposed by Di Crescenzo and Saraswat [CS07]. They provide the first public-key encryption with keyword search (PEKS) that is not based on pairings.

A PEKS scheme [BCOP04] gives a trusted third party the ability to test wether some keywords chosen by the recipient are mentioned in a message. This third party learns nothing else about incoming emails. CPA-secure encryption schemes do not allow a third party, who has no knowledge of the decryption key, to check the presence of a specific keyword by simply observing the ciphertext they produce. In some cases this can be considered a limitation: suppose that a user wants to read its email on different devices, depending on which keywords they include. As an example, when a message contains the keyword *urgent*, that message is routed to the user's cellphone. The only way to provide this functionality is to allow the email gateway to decrypt the whole content of any message encrypted for that particular user, violating the user's privacy. Using a PEKS scheme, the email gateway can provide this functionality without violating the privacy of the user. Abdalla et al. observe in [ABC$^+$05] that a PEKS scheme can be constructed from any anonymous IBE scheme. They also prove that an anonymous IBE scheme can be constructed from a PEKS scheme, showing that the two notions are indeed equivalent.

The scheme of Di Crescenzo and Sarawat is secure under a new assumption they introduced which they called the *Quadratic Indistinguishability* assumption. Basically, their

assumption states that it is hard to distinguish between the following two distributions:

$$D_0 = \{(a, c, N) : N \stackrel{R}{\leftarrow} Blum(1^n),\ a \stackrel{R}{\leftarrow} \mathbb{Z}_N^*[+1],\ c \stackrel{R}{\leftarrow} \{c\ \ s.t.\ \ (c^2 - 4a) \in \mathbb{Z}_N^*[-1] \cup S_a[N]\}\}$$

and

$$D_1 = \{(a, c, N) : N \stackrel{R}{\leftarrow} Blum(1^n),\ a \stackrel{R}{\leftarrow} \mathbb{Z}_N^*[+1],\ c \stackrel{R}{\leftarrow} \mathbb{Z}_N^*\}$$

where N is the product of two $n/2$ bits Blum primes, $QR(N)$ is the set of quadratic residues in $\mathbb{Z}_N^*$ and $\mathbb{Z}_N^*[\alpha]$ is the set of elements in $\mathbb{Z}_N^*$ with Jacobi symbol α. The PEKS scheme introduced in [CS07] is built out of an anonymous IBE scheme based on Cocks IBE. Although their scheme is suitable for PEKS, it is quite impractical when turned into an IBE scheme. In particular, for a t-bit plaintext, the ciphertext expansion in their scheme becomes $8t \cdot n$ (where n is the security parameter in Cocks IBE, e.g., $n = 1024$) and each recipient must store $4t$ secret keys. Indeed, if $W \in \{0,1\}^t$ is a keyword then the PEKS encryption algorithm outputs $4t$ elements in $\mathbb{Z}_N^*[+1]$, that is $a_1 = H(W \mid\mid 1)$ $, \ldots, a_{4t} = H(W \mid\mid 4t)$ where H is a collision resistant hash function which maps strings of arbitrary length in elements of $\mathbb{Z}_N^*$ with Jacobi symbol 1, and releases the encrypted keyword as $(s_1, \ldots, s_{4t})$, where $s_i^2 - 4a_i$ has Jacobi symbol 1. Whenever s_i is in the ciphertext space for identity $W \mid\mid i$, it will encrypt the bit "1". The decryption algorithm takes as input $4t$ secret keys and checks that all the s_i in the ciphertext space encrypt the bit "1". When used as IBE for arbitrary messages, clearly any $s_i \in S_{a_i}[N]$ could also encrypt the bit "-1", but then the sender does not know whether a_i or $-a_i$ is a square and thus must encrypt for both (as in Cocks IBE), thus effectively doubling the amount of bandwidth required.

Although the scheme by Di Crescenzo and Saraswat [CS07] is suitable for PEKS, we note that when used as an IBE it becomes quite impractical: it uses four times the amount of bandwidth required by Cocks and it requires each user to store and use a very large number of secret keys (four keys per each bit of the plaintext). In addition, the security of their scheme is based on a new assumption they introduce. Later in this thesis we show that their assumption is equivalent to the standard quadratic residuosity one.

Universal anonymity is a new and exciting notion introduced at Asiacrypt 2005 by Hayashi and Tanaka [HT05]. An encryption scheme is universally anonymous if ciphertexts can be made anonymous by anyone and not just by whoever created the ciphertexts. Specifically, a universally anonymizable public-key encryption scheme consists of a standard public-key encryption scheme and two additional algorithms: one is used to anonymize ciphertexts, which takes as input only the public key of the recipient, and the other is used by the recipient to decrypt anonymized ciphertexts. As described in [HT05], a company may mandate that internal emails should be encrypted but not anonymous (for efficiency and legal requirements) when routed internally but be anonymized whenever they are sent outside the domain of the company. Using a universally anonymous encryption, a gateway server can be easily setup to anonymize any ciphertext sent outside the domain while internal clients can encrypt messages efficiently.

The main aspect characterizing universal anonymity is the ability to separate the role of the sender of encrypted messages from the role of the anonymizer. It is trivial to achieve this "separation of roles" with the Boneh-Franklin scheme or with any other anonymous scheme: it is enough to append to the ciphertext a claim about the recipient public key together with a "proof of consistency", i.e., a proof that the claim about the recipient key is valid. In order to restore anonymity, this extra information that is appended to the original ciphertext can simply be removed and this can be done without knowing any secret information. Depending on the underlying IBE scheme, this "proof of consistency" may or may not be efficient. In addition, this approach inevitably makes the non-anonymous scheme more expensive than the anonymous version while ideally the opposite should hold. One could try to find a variant of Boneh-Franklin IBE that is not anonymous but more efficient than the original scheme and then make it universally anonymous by applying the techniques in [HT05]. Even assuming that this is possible, the new scheme would be different and more expensive than the original one and still depending on pairing-based assumptions.

The following observations are obvious but worth emphasizing: (1) A universally anonymous scheme is also key-private in the sense of Bellare et al. [BBDP01]. What makes universally anonymity interesting and unique is that anyone can anonymize ciphertexts using just the public key of the recipient. (2) Key-private schemes can be more expensive than their non-private counterparts. For instance, RSA-OAEP can be made key-private as shown in [BBDP01] but the new anonymous variant is more expensive. (3) The concept of universal anonymity makes sense also for schemes that are already key-private. For instance, ElGamal is key-private only by assuming that all keys are generated in the same group and participants share the same public parameters. But in many scenarios this is not the case. In PGP, for instance, parameters for each user are selected in distinct groups. Evidently, ElGamal applied in different algebraic groups is not anonymous anymore as one can test whether a given ciphertext is in a group or not.

It is well-known that in order to encrypt long messages, asymmetric encryption can be used in combination with symmetric encryption for improved efficiency. This simple and well-known paradigm has been formalized only recently by Cramer and Shoup [CS98, CS04] and Shoup [Sho]. It is introduced as the KEM-DEM construction which consists of two parts: the key encapsulation mechanism (KEM), used to encrypt a symmetric key, and the data encapsulation mechanism (DEM) that is used to encrypt the plaintext via a symmetric cipher.

Cramer and Shoup [CS04] showed that a hybrid encryption scheme is CCA secure (i.e., secure against the adaptive chosen ciphertext attack) in the standard model if the KEM component is CCA secure and the DEM component is a CCA-secure one-time symmetric encryption. Later, Kurosawa and Desmedt [KD04] showed that the KEM component does not have to be CCA-secure as long as the CCA-secure one-time symmetric

encryption satisfies an extra condition (which is satisfied by the DEM scheme proposed by Shoup [CS04, Sho, Sho00]). The construction of CCA-secure one-time symmetric encryption is standard and it is usually accomplished by coupling a message authentication code (MAC) with a symmetric encryption. In particular, it can be shown that applying a one-time MAC on the output of a CPA-secure symmetric encryption results in a CCA-secure symmetric encryption scheme (see e.g. Cramer and Shoup [CS04]).

In chapter 2 we introduce a variant of the Cocks IBE which can be proven secure only in the random oracle model. Thus, it makes sense to consider KEM-DEM constructions that are CCA-secure in such a model.

In this case, the most relevant work is the one from Fujisaki and Okamoto [FO99] that shows how to build CCA-secure hybrid encryption schemes and how to convert any CPA-secure asymmetric scheme into a CCA-secure one in the random oracle model. Even more relevant is the work of Bentahar et al. [BFMLS08] that formalizes the concept of id-based KEM-DEM and provides a generic transformation from any IBE scheme to CCA-secure ID-based KEM in the random oracle model. In particular, it is possible to show (see, e.g., Bentahar et al. [BFMLS08]) that if a KEM returns $(\text{Encrypt}_{\mathsf{IBE}}(K), F(K))$, where $\text{Encrypt}_{\mathsf{IBE}}(K)$ is a one-way encryption for an identity and F is a hash function modeled as a random oracle, then the combination of this KEM with a CCA-secure DEM results in a CCA-secure hybrid encryption. (Note that, unless the encryption is a permutation, an additional random oracle is needed within the encryption to compute the randomness from the plaintext, see Bentahar et al. [BFMLS08] for details.)

Deniable Encryption

An adversary can violate data privacy not only when the information is in transit, but also when it is *at rest* at one of the end points of the communication. Regular encryption works well against an adversary who doesn't have access to the decryption key. Semantic security basically implies that an adversary with access to an encrypted file learns nothing about the corresponding unencrypted content. Whatever it can compute after observing the ciphertext could be also calculated from scratch. Several tools are available to end users to encrypt their documents. As an example, full disk encryption allows a user to encrypt the whole content of a disk drive rendering it useless unless the decryption key is revealed to the adversary. When the adversary has access to the private information associated with an encrypted message (e.g. the decryption key or the randomness used to encrypt the message), traditional encryption fails to preserve data confidentiality. Consider the following scenario: an adversary obtains access to a laptop computer, the owner of which keeps the stored data encrypted. The adversary forces the owner to decrypt the data by revealing any key material and necessary auxiliary information. The owner would certainly like to have the ability to open the data in two different ways through a usable mechanism,

e.g., by entering a password that will unlock the key material and decrypt the data. That is, the data owner can be in possession of two different passwords: one uncovers the true data stored on the laptop, and another uncovers a different (and meaningful) version of the data stored on the disk. These requirements are captured by the notion of deniable encryption.

A deniable encryption scheme is an encryption scheme with an additional property that allows the sender to open the ciphertext in such a way as to reveal a different version of the plaintext than the one originally used in producing the ciphertext. The type of deniable encryption where the ciphertext sender is coerced and opens the ciphertext is called *sender-deniable* encryption. Similarly, *receiver-deniable* encryption can be defined as a deniable encryption scheme where ciphertext recipient is coerced into opening the ciphertext. Additionally, *sender-and-receiver deniable* schemes can be defined to combine both of the above properties. Deniable encryption was introduced in 1997 by Canetti et al. in [CDNO97]. Their work shows that public-key ad-hoc encryption can be built using translucent sets and trapdoor permutation. Each ciphertext encodes a single bit and can be opened to either value. [CDNO97] also reports that a shared-key sender-deniable plan-ahead encryption can be constructed by either concatenating encryptions of several messages under different keys (and thus increasing the ciphertext size) or sharing a key proportional in size to the message length. Additionally, [CDNO97] shows that sender-deniable encryption can be converted to receiver-deniable encryption using an extra round of communication; namely, the receiver uses a sender-deniable scheme to transmit encryption of a random bit r to the sender, and the sender transmit $b \oplus r$ to the receiver in the clear, where bit b is the actual message to be transmitted. (Similarly, receiver-deniable encryption can be converted to sender-deniable.) This interactive solution, however, is not applicable to the scenario of encrypted storage considered in this work.

Ibrahim [Ibr09a] gives ad-hoc sender-deniable public-key encryption schemes based on the hardness of the quadratic residuosity problem for a composite modulus N, which is a product of two large primes. The first scheme allows one to transmit one bit per ciphertext, where the ciphertext size is $\log N + 2\ell + 1$ and ℓ is the output size of a hash function. The second scheme can incorporate several message bits into a single ciphertext, but still can transmit only messages of a few bits long. More precisely, to transmit m bits, the ciphertext size is $m \log\log N + \log N + 2^m \ell$.

Ibrahim [Ibr09b] gives an ad-hoc receiver-deniable public-key scheme, which is also more efficient in terms of its bandwidth than other ad-hoc deniable schemes. In particular, the scheme is based on RSA and 1-out-of-n oblivious transfer. It permits transmission of messages of $\log N - \delta$ bits long with ciphertexts of size $2 \log N$, where N is the RSA two-prime modulus and δ is a randomizing string (e.g., 128 bits long). This solution, however, requires usage of mediated RSA, where a user does not have full knowledge of its secret keys, i.e., each private key is split between the owner of the key and a third party. This

means that such a solution have a limited applicability and will not work in our setting. Additionally, this solution relies on trusted hardware such as RSA-SecureID tokens [RSA].

Another recent work of Klonowski et al. [KKK08] extends the ad-hoc sender-deniable public-key scheme of Canetti et al. through a nested construction that additionally uses trapdoor permutations. The authors also show how hidden messages can be transmitted using standard ElGamal encryption, which achieves the functionality of plan-ahead receiver-deniable encryption. In particular, the sender and receiver are assumed to share a secret S, and the sender also has access to the receiver's private key (associated with the public key the encryption scheme uses). This allows a ciphertext to encode, besides a regular fake message, an additional secret message of the same size. In fact, this provides a broad-band subliminal channel if the receiver is willing to trust the sender with its private key. Meng and Wang [MW09] extent the idea of Klonowski et al. [KKK08] providing a receiver deniable encryption scheme based on BCP commitment scheme of Bresson et al. [BCP03]. Their scheme does not require sender and receiver to exchange any pre-encryption information.

Deniability in Practice. TrueCrypt [Tru] is a widely used disk encryption tool that provides deniability in the form of a deniable filesystem. TrueCrypt refers to these deniable filesystems as *hidden volumes*. In a typical scenario, a hidden volume is created inside a non-deniable encrypted disk image, using the space marked as not allocated by the filesystem. To mount the hidden volume, a user must first provide the password to decrypt the non-deniable filesystem and then provide a second password to decrypt and mount the hidden volume [Hidb]. If the second password is not known, the unallocated space is indistinguishable from random data. Since TrueCrypt always fills the unallocated space with random data, the existence of the hidden volume can be denied. Moreover, the non-deniable encrypted disk image should contain some sensitive-looking files which the user is not actually interested in hiding, in order to provide plausible deniability [Hidb].

Users must be aware that, when the hidden volume is not mounted, it appears as empty space in the *outer* non-deniable encrypted filesystem. Therefore writing new data on the *outer* filesystem can lead to overwriting part of the hidden volume. There are some forms of integrity protections for the hidden volumes which require a separate password and are claimed not to compromise its deniability.

In their paper [CHK+08], Czeskis et al. examine the efficacy with which TrueCrypt v5.1a provides a deniable filesystem. What they found is that Microsoft Windows Vista and some commonly used applications and utilities, such as Microsoft Word and Google Desktop, compromise the deniability of TrueCrypt hidden volumes. As an example, Windows Vista keeps track of the serial number of each volume it mounts, together with the drive letter assigned to it. It also creates a link to a document every time that document is opened. Both these features clearly compromise the deniability of a hidden volume. They conclude that there are several fundamental challenges to the creation and use of any de-

niable filesystem: even if a filesystem may be deniable in the pure, mathematical sense, the environment surrounding that filesystem can undermine its deniability as well as its contents [CHK+08].

After Czeskis et al.'s paper was published, a new version of TrueCrypt was released with support for hidden operating systems [Hida], which should solve the problems highlighted in [CHK+08]. A hidden operating system is a system installed in a hidden TrueCrypt volume. TrueCrypt authors claim that it is impossible to prove that a hidden operating system exists. However, at the time of writing, there is no independent confirmation or denial of this claim. The idea is that by hiding the whole system, no information can leak to the non-deniable storage areas, and therefore the adversary cannot gather any meaningful information about the existence of a hidden operating system.

Assange and Weinmann proposed a deniable file system called Rubberhose [Rub]. Rubberhose transparently encrypts data on a storage device and allows users to hide that encrypted data. Released in 1997, it was claimed to be the first available practical program for deniable encryption. Instead of hiding the existence of encrypted data completely, Rubberhose makes hard for an adversary to guess how much information was actually encrypted. Its authors claim that thanks to this feature, a coercer will not be able to determine if all encrypted data was decrypted by the owner. Rubberhose needs a dedicated hard disk to work. It creates several partitions in which blocks are not contiguous; instead they are scattered in a pseudorandom manner across the drive. Clearly each partition must be encrypted using a different passphrase.

Phonebook FS [Pho] is a filesystem for Linux which offers encryption and plausible deniability. It works by creating several "layers" of encryption. In order to access the next "layer", a user must provide the correct passphrase. The authors claim that a coercer will not be able to determine the number of available layers, so users can plausibly deny the existence of one or more of them.

Unfortunately none of the available deniable filesystems provide a solution for the scenario proposed earlier in this section, where an adversary obtains access to a laptop and also records communication. The owner of the filesystem cannot deny the existence of the encrypted data for which an update was sent to its collaborators, since we are assuming that the adversary can capture every message exchanged between them. Moreover, all the available deniable filesystems require users to share a secret key, which may not be always possible and severely limits the flexibility of the filesystem, as discussed in section 6.2.

Anonymity and Peer-to-peer Networks

Current anonymous or censorship-resistant P2P networks do not perform any kind of user authentication, and try to unlink actions from users who perform them. Moreover, the very

goals of anonymity and censorship-resistance are perceived as a danger by some, usually powerful, entities interested in exerting control over user actions. Thus, it is no wonder that harmful behavior by malicious peers is considered as a primary threat in all these networks. One of the main threats for these systems is data corruption performed by adversarial peers. Thus, message integrity check is a primary need for these networks, in addition to techniques for corruption confinement.

Several general purpose integrity and/or authentication mechanisms exist; most of them are based on public key or secret key cryptographic algorithms. Integrity (and authentication) can be provided by message authentication code (MAC) algorithms and digital signatures. The aim of a message authentication code is to prevent an adversary from modifying a message sent from one party to another, without the parties detecting that a modification has been made [KL07]. To achieve this, the two parties must share a secret key that the adversary does not know. After running the MAC algorithm over a message using the shared key, the sender obtains a MAC tag. The tag is sent along with the message to the recipients, who applies the same algorithm on the message and verifies the correspondence between the tag it received and the tag it calculated. If the tags don't match, the message is discarded as invalid.

Digital signatures are another way to verify integrity and authenticity of a message. Unlike MACs, the key used to generate a signature (i.e., the private key) is different from the key used to verify the signature (i.e., the public key). Digital signatures schemes can thus be easily obtained from existing public key encryption algorithms, like RSA-PSS [BR96], Rabin [BR96] or ElGamal[1] [Gam85], or from newly designed primitives, like DSS [Nat00]. It is important to note that with digital signatures, the association between a user and its public key must be obtained either by querying the user on a secure channel, or through a trusted third party.

When both privacy and integrity are required, a Chosen Ciphertext Attack (CCA) secure encryption algorithm [BDR98] provides integrity "for free". CCA-security is known to be equivalent to non-malleability [BS99], and therefore all CCA-secure encryption algorithms guarantee that the recipient of an encrypted message can detect any modification to the ciphertext performed by a resource-bounded adversary [BS99]. However, not all public key encryption schemes provide CCA-security. Kurosawa and Desmedt [KD04] have shown that a hybrid encryption scheme is CCA-secure even if the public key algorithm used is not CCA-secure as long as the symmetric cipher provides a CCA-secure one-time encryption. The construction of CCA-secure one-time symmetric encryption is standard and it is usually accomplished by coupling a MAC with a symmetric encryption. In particular, it can be shown that applying a one-time MAC on the output of a Chosen Plaintext Attack (CPA) secure symmetric encryption results in a CCA-secure symmetric encryption scheme (see

[1] Bleichenbacher showed in [Ble96] a method to forge ElGamal signatures if the public parameters are not chosen properly

e.g. Cramer and Shoup [CS04]). The same result is obtained, with block ciphers, using an authenticated mode of operation instead of a MAC. This solution usually provides a slightly higher efficiency and a simpler construction. There are several provably secure authenticated modes, like OCB [Rog00], GCM [MV04] or CWC [KV04].

Anonymous and censorship-resistant P2P networks cannot trust any third party or central authority and must work in absence of user identities, so they need different kinds of data integrity techniques. To make things even more complex, most such systems gain anonymity by repeatedly routing a message from one node to another, so as to gradually lose the sender-related information carried by a message along its way. This approach poses at least two challenges: queries and responses must be made unintelligible to intermediate nodes, the loyalty of which cannot be relied upon; at the same time, intermediate nodes should be made capable of filtering out corrupted messages, in order to confine a possible attack on data integrity.

The main encoding scheme of Freenet [CSWH00], called CHK (Content-Hash Key), associates any block of data D to a statistically unique address in a distributed repository. The address is yielded by a cryptographic hash H (the paper mentions SHA-1). The data are first encrypted under a symmetric cipher, using a randomly-generated key P, then stored at address $H(D)$. The information for retrieving D is thus the CHK $\langle H(D), P\rangle$, which is *not* stored with the data itself. The CHK is thus oblivious of the identity of the user who had originally inserted D. A requestor node will send $H(P)$ as a query, and subsequently use P for deciphering the response. The requestor does not disclose P to any other node, in order to prevent others from inspecting and possibly censor the response. The CHK also serves for the integrity check: once D is delivered back and deciphered, hashing D must yield back the CHK. With this technique, however, data tampering cannot be confined as the integrity check can only be accomplished by the requestor node (which is the only one that knows P), so no early filtering of invalid data is possible. Another drawback is that even the smallest update on D requires changing the CHK, so CHKs are impractical as references to writable data. To overcome the latter problem, CHKs are more often associated to human-readable descriptions by way of indirection entities called SSKs (Signed Subspace Keys). A SSK denotes a block of metadata M, stored under an address Q, that may contain one or more CHKs associated to human-readable keywords, and also has a human-readable description for itself as a whole. An anonymous user is allowed to create a SSK by choosing the human-readable description string $Desc$; the system then generates a random key $\langle P_k, S_k\rangle$ for an asymmetric cipher, encrypts the metadata using the description $Desc$ as a symmetric key, signs the encrypted metadata under S_k, then stores the encrypted and signed metadata at address $Q = H(Desc \oplus P_k)$. The information for retrieving the metadata is thus the SSK $\langle P_k, Desc\rangle$, which again is *not* stored with the metadata itself. A requestor node will send $Q = H(Desc \oplus P_k)$ as a query; no other node can learn about the query or decipher the response, since both $Desc$ and P_k are kept undisclosed. Metadata may be changed later without altering the storing address Q, as Q

is not the hash of the content, provided the public key and the descriptive string remain the same (this amounts to say a SSK has a "name" and is permanently owned by the anonymous user who first created, named and signed it). Integrity check of SSKs relies on digital signatures, that however can only be verified by the requestor node, so tampering confinement is again impossible.

GNUnet [BG03] uses an encoding called ECRS (Encoding for Censorship-Resistant Sharing [GGHL05]) that improves over Freenet in various ways. With GNUnet, large files are segmented into fixed-size independent blocks, a feature that may greatly improve download performance by allowing so-called *file swarming*, that is, download of a file from more sources at a time[2]. Each block B of data is individually encrypted under the symmetric key $H(B)$, then the encrypted block E(B) is stored at address $H(E(B))$. The information for retrieving B is thus the CHK $\langle H(E(B)), H(B)\rangle$. The collection of CHKs denoting all file segments is then arranged in a logical tree, using indirection blocks similar to filesystem inodes which in the end converge to a root CHK. The root CHK may then be associated to human-readable keywords by means of something similar to a Freenet SSK. With GNUnet, a requestor node querying the CHK $\langle Q, K\rangle$ will send Q as a query; responses are of the kind $\langle Q, E(B)\rangle$, that is, the ciphered block travels back along with the query itself. Intermediate node cannot decipher the response, since they are oblivious of $K = H(B)$; yet, they can perform the integrity check (hashing $E(B)$ must yield Q), so tampered data can be filtered out very early, and an attack on data integrity essentially amounts to not forwarding data at all.

Other anonymous P2P networks rely on simple information dispersal for improving censorship-resistance. In some of these systems [DFM00, WO02], a block of data is encoded as a redundant set of segments, none of which is intelligible on its own; segments are then stored at different nodes, and a minimum subset of segments is needed for the original data to be restored [Rab89]. Publius [WRC00], instead, performs a dispersal of the encryption key using Shamir's algorithm [Sha79] then simply replicates the encrypted data to any number of nodes. These systems essentially trade robustness for performance and hardware resources, with all the difficulties arising from seeking a balance among conflicting requirements. In these systems, attacks to data integrity are mitigated by data replication and thus paid in terms of performance and storage resources.

There exist other anonymous networks, like Tor [DMS04] or MorphMix [RP02], which provide a multi-hop encrypted socket proxy service with no abstraction of a storage. Censorship resistance is not a primary goal with these systems, so pollution attacks are considered of lesser importance compared to attacks to anonymity. Message integrity in Tor is performed only at the two ends of each proxy circuit, by labeling each block of data with (the first four bytes of) the hash of the end-to-end session key (unknown to all other

[2]Recent versions of Freenet also include such a file segmentation feature, but we were unable to find a paper documenting it

nodes along the circuit) combined with the data itself; intermediate nodes can thus pollute data and this will only be detected at the receiver end of the circuit, with some waste of resources along the circuit. MorphMix seems not to address integrity issues at all.

A number of integrity mechanisms specific for network coding have been proposed so far. In [BFW09], Katz and Waters introduce an efficient homomorphic signature scheme that can be used to authenticate blocks. Their scheme is based on the computational Diffie-Hellman assumption in bilinear groups. Gkantsidis et al. [GR06] extend the homomorphic hash function of Krohn et al. [KFM04] to network coding. Their solution is severely limited by the need of a secure channel, which is used by forwarders and sinks to retrieve the hash values. Yu et al. [YWRG08] extend the work of Gkantsidis et al. by proposing a signature-based scheme that allows the source to delegate its signing authority to forwarders. Forwarders can generate new signatures without interacting with the original source only if the newly encoded blocks are not corrupted. This result is achieved using a homomorphic signature scheme.

Network Coding in Peer to Peer File Sharing Networks

Peer-to-peer (P2P) networks have been designed to support data exchange among interconnected peer entities. In a P2P network any peer acts both as a data requestor and a data provider. From a network perspective a peer accomplishes routing functionalities by sending data to neighbors.

P2P networks have been extensively studied as substrates for anonymous content distribution. One of the main goals is to make it unfeasible to identify the sender or the requestor of any block of data passing through the system. However, in these systems anonymity is usually obtained at the expense of poor performance: anonymous P2P networks are notoriously much slower than non-anonymous ones. In general, the privacy/performance tradeoff of anonymous P2P networks is still an open research issue. It is worth noting that both anonymity and P2P networks always rely on sufficiently many peers cooperating in the network, but poor performance discourages users from joining; thus, good privacy/performance tradeoffs are of primary importance for these networks to be deployed and gain user acceptance.

Recent studies (e.g. [CW07] and [BFW09]) have shown that the use of network coding as a routing technique may provide advantages in terms of throughput, delay, and fault tolerance in decentralized networks.

The central idea of network coding is that any node in the network can forward random linear combinations of incoming packets instead of merely forwarding unmodified packets. This implies that individual blocks of data shall be routed onto many network paths, although in combination with data from other sources. One notable consequence is that

redundant paths in the network are always exploited simultaneously rather than as alternatives. This leads to a potential increase of performance and fault tolerance, since there are more ways for a receiver to obtain a required data block. Moreover, different receivers may be actually sharing a common routing path where their requested data are traveling as linear combination bundles.

In general, the introduction of network coding in P2P system is useful for reducing the selfish behavior (free-riding) typical of those systems. Indeed, any peer needs to collect a high number of other blocks in order to be able to retrieve the required ones as an inverse linear combination. Therefore the adoption of network coding by itself promotes the sharing of blocks by selfish peers, even if some blocks are not of primary interest for them, just to be able to decode interesting ones.

Chapter 2

Universally Anonymous Identity-based Encryption Under Standard Assumptions

In this chapter we present UAnonIBE, the first anonymous identity-based encryption scheme which is both efficient and based on standard assumptions. We construct UAnonIBE by extending Cocks' scheme to support universal anonymity. Unlike previous proposals, our scheme has efficiency, storage, and bandwidth requirements similar to those of the original scheme by Cocks (which is not anonymous).

More specifically, in this chapter we provide the following contributions:

(1) We enhance Cocks IBE and make it universally anonymous, and thus key-private in the sense of Bellare et al. [BBDP01]. Our variant of Cocks IBE can be seen as the most efficient anonymous IBE whose security is based on the quadratic residuosity assumption. The efficiency of our scheme is comparable to that of Cocks IBE. In fact, it is substantially more efficient than the recent scheme of Boneh et al. [BGH07] and the IBE that derives from the PEKS construction by Di Crescenzo et al. [CS07]. In addition, the ciphertext expansion of our scheme is comparable to that of Cocks IBE.

(2) We implemented our variant and measured its performance. We show that in practice the efficiency of Cocks IBE and the variant we propose in this chapter compare favorably even with that of the Boneh-Franklin scheme.

(3) Incidentally, our solutions and techniques can be used to simplify the PEKS construction in [CS07] and our Lemma 2.2.2 in Section 2.2 can be used to show that the new security assumption introduced in [CS07] is actually equivalent to the standard quadratic residuosity assumption, thus making the elegant Di Crescenzo-Saraswat PEKS scheme the

first one whose security is based solely on such a standard assumption (which was left as an open problem in the area).

Part of this chapter appeared in "G. Ateniese and P. Gasti, Universally Anonymous IBE Based on the Quadratic Residuosity Assumption", Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009 [AG09].

2.1 Preliminaries

In this section, we recall first the IBE scheme proposed by Cocks [Coc01]. Then we show that Cocks IBE is not anonymous due to a test proposed by Galbraith, as reported in [BCOP04]. Finally, we show that Galbraith's test is the "best test" possible against the anonymity of Cocks IBE. We assume that N is a large-enough RSA-type modulus. Hence, throughout the chapter, we will omit to consider cases where randomly picked elements are in $\mathbb{Z}_N$ but not in $\mathbb{Z}_N^*$ or, analogously, have Jacobi symbol over N equal to 0 since these cases occur only with negligible probability[1]. Therefore, for consistency, we always assume to work in $\mathbb{Z}_N^*$ rather than in $\mathbb{Z}_N$ even though $\mathbb{Z}_N^*$ is not closed under modular addition.

We will denote with $\mathbb{Z}_N^*[+1]$ ($\mathbb{Z}_N^*[-1]$) the set of elements in $\mathbb{Z}_N^*$ with Jacobi symbol $+1$ (-1, resp.) and with $\mathbb{QR}(N)$ the set of quadratic residues (or squares) in $\mathbb{Z}_N^*$. The security of Cocks IBE (and our variants) relies on the standard quadratic residuosity assumption which simply states that the two distributions $DQR(n) = \{(c, N) : (N, p, q) \xleftarrow{R} Gen(1^n),\ c \xleftarrow{R} \mathbb{QR}(N)\}$ and $DQRN(n) = \{(c, N) : (N, p, q) \xleftarrow{R} Gen(1^n),\ c \xleftarrow{R} \mathbb{Z}_N^*[+1] \setminus \mathbb{QR}(N)\}$ are computationally indistinguishable, where n is a security parameter and $Gen(\cdot)$ generates a RSA-type n-bit Blum modulus and its two prime factors.

2.1.1 Cocks' IBE Scheme

Let $N = pq$ be a Blum integer, i.e., where p and q are primes each congruent to 3 modulo 4. In addition, we consider $H : \{0,1\}^* \to \mathbb{Z}_N^*[+1]$ a full-domain hash which will be modeled as a random oracle in the security analysis.

Master Key: The secret key of the trusted authority is (p, q) while its public key is $N = pq$.

[1]The Jacobi symbol of $a \in \mathbb{Z}_N$ is denoted as $\left(\frac{a}{N}\right)$ and is either -1, 0, or $+1$. However, $\left(\frac{a}{N}\right) = 0$ if and only if $\gcd(a, N) \neq 1$, thus this case happens only with negligible probability since the value $\gcd(a, N)$ would be a non-trivial factor of N.

Key Generation: Given the identity ID, the authority generates $a = H(ID)$ (thus the Jacobi symbol $\left(\frac{a}{N}\right)$ is $+1$). The secret key for the identity ID is a value r randomly chosen in $\mathbb{Z}_N^*$ such that $r^2 \equiv a \bmod N$ or $r^2 \equiv -a \bmod N$. This value r is stored and returned systematically.

Encryption: To encrypt a bit $b \in \{-1, +1\}$ for identity ID, choose uniformly at random two independent values $t, v \in \mathbb{Z}_N^*$, such that $\left(\frac{t}{N}\right) = \left(\frac{v}{N}\right) = b$, and compute:

$$(c, d) = \left(t + \frac{a}{t} \bmod N, v - \frac{a}{v} \bmod N\right)$$

Decryption: Given a ciphertext (c, d), first set $s = c$ if $r^2 \equiv a \bmod N$ or $s = d$ otherwise. Then, decrypt by computing:

$$\left(\frac{s + 2r}{N}\right) = b$$

Notice that $s + 2r \equiv w(1 + r/w)^2 \bmod N$, thus the Jacobi symbol of $s + 2r$ is equal to that of w, where w is either t or v.

Security Analysis: The security analysis provided by Cocks is quite ingenious. Cocks' scheme is CPA-secure (i.e., secure against the chosen plaintext attack) under the standard quadratic residuosity assumption in the random oracle model.

Assume there is an adversary A that breaks Cocks' scheme with probability $1/2 + \delta$. The simulator S is given a random $a \in \mathbb{Z}_N^*$ such that $\left(\frac{a}{N}\right) = +1$ and must decide whether a is a quadratic residue (*square*) or not.

When A asks for the secret key for identity ID_i, the simulator returns a random $r_i \in \mathbb{Z}_N^*$ and flips a coin. If the result is "tail" then S sets $H(ID) = r_i^2$ otherwise $H(ID) = -r_i^2$ (notice that, in both cases, the Jacobi symbol of $H(ID)$ will be $+1$). At a certain point, A selects a target identity ID^* and requests an encryption under ID^* of either -1 or 1. The simulator sets $H(ID^*) = a$, picks $b \in_R \{-1, 1\}$, and sends A the encryption $c = t + a/t \bmod N$, where $\left(\frac{t}{N}\right) = b$. If A returns correctly b then S states that a is a square. If A does not return b then S states that a is a non-square.

Clearly if a is a square then A returns $\left(\frac{t}{N}\right)$ with probability $1/2+\delta$. If a is not a square then the claim is that A returns $\left(\frac{t}{N}\right)$ correctly with probability $1/2$. Indeed, given $c = t + a/t$, consider three values t_1, t_2, t_3 defined as [Coc01]:

$$\begin{array}{ll} t_1 \equiv t \bmod p & t_1 \equiv a/t \bmod q \\ t_2 \equiv a/t \bmod p & t_2 \equiv t \bmod q \\ t_3 \equiv a/t \bmod p & t_3 \equiv a/t \bmod q \end{array}$$

By definition, $c \equiv t_1 + a/t_1 \equiv t_2 + a/t_2 \equiv t_3 + a/t_3 \bmod N$. Now, since a is a non-square and $\left(\frac{a}{N}\right) = +1$, we have that $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$. It follows that [Coc01]:

$$\left(\frac{t_1}{N}\right) = \left(\frac{t_2}{N}\right) = -\left(\frac{t}{N}\right) = -\left(\frac{t_3}{N}\right)$$

Thus, the adversary A cannot return $\left(\frac{t}{N}\right)$ correctly with probability more than $1/2$ whenever a is a non-square. Hence, the simulator S can use A to distinguish between squares and non-squares among elements with Jacobi symbol $+1$ in $\mathbb{Z}_N^*$.

2.1.2 Galbraith's Test (GT)

As mentioned in the paper by Boneh et al. [BCOP04], Galbraith showed that Cocks' scheme is not anonymous. Indeed, let $a \in \mathbb{Z}_N^*[+1]$ be a public key and consider the following set:

$$S_a[N] = \left\{ t + \frac{a}{t} \bmod N \mid t \in \mathbb{Z}_N^* \right\} \cap \mathbb{Z}_N^*$$

Given two random public keys $a, b \in \mathbb{Z}_N^*[+1]$, Galbraith's test (which we will denote with "$GT(\cdot)$") allows us to distinguish the uniform distribution on the set $S_a[N]$ from the uniform distribution on the set $S_b[N]$. Given $c \in \mathbb{Z}_N^*$, the test over the public key a is defined as the Jacobi symbol of $c^2 - 4a$ over N, that is:

$$GT(a, c, N) = \left(\frac{c^2 - 4a}{N}\right)$$

Notice that when c is sampled from $S_a[N]$, the test $GT(a, c, N)$ will return $+1$ with overwhelming probability given that $c^2 - 4a = (t - (a/t))^2$ is a square. However, if c is sampled from $S_b[N]$ the test is expected to return $+1$ with probability negligibly close to $1/2$ since, in this case, the distribution of the Jacobi symbol of the element $c^2 - 4a$ in $\mathbb{Z}_N^*$ follows the uniform distribution on $\{-1, +1\}$.

It is mentioned in [BCOP04] that since Cocks ciphertext is composed of several values sampled from either $S_a[N]$ (and $S_{-a}[N]$) or $S_b[N]$ (and $S_{-b}[N]$, respectively), then an adversary can repeatedly apply Galbraith's test to determine with overwhelming probability whether a given ciphertext is intended for a or b. However, one must first prove some meaningful results about the distribution of Jacobi symbols of elements of the form $c^2 - 4b$ in $\mathbb{Z}_N^*$, for *fixed* random elements $a, b \in \mathbb{Z}_N^*[+1]$ and for $c \in S_a[N]$. These results are reported in the next section.

2.2 Relevant Lemmata and Remarks

Damgård in [Dam90] studied the distribution of Jacobi symbols of elements in $\mathbb{Z}_N^*$ in order to build pseudo-random number generators. In his paper, Damgård reports of a study performed in the 50s by Perron in which it is proven that for a prime p and for any a, the set $a + \mathbb{QR}(p)$ contains as many squares as non squares in $\mathbb{Z}_p^*$ when $p \equiv 1 \mod 4$, or the difference is just 1 when $p \equiv 3 \mod 4$. It is possible to generalize Perron's result to study the properties of the set $a + \mathbb{QR}(N)$ in $\mathbb{Z}_N^*$ but we also point out that the security of Cocks IBE implicitly depends on the following Lemma:

Lemma 2.2.1. *Let (a, N) be a pair such that $(N, p, q) \xleftarrow{R} Gen(1^n)$ and $a \xleftarrow{R} \mathbb{Z}_N^*[+1]$. The distribution $\left\{ \left(\frac{t^2+a}{N} \right) : t \xleftarrow{R} \mathbb{Z}_N^* \right\}$ is computationally indistinguishable from the uniform distribution on $\{-1, +1\}$ under the quadratic residuosity assumption.*

To prove the Lemma above it is enough to observe that if we compute the Jacobi symbol of a value $c \in S_a[N]$ we obtain:

$$\left(\frac{c}{N}\right) = \left(\frac{(t^2+a)/t}{N}\right) = \left(\frac{t^2+a}{N}\right)\left(\frac{t}{N}\right)$$

However, the Jacobi symbol of t over N is the plaintext in Cocks IBE and thus Lemma 2.2.1 must follow otherwise the CPA-security of Cocks IBE would not hold.

Remark. Let's pick c uniformly at random from $\mathbb{Z}_N^*$. If $GT(a, c, N) = -1$, we can clearly conclude that $c \notin S_a[N]$. However, if $GT(a, c, N) = +1$, what is the probability that $c \in S_a[N]$? The answer is $1/2$ since a t exists such that $c = t + a/t$ whenever $c^2 - 4a$ is a square and this happens only half of the times. Clearly $GT(a, c, N)$ is equal to 0 with negligible probability hence we do not consider this case. To summarize:

$$GT(a, c, N) = \begin{cases} +1 \Longrightarrow c \in S_a[N] \text{ with prob. } 1/2 \\ -1 \Longrightarrow c \notin S_a[N] \end{cases}$$

We will argue that there is no *better* test against anonymity over an encrypted bit. That is, we show that a test that returns $+1$ to imply that $c \in S_a[N]$ with probability $1/2 + \delta$ (for a non-negligible $\delta > 0$) cannot exist under the quadratic residuosity assumption. We first notice that $c \in S_a[N]$ if and only if $\Delta = c^2 - 4a$ is a square in $\mathbb{Z}_N$. Indeed, if $c = t + a/t$ then $\Delta = (t - a/t)^2$. If Δ is a square[2] then the quadratic equation $c = t + a/t$ has solutions for t in $\mathbb{Z}_N^*$. Thus $S_a[N]$ can alternatively be defined as the set of all $c \in \mathbb{Z}_N^*$ such that $c^2 - 4a$ is a square in $\mathbb{Z}_N$.

[2] If $\Delta \in \mathbb{Z}_N \setminus \mathbb{Z}_N^*$ then Δ has one or two square roots but this happens with negligible probability.

Intuitively, we can see Galbraith's test as an algorithm that checks whether the discriminant Δ has Jacobi symbol $+1$ or -1, and this is clearly *the best it can do* since the factors of the modulus N are unknown. (Recall that we do not consider cases where the Jacobi symbol is 0 since they occur with negligible probability.) Indeed, if $\pm x$ and $\pm y$ are the four distinct square roots modulo N of Δ, then $t^2 - ct + a$ is congruent to 0 modulo N whenever t is congruent modulo N to any of the following four distinct values:

$$\frac{c \pm x}{2} \text{ and } \frac{c \pm y}{2}$$

We denote with $GT_a^N[+1]$ the set $\{c \in \mathbb{Z}_N^* \mid GT(a, c, N) = +1\}$. Analogously, we define $GT_a^N[-1]$ as the set $\{c \in \mathbb{Z}_N^* \mid GT(a, c, N) = -1\}$. We prove the following Lemma:

Lemma 2.2.2. [VQR–Variable Quadratic Residuosity] *The two distributions* $D_0(n) = \left\{(a, c, N) : (N, p, q) \xleftarrow{R} Gen(1^n),\ a \xleftarrow{R} \mathbb{Z}_N^*[+1],\ c \xleftarrow{R} S_a[N]\right\}$ *and* $D_1(n) = \left\{(a, c, N) : (N, p, q) \xleftarrow{R} Gen(1^n),\ a \xleftarrow{R} \mathbb{Z}_N^*[+1],\ c \xleftarrow{R} GT_a^N[+1] \setminus S_a[N]\right\}$ *are computationally indistinguishable under the quadratic residuosity assumption.*

Proof. We assume there is a PPT adversary $\mathcal{A}$ that can distinguish between $D_0(n)$ and $D_1(n)$ with non-negligible advantage and we use it to solve a random instance of the quadratic residuosity problem. We make no assumptions on how $\mathcal{A}$ operates and we evaluate it via oracle access.

The simulator is given a random tuple (N, x) where $(N, p, q) \xleftarrow{R} Gen(1^n)$ and $x \xleftarrow{R} \mathbb{Z}_N^*[+1]$. The simulation proceeds as follows:

1. Find a random $r \in \mathbb{Z}_N^*$ such that $a = (r^2 - x)/4$ has Jacobi symbol $+1$ (see Lemma 2.2.1). $\mathcal{A}$ receives as input (a, r, N), where $a \in \mathbb{Z}_N^*[+1]$ and $r \in GT_a^N[+1]$. Notice that releasing the public key a effectively provides $\mathcal{A}$ with the ability to generate several values in $S_a[N]$ (and $S_{-a}[N]$);
2. If $\mathcal{A}$ responds that $r \in S_a[N]$ then output *"x is a square"* otherwise output *"x is not a square"*.

The value a is distributed properly. Indeed, if $x = r^2 - 4a$ then about half of the values that x assumes for all $r \in \mathbb{Z}_N^*$ have Jacobi symbol $+1$ (thanks to Lemma 2.2.1). Let $\mathsf{Pairs}_a = \{(r^2, x) \mid x \in \mathbb{Z}_N^*[+1], r \in \mathbb{Z}_N^*, a = (r^2 - x)/4\}$. For any random $a' \xleftarrow{R} \mathbb{Z}_N^*[+1]$, with $a' \neq a$, the sets Pairs_a and $\mathsf{Pairs}_{a'}$ have about the same cardinality and their intersection is empty. Thus, the pair (r^2, x) provided by the simulator represents a random pair from the union over a of all sets Pairs_a, and therefore the value a is uniformly distributed to a PPT adversary.

It has already been established that $r \in S_a[N]$ if and only if $r^2 - 4a$ is a square. But $r^2 - 4a = x$, therefore $\mathcal{A}$ cannot have non-negligible advantage under the quadratic residuosity assumption.

□

The next Lemma easily follows from Lemma 2.2.1 since $c^2 - 4a$ can be written as $c^2 + h$ for a fixed $h \in \mathbb{Z}_N^*[+1]$.

Lemma 2.2.3. *Let (a, N) be a pair such that $(N, p, q) \xleftarrow{R} Gen(1^n)$ and $a \xleftarrow{R} \mathbb{Z}_N^*[+1]$. The distribution $\left\{GT(a, c, N) : c \xleftarrow{R} \mathbb{Z}_N^*\right\}$ is computationally indistinguishable from the uniform distribution on $\{-1, +1\}$.*

Remark. Thanks to Lemma 2.2.2, we can construct a first-attempt anonymous scheme. Let (C_1, C_2) be a Cocks IBE ciphertext of t bits for public key a, i.e., C_1 contains t values in $S_a[N]$ and C_2 contains t values in $S_{-a}[N]$. Proceed as follows: Generate a sequence of $4 \cdot t$ random values in $\mathbb{Z}_N^*$ for a, (2) individuate the first t elements in the sequence whose Galbraith's test under a returns $+1$ and substitute those in the sequence with the values in C_1, (3) finally, repeat the process for $-a$ and the ciphertext C_2. Although the scheme is anonymous, it is not efficient. Indeed, one must send $8t$ elements in $\mathbb{Z}_N^*$ for t bits, while Cocks IBE sends only $2t$ elements. In the next section we provide a substantially better solution.

2.3 Our Basic Construction and Its Efficient Variants

We extend Cocks' scheme to support anonymity. Unlike previous proposals, our scheme UAnonIBE has efficiency, storage, and bandwidth requirements similar to those of the original scheme by Cocks (which is not anonymous). Our scheme is also the first universally anonymous IBE, according to the definition in [HT05] (although we do not include the extra algorithms as in [HT05] to keep the presentation simple).

2.3.1 The Basic Scheme

Let $H : \{0, 1\}^* \to \mathbb{Z}_N^*[+1]$ be a full-domain hash modeled as a random oracle. Let n and m be two security parameters (we assume w.l.o.g. that m depends on n). The algorithms which form UAnonIBE are defined as follows (all operations are performed modulo N):

Master Key: The public key of the trusted authority is the n-bit Blum integer $N = pq$, where p and q are $n/2$-bit primes each congruent to 3 modulo 4.

Key Generation: Given the identity ID, the authority generates $a = H(ID)$ (thus the Jacobi symbol $\left(\frac{a}{N}\right)$ is $+1$). The secret key for the identity ID is a value r randomly chosen in $\mathbb{Z}_N^*$ such that $r^2 \equiv a \bmod N$ or $r^2 \equiv -a \bmod N$. This value r is stored and returned systematically.

Encryption: To encrypt a bit $b \in \{-1, +1\}$ for identity ID, choose uniformly at random two values $t, v \in \mathbb{Z}_N^*$, such that $\left(\frac{t}{N}\right) = \left(\frac{v}{N}\right) = b$, and compute $(c, d) = \left(t + \frac{a}{t}\ , v - \frac{a}{v}\right)$.

Then, compute the *mask* to anonymize the ciphertext (c, d) as follows:

1. Pick two indices k_1 and k_2 independently from the geometric distribution[3] D with probability parameter $1/2$;
2. Choose $T, V \xleftarrow{R} \mathbb{Z}_N^*$ independently and uniformly at random and set $Z_1 = c + T$ and $Z_2 = d + V$;
3. For $1 \leq i < k_1$, choose $T_i \xleftarrow{R} \mathbb{Z}_N^*$ independently and uniformly at random such that $GT(a, Z_1 - T_i, N) = -1$;
4. For $1 \leq i < k_2$, choose $V_i \xleftarrow{R} \mathbb{Z}_N^*$ independently and uniformly at random such that $GT(-a, Z_2 - V_i, N) = -1$;
5. Set $T_{k_1} = T$ and $V_{k_2} = V$;
6. For $k_1 < i \leq m$, choose $T_i \xleftarrow{R} \mathbb{Z}_N^*$ independently and uniformly at random;
7. For $k_2 < i \leq m$, choose $V_i \xleftarrow{R} \mathbb{Z}_N^*$ independently and uniformly at random;

Finally, output $(Z_1, T_1, \ldots, T_m) \in (\mathbb{Z}_N^*)^{m+1}$ and $(Z_2, V_1, \ldots, V_m) \in (\mathbb{Z}_N^*)^{m+1}$.[4]

Decryption: Given a ciphertext $(Z_1, T_1, \ldots, T_m)$ and $(Z_2, V_1, \ldots, V_m)$, first discard one of the two tuples based on whether a or $-a$ is a square. Let's assume we keep the tuple $(Z_1, T_1, \ldots, T_m)$ and we discard the other. In order to decrypt, find the smallest index $1 \leq i \leq m$ s.t. $GT(a, Z_1 - T_i, N) = +1$ and output:

$$\left(\frac{Z_1 - T_i + 2r}{N}\right) = b$$

[3]The geometric distribution is a discrete memoryless random distribution for $k = 1, 2, 3, \ldots$ having probability function $\Pr[k] = p(1-p)^{k-1}$ where $0 < p < 1$. Therefore, for $p = 1/2$ the probability that $k_1 = k$ is 2^{-k}. For more details see, e.g., [Spi92].

[4]Note that if we build a sequence $c_1, \ldots, c_m$ by selecting a k from D and setting $c_i = -1$ for $1 \leq i < k$, $c_k = 1$, and random $c_i \xleftarrow{R} \{-1, 1\}$ for $k < i \leq m$, we have that $\Pr[c_i = 1] = 2^{-i} + \sum_{j=2}^{i} 2^{-j} = 1/2$.

We run the same procedure above if the second tuple is actually selected and the first is discarded. It is enough to replace a with $-a$, Z_1 with Z_2, and T_i with V_i.

2.3.2 Security Analysis

We need to show that our scheme, UAnonIBE, is ANON-IND-ID-CPA-secure [ABC+05, BGH07], that is, the ciphertext does not reveal any information about the plaintext and an adversary cannot determine the identity under which an encryption is computed, even thought the adversary selects the identities and the plaintext.

In [Hal05], Halevi provides a sufficient condition for a CPA public-key encryption scheme to meet the notion of key-privacy, or anonymity, as defined by Bellare et al. in [BBDP01]. In [ABC+05], Abdalla et al. extend Halevi's condition to identity-based encryption. In addition, their notion is defined within the random oracle model and Halevi's statistical requirement is weakened to a computational one. Informally, it was observed that if an IBE scheme is already IND-ID-CPA-secure then the challenger does not have to encrypt the message chosen by the adversary but can encrypt a random message of the same length. The game where the challenger replies with an encryption on a random message is called ANON-RE-CPA. In [ABC+05], it was shown that if a scheme is IND-ID-CPA-secure and ANON-RE-CPA-secure then it is also ANON-IND-ID-CPA-secure.

ANON-RE-CPA game. We briefly describe the security game introduced by Abdalla et al. in [ABC+05]. MPK represents the set of public parameters of the trusted authority. The adversary $\mathcal{A}$ has access to a random oracle H and to an oracle KeyDer that given an identity ID returns the private key for ID according to the IBE scheme.

Experiment $\mathsf{Exp}^{\mathsf{anon-re-cpa}}_{\mathsf{IBE},\mathcal{A}}(n)$:

pick random oracle H;

$(ID_0, ID_1, msg, state) \xleftarrow{R} \mathcal{A}^{\mathsf{KeyDer}(\cdot),H}(\mathtt{find}, MPK)$;

$b \xleftarrow{R} \{0,1\}$;

$W \xleftarrow{R} \{0,1\}^{|msg|}; c \xleftarrow{R} \mathsf{Enc}^H(MPK, ID_b, W)$;

$b' \xleftarrow{R} \mathcal{A}^{\mathsf{KeyDer}(\cdot),H}(\mathtt{guess}, c, state)$;

return 1 if $b' = b$, 0 otherwise.

The adversary cannot request the private key for ID_0 or ID_1 and the message msg must be in the message space associated with the scheme. A scheme is said to be ANON-RE-CPA-secure if for all polynomial time adversaries $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\Pr[\mathsf{Exp}^{\mathsf{anon-re-cpa}}_{\mathsf{IBE},\mathcal{A}}(n) = 1] \leq 1/2 + \mathsf{negl}(n)$$

We define the function $\mathsf{Adv}^{\mathsf{anon-re-cpa}}_{\mathsf{UAnonIBE},\mathcal{A}}(\cdot)$ as the advantage of the adversary $\mathcal{A}$ against UAnonibe, that is:

$$\mathsf{Adv}^{\mathsf{anon-re-cpa}}_{\mathsf{UAnonIBE},\mathcal{A}}(n) \stackrel{\mathsf{def}}{=} \Pr[\mathsf{Exp}^{\mathsf{anon-re-cpa}}_{\mathsf{UAnonIBE},\mathcal{A}}(n) = 1] - 1/2$$

Note that the algorithm Enc has access to the random oracle H even though H is not used to construct the challenge c. In our basic scheme, H is used only to generate public keys from identity strings.

Theorem 2.3.1. UAnonIBE *is* ANON-IND-ID-CPA-*secure in the random oracle model under the quadratic residuosity assumption.*

Proof of theorem 2.3.1. It must be clear that UAnonIBE is IND-ID-CPA-secure since Cocks IBE is IND-ID-CPA-secure in the random oracle model under the quadratic residuosity assumption and the mask is computed without knowing the plaintext or the secret key of the intended recipient. Thus, we only need to show that UAnonIBE is ANON-RE-CPA-secure.

In order to simplify the proof of theorem 2.3.1, we present it as a sequence of experiments which we call *games* following the standard literature. Let win_i denote the event that the adversary $\mathcal{A}$ wins in game i.

Game 0: This game is identical to the ANON-RE-CPA experiment defined above, where the adversary $\mathcal{A}$ chooses two identities ID_0 and ID_1 and receives two sequences $(Z_1, T_1, \ldots, T_m) \in (\mathbb{Z}_N^*)^{m+1}$ and $(Z_2, V_1, \ldots, V_m) \in (\mathbb{Z}_N^*)^{m+1}$. Hence, by definition, we have that:

$$\Pr[\mathsf{win}_0] - 1/2 = \mathsf{Adv}^{\mathsf{anon-re-cpa}}_{\mathsf{UAnonIBE},\mathcal{A}}(n)$$

In the rest of the proof, we consider w.l.o.g. only the first sequence of the challenge ciphertext, that is the sequence $(Z_1, T_1, \ldots, T_m)$. To prove that this is indeed w.l.o.g. notice that the second sequence $(Z_2, V_1, \ldots, V_m)$ is completely independent from the first one. In particular, the parameters and elements used to build the second sequence are generated independently from those used to build the first one, except that both sequences encrypt the same message. However, since the encryption is CPA-secure, this affects the advantage of the adversary only negligibly.

Game 1: In this game, we modify the way the challenger constructs the ciphertext. In particular, first set $a_0 = H(ID_0)$ and $a_1 = H(ID_1)$. Choose a random bit $b \xleftarrow{R} \{0,1\}$. Then, a sequence $(v_1, \ldots, v_m) \in (\mathbb{Z}_N^*)^m$ is built as follows: (1) Select an index k according to the geometric distribution D with parameter $1/2$. (2) Choose an element c_b uniformly at random from $S_{a_b}[N]$ and set $v_k = c_b$. (3) Choose elements $v_1, \ldots, v_{k-1}$ independently and uniformly at random from $GT_{a_b}^N[-1]$ and elements $v_{k+1}, \ldots, v_m$ from $\mathbb{Z}_N^*$.

The challenger chooses Z_1 uniformly at random from $\mathbb{Z}_N^*$ and sets $T_i = Z_1 - v_i \bmod N$, for $1 \leq i \leq m$. The challenge ciphertext is $(Z_1, T_1, \ldots, T_m)$.

The adversary $\mathcal{A}$ cannot distinguish Game 1 from Game 0 since the index k is chosen from the same distribution and the values Z_1 and T_i, for $1 \leq i \leq m$, in Game 1 are distributed as in Game 0 with overwhelming probability. Therefore, there exists a negligible function $\mathsf{negl}_0(\cdot)$ such that:

$$|\Pr[\mathsf{win}_0] - \Pr[\mathsf{win}_1]| \leq \mathsf{negl}_0(n)$$

Game 2: In this game, we make only a minor change to Game 1. In particular, the challenger chooses c_b uniformly at random from $GT_{a_b}^N[+1]$ (and not from $S_{a_b}[N]$ as in Game 1). The rest of the construction is identical to the one in Game 1.

Thanks to Lemma 2.2.2 (VQR), the adversary $\mathcal{A}$ cannot distinguish Game 2 from Game 1. Namely, there exists a negligible function $\mathsf{negl}_1(\cdot)$ such that:

$$|\Pr[\mathsf{win}_1] - \Pr[\mathsf{win}_2]| \leq \mathsf{negl}_1(n)$$

Game 3: This game proceeds exactly as Game 2 except that the sequence $(v_1, \ldots, v_m)$ is now chosen uniformly at random from $(\mathbb{Z}_N^*)^m$ (i.e., steps (1) through (3) described in Game 1 and in Game 2 are not performed and each v_i is independently and uniformly distributed in $\mathbb{Z}_N^*$).

It is easy to show that Game 3 is indistinguishable from Game 2. Indeed, the only difference between the two games is in the way the sequence $(v_1, \ldots, v_m) \in (\mathbb{Z}_N^*)^m$ is built. In Game 2, an index k is picked according to the geometric distribution with parameter $1/2$, v_k is uniformly distributed in $GT_{a_b}^N[+1]$, and the elements before and after v_k in the sequence are independently and uniformly distributed respectively in $GT_{a_b}^N[-1]$ and $\mathbb{Z}_N^*$. In Game 3, each v_i is chosen independently and uniformly at random from $\mathbb{Z}_N^*$. The adversary $\mathcal{A}$ will not detect any difference between these two distributions thanks to Lemma 2.2.3. Namely, we know that $\Pr[GT(a_b, v_i, N) = +1]$ is negligibly close to $1/2$ whether v_i is from the distribution in Game 2 or from the one in Game 3. Therefore, there exists a negligible function $\mathsf{negl}_2(\cdot)$ such that:

$$|\Pr[\mathsf{win}_2] - \Pr[\mathsf{win}_3]| \leq \mathsf{negl}_2(n)$$

Combining the probabilities in all games, we have that there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\mathsf{Adv}^{\mathsf{anon-re-cpa}}_{\mathsf{UAnonIBE},\mathcal{A}}(n) = \Pr[\mathsf{win}_0] - 1/2 \leq \Pr[\mathsf{win}_3] - 1/2 + \mathsf{negl}(n)$$

To conclude the proof we must show that $\Pr[\mathsf{win}_3]$ is exactly $1/2$. This follows simply because the challenge ciphertext $(Z_1, T_1, \ldots, T_m)$ in Game 3 is independent of the bit b. Indeed, $Z_1 \leftarrow \mathbb{Z}_N^*$ and $T_i = Z_1 - v_i \in \mathbb{Z}_N^*$ where the v_i's are independently and uniformly distributed in $\mathbb{Z}_N^*$ (recall that, as in the rest of the chapter, we explicitly use $\mathbb{Z}_N^*$ rather than $\mathbb{Z}_N$ since elements uniformly distributed in $\mathbb{Z}_N$ are in $\mathbb{Z}_N^*$ with overwhelming probability.) □

2.3.3 A First Efficient Variant: Reducing Ciphertext Expansion

The obvious drawback of the basic scheme is its ciphertext expansion. Indeed, for each bit of the plaintext $2 \cdot (m+1)$ values in $\mathbb{Z}_N^*$ must be sent while in Cocks IBE each bit of the plaintext requires two values in $\mathbb{Z}_N^*$. Therefore, we need a total of $2 \cdot (m+1) \cdot n$ bits for a single bit in the plaintext, where n and m are the security parameters (e.g., $n = 1024$ and $m = 128$). This issue, however, is easy to fix. Intuitively, since our scheme requires the random oracle model for its security, we could use another random oracle that expands a short seed into a value selected uniformly and independently in $\mathbb{Z}_N^*$. Specifically, a function $G : \{0,1\}^* \to \mathbb{Z}_N^*$ is used, which we model as a random oracle, that maps a e-bit string α to a random value in $\mathbb{Z}_N^*$. The parameter e must be large enough, e.g., $e = 160$.

It is tempting to use the oracle G and a single short seed α plus a counter to generate all values $T_1, \ldots, T_m$ and $V_1, \ldots, V_m$. This first solution would provide minimal ciphertext expansion, since only the seed α must be sent, however it may turn out to be computationally expensive. To see this, consider that an α must be found such that $GT(a, Z_1 - T_i, N) = -1$ for $1 \leq i < k_1$. Now, if k_1 happens to be large, say $k_1 = 20$, then clearly finding a suitable α could be computationally intensive. Nevertheless, we prove that this scheme is secure as long as the basic $\mathsf{UAnonIBE}$ scheme is secure. More importantly, we emphasize that the proof of security of all other schemes proposed after this first one can easily be derived from the proof of the following theorem.

Theorem 2.3.2. *The first efficient variant of* $\mathsf{UAnonIBE}$ *is* ANON-IND-ID-CPA*-secure in the random oracle model under the quadratic residuosity assumption.*

Proof of theorem 2.3.2. We let the simulator $\mathcal{S}$ play the role of a man-in-the-middle attacker between two ANON-RE-CPA games: the first game is against the basic UAnonIBE and the second game is against an adversary $\mathcal{A}$ that has non-negligible advantage in breaking the first variant of UAnonIBE. We show that $\mathcal{S}$ can use $\mathcal{A}$ to win in the first ANON-RE-CPA game, thus violating the quadratic residuosity assumption. The simulation is straightforward: $\mathcal{S}$ forwards the H-queries and KeyDer-queries to the respective oracles. When $\mathcal{A}$ challenges for identities ID_0 and ID_1, $\mathcal{S}$ challenges on the same identities in the first ANON-RE-CPA game. Then $\mathcal{S}$ receives the ciphertext $(Z_1, T_1, \ldots, T_m)$, $(Z_2, V_1, \ldots, V_m)$. $\mathcal{S}$ sends $(Z_1, \alpha), (Z_2, \beta)$ to $\mathcal{A}$, where α and β are chosen uniformly at random in $\{0,1\}^e$. At this point, the simulator responds to the G-queries as follows:

$$G(\alpha \mid\mid i) = T_i \text{ and } G(\beta \mid\mid i) = V_i \text{ , for } 1 \leq i \leq m,$$

and with random values in $\mathbb{Z}_N^*$ in any other cases. The adversary $\mathcal{A}$ eventually returns its guess which $\mathcal{S}$ uses in the first game in order to win with non-negligible advantage. □

The obvious next-best solution is to use a single seed per value. Thus, rather than sending the ciphertext as per our basic scheme, that is $(Z_1, T_1, \ldots, T_m)$ and $(Z_2, V_1, \ldots, V_m)$, the following values could be sent:

$$(Z_1, \alpha_1, \ldots, \alpha_m) \text{ and } (Z_2, \beta_1, \ldots, \beta_m),$$

where α_i, β_i are chosen uniformly at random in $\{0,1\}^e$ until the conditions in steps 3. and 4. of the encryption algorithm of the basic scheme are satisfied. The recipient would then derive the intended ciphertext by computing $T_i = G(\alpha_i)$ and $V_i = G(\beta_i)$, for $1 \leq i \leq m$. If we set e to be large enough, say $e = 160$, then clearly the security of this variant derives from the one of the basic scheme in the random oracle model and a single bit of the plaintext would require $2 \cdot (m \cdot e + n)$ bits rather than $2 \cdot (m \cdot n + n)$, where $e < n$. Hence, for $n = 1024$, $m = 128$ and $e = 160$, we need to send $2 \cdot (160 \cdot 128 + 1024)$ bits while Cocks' scheme requires only $2 \cdot (1024)$ bits.

On a closer look, however, it is easy to see that since G is a random oracle we just need to ensure that its inputs are repeated only with negligible probability. Let $X = x^{(1)}x^{(2)} \ldots x^{(t)}$ be the plaintext of t bits. For each plaintext X, the sender selects a random message identifier $MID_X \in \{0,1\}^{e_1}$ which is sent along with the ciphertext. For bit $x^{(j)}$, the sender computes:

$$(Z_1^{(j)}, \alpha_1^{(j)}, \ldots, \alpha_m^{(j)}) \text{ and } (Z_2^{(j)}, \beta_1^{(j)}, \ldots, \beta_m^{(j)}),$$

where the coefficients $\alpha_i^{(j)}$, $\beta_i^{(j)}$ are chosen uniformly at random in $\{0,1\}^e$ until the conditions in steps 3. and 4. of the encryption algorithm of the basic scheme are satisfied

(thus notice that e can be small but still big enough to be able to find those values $T_i^{(j)}$ and $V_i^{(j)}$ that satisfy such conditions). The recipient will derive the intended ciphertext by computing:

$$T_i^{(j)} = G(MID_X \,||\, 0 \,||\, \alpha_i^{(j)} \,||\, i \,||\, j) \text{ or } V_i^{(j)} = G(MID_X \,||\, 1 \,||\, \beta_i^{(j)} \,||\, i \,||\, j),$$

where $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, t\}$. As an example, we can set $m = 128$, $e_1 = 160$, and $e = 8$. In this case the ciphertext expansion per single bit of the plaintext is only $2 \cdot (1024 + 1024)$ bits which is twice the amount required by Cocks IBE for $n = 1024$. (In addition, extra 160 bits are needed for MID_X but these bits are transmitted only once per message.)

2.3.4 Trade-off Between Ciphertext Expansion and Performance

We propose a second variant of UAnonIBE which provides an optimal trade-off between efficiency and ciphertext expansion. Our performance tests show that this variant is in practice as efficient as any of the previous variants and at the same time it provides the smallest ciphertext expansion (thus we recommend this version for practical systems).

We fix a new global parameter ℓ which is a small positive integer. Let $X = x^{(1)}x^{(2)} \ldots x^{(t)}$ be the plaintext of t bits. For each plaintext X, the sender selects a random identifier $MID_X \in \{0,1\}^{e_1}$ which is sent along with the ciphertext. For bit $x^{(j)}$, the sender computes:

$$(Z_1^{(j)}, \alpha_1^{(j)}, \ldots, \alpha_\ell^{(j)}) \text{ and } (Z_2^{(j)}, \beta_1^{(j)}, \ldots, \beta_\ell^{(j)})$$

where $\alpha_i^{(j)}$, $\beta_i^{(j)}$ are in $\{0,1\}^e$, when $i < \ell$, and $\alpha_\ell^{(j)}$, $\beta_\ell^{(j)}$ are in $\{0,1\}^{e'}$, for some $e' > e$. The intended ciphertext is derived by the recipient by computing:

$$T_i^{(j)} = G(MID_X \,||\, 0 \,||\, \alpha_i^{(j)} \,||\, i \,||\, j) \text{ or } V_i^{(j)} = G(MID_X \,||\, 1 \,||\, \beta_i^{(j)} \,||\, i \,||\, j)$$

for $i < l$, and

$$T_i^{(j)} = G(MID_X \,||\, 0 \,||\, \alpha_\ell^{(j)} \,||\, i \,||\, j) \text{ or } V_i^{(j)} = G(MID_X \,||\, 1 \,||\, \beta_\ell^{(j)} \,||\, i \,||\, j)$$

for $i \geq \ell$. Note that in this variant of our basic scheme an arbitrary number of $T_i^{(j)}$ and $V_i^{(j)}$ can be generated (i.e., there is no fixed global parameter m).

Given the distribution of k_1, k_2, for a large enough ℓ, we expect $k_1 \leq \ell$ or $k_2 \leq \ell$ with high probability. When $k_1 \leq \ell$ or $k_2 \leq \ell$ the scheme is as efficient as the first variant. When $k_1 > \ell$ (or $k_2 > \ell$) the computational cost of finding a value for $\alpha_\ell^{(j)}$ (or $\beta_\ell^{(j)}$) is exponential in $k_1 - \ell$ ($k_2 - \ell$, respectively).

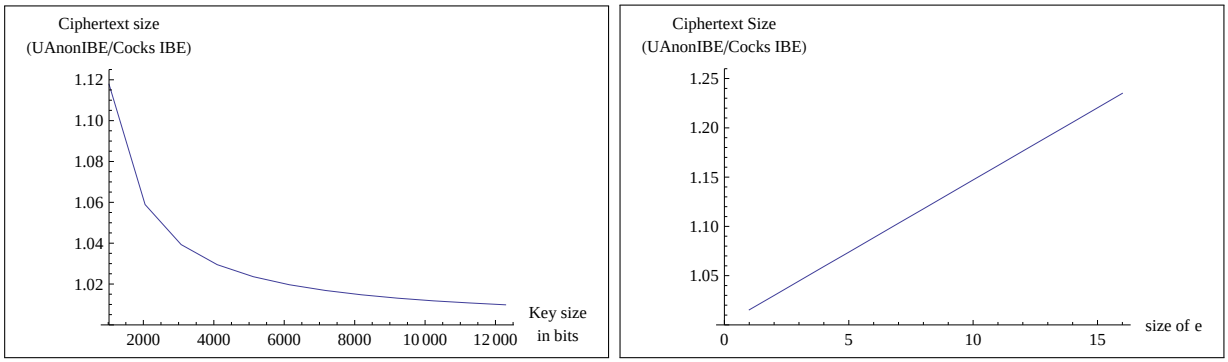

Figure 2.1: The two graphs show UAnonIBE's ciphertext size relative to Cocks' scheme. The first shows how the relative bandwidth overhead introduced by our solution decreases with the size of the master parameter, while the second shows how the size of the ciphertext increases, varying the size of e and fixing $e' = 10 \cdot e$, compared to Cocks' ciphertext.

As an example, we set the global parameter $\ell = 6$ and then $e_1 = 160$, $e = 8$, $e' = 80$, and $n = 1024$. The ciphertext expansion of this variant of UAnonIBE is $2 \cdot ((\ell - 1) \cdot e + e' + n)$, therefore, the ciphertext size for a single bit of the plaintext is now only $2 \cdot (120 + 1024)$ bits which is very close to the number of bits $(2 \cdot (1024))$ required by Cocks IBE (which is not anonymous). Note that for each message, the sender also transmits the random message identifier MID_X.

2.4 Optimizations and Implementation

An important aspect that should be considered in order to implement UAnonIBE efficiently is the value of the parameters ℓ, e (the size of $\alpha_1^{(j)}, \ldots, \alpha_{\ell-1}^{(j)}$) and e' (the size of $\alpha_\ell^{(j)}$). These values affect both the ciphertext expansion and the encryption time significantly, therefore they must be selected carefully. Choosing e or e' to be too small can reduce the probability of encrypting to an unacceptable level. Choosing ℓ to be too small can make the encryption process very slow. If we set $e = 8$ and $e' = 80$, we can find a suitable value for each $\alpha_i^{(j)}$, and therefore encrypt, with a probability of at least $1 - 2^{-80}$. We found that the value $\ell = 6$ is the best compromise between encryption time and ciphertext expansion. If we set $e = 8$, $e' = 80$ and $\ell = 6$, the ciphertext expansion for a 128-bit message is 3840 bytes more than a Cocks encryption for both $+a$ and $-a$: for a 1024-bit modulus N the encrypted message size is about 36KB instead of 32KB with Cocks IBE.

Size of e	2	4	6	8	10	Cocks IBE
Ciphertext size in bytes	33748	34708	35668	36628	37588	32768

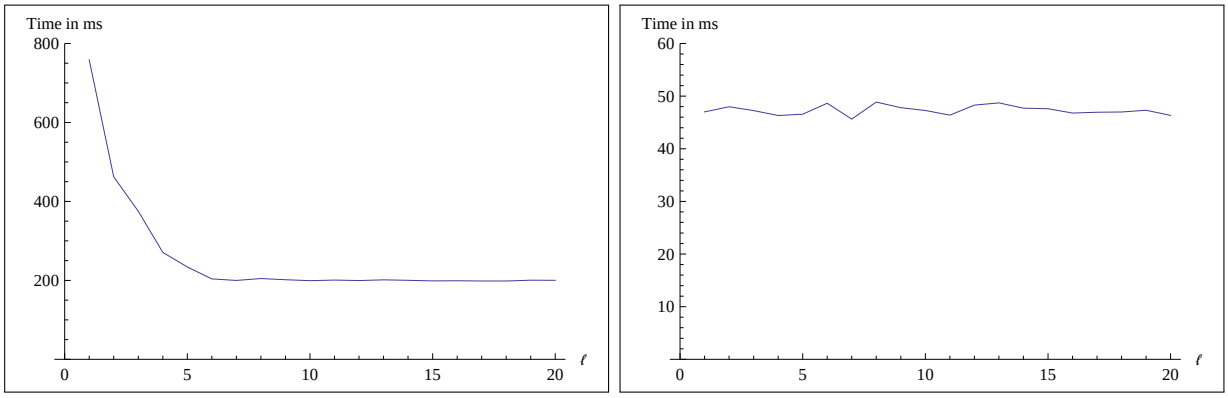

Figure 2.2: The two graphs show the average time required respectively to anonymize and de-anonymize a 128-bit message encrypted with Cocks IBE varying ℓ.

We implemented the second efficient variant of UAnonIBE and compared it with the original Cocks IBE [Coc01] and the scheme proposed by Boneh and Franklin based on pairings [BF03b]. We have two goals in mind. The first is to show that Cocks IBE and our schemes are practical when used as hybrid encryption algorithms (following the KEM-DEM paradigm), even when compared with the Boneh-Franklin IBE, with the clear advantage compared to other IBE schemes of relying on a well-established assumption. Our second goal is to show that the efficiency of our scheme is comparable to that of the original scheme by Cocks.

For our performance analysis, we set the size of the values $\alpha_i^{(j)}$ and $\beta_i^{(j)}$ with $1 \leq i < \ell$ to 8 bits and the size of $\alpha_\ell^{(j)}$ and $\beta_\ell^{(j)}$ to 80 bits. However, our tests showed that the size of those parameters have no measurable impact on the performance of the scheme. In order to calculate the optimal value for ℓ, we measured the time required to anonymize a key of 128 bit (for a total of 256 encrypted bits, considering both cases $+a$ and $-a$). Figure 2 summarizes our results. The value $\ell = 6$ seems to be optimal, since further increasing ℓ does not noticeably affect the time required to anonymize or decrypt a message.

Experimental Setup. We employed the MIRACL software package, developed by Shamus Software [Sha], to run our tests. MIRACL is a comprehensive library often used to implement cryptographic systems based on pairings. We used the optimized implementation of the Boneh-Franklin IBE provided by the library and we implemented Cocks IBE and our scheme with an RSA modulus of 1024 bits. The implementation of the Boneh-Franklin IBE uses a 512-bit prime p, Tate pairing and a small 160-bit subgroup q. The curve used is $y^2 = x^3 + x$ instead of $y^2 = x^3 + 1$ because it allows for a faster implementation. Those two settings should provide the same level of security according to NIST [NISb]. The tests were run on a machine that consisted of an Intel Pentium 4 2.8GHz with 512MB RAM. The system was running Linux kernel 2.6.20 with the compiler GCC 4.1.2.

We implemented the cryptographic primitives using version 5.2.3 of the MIRACL library. Every source file was compiled with optimization '-02', as suggested in the MIRACL documentation. The table below shows average times over 1000 runs of the cryptographic operations specified in the first row. The symmetric key encrypted in our tests is of 128 bits.

	Extract	Encrypt	Decrypt	Anonymous	Universally Anonymous
Boneh-Franklin	9.1 ms	91.6 ms	85.4 ms	YES	NO
Cocks IBE	14.2 ms	115.3 ms	35.0 ms	NO	NO
Our Scheme	14.2 ms	319.4 ms	78.1 ms	YES	YES

In the table we also indicate whether a scheme is anonymous or not. Cocks IBE is not anonymous while Boneh-Franklin IBE is anonymous but not universally anonymous.

Pre-computation. All three schemes can use pre-computation to significantly speed up encryption. The implementation of Boneh-Franklin scheme provided by MIRACL uses an optimization proposed by Brickell et al. [BGMW92] that creates a small table of integers in order to speed up exponentiations with a fixed base. The performance of our scheme and Cocks IBE can be improved by pre-computing the random values t_i's and their inverses. These are the values that encode the bits of the plaintext. It is possible to create two sets, one with random elements in $\mathbb{Z}_N^*[+1]$ and the other with random elements in $\mathbb{Z}_N^*[-1]$. Once the identity is known, the encryption is quite fast as it requires the computation of a modular multiplication and a sum.

Clearly our scheme and Cocks IBE are predisposed to parallelization and could take advantage of multiprocessor / multicore systems. Indeed in both UAnonIBE and Cocks IBE every bit is encrypted independently on each other and each encryption for a is independent from the encryption computed for $-a$. In addition we notice that our scheme and Cocks IBE when used as KEM are much faster than when used to encrypt standard messages: there is no need to discard random values whose Jacobi symbol does not match the bit of the plaintext. We just set the bit of the session key as the Jacobi symbol of the random value generated, thus we cut the cost in half on average.

2.5 A Toy Example: A Plugin for an Email Client

The general lack of interest in the use of encrypted e-mail is principally due to the intrinsic complexity of employing public key cryptography, which clearly overweights the perceived usefulness of message confidentiality; users must generate keys and retrieve public-keys of others before encrypting messages. Identity-based encryption can be seen as a valid

alternative to public-key encryption. Indeed, messages can be encrypted directly under the email address of the recipients.

Identity-based encryption provides an elegant solution to this problem by making the cryptographic aspects of the communication almost transparent to the user. All the parameters are set once for all the system by an authority, leveraging the user from the need to understand the underlying complexity. A user willing to send an encrypted message does not need to even know if the recipient already created a key pair, because it can calculate the receiver's public key independently and off line once obtained the master parameter from the authority.

Figure 2.3 illustrates how a ciphertext encrypted with our command line tool appears to the recipient when it cannot be decrypted by the email software. The attached instructions should help the user to retrieve its private key and to download and install the decryption software.

```
-----BEGIN ENCRYPTED MESSAGE-----
Note: You see this message because your email
Note: software was unable to decrypt the content
Note: of this email.
Note: If you haven't request your private key
Note: yet, please click on the link below

Link: http://www.authority.com/requestkey.php

ARgAgF5EaJks7XMi+DvS5CKRFckpV4+DdQuRoowhnx3O8WgLwQ
3tVyg5EdzWfQ+ZiulEmZV9/c6NUhWYnHP61u7rATahl02w6/c4
8PLO8wufNe5UgKv7MyY/0q6GfK7jEtWCkX6rjGo1re557P+PvS
yS0kKSr3wYJeQSi2pidz3rZdQ9KOG44+ElVGN5YfhaZwx9Ez4e
[...omissis...]
ASFnfg==
-----END ENCRYPTED MESSAGE-----
```

Figure 2.3: Example message encrypted with the command line utility implementing the Final Scheme

We implemented our anonymous IBE as command line encryption utility. This utility takes a set of identities as input and a working mode (encryption / decryption) and accepts the plaintext (or ciphertext, if working in decryption mode) from the standard input or from a file. It performs encryption and decryption operations according to the specifics of our scheme and the result is redirected to the standard output (see Figure 2.4). Our scheme was re-implemented using the OpenSSL library instead of MIRACL mostly because of its

wide availability. OpenSSL is a well-known open source cryptographic toolkit, based on the SSLeay library. OpenSSL implements several cryptographic primitives and provides an interface for arbitrary sized integers, which we used to implement UAnonIBE.

```
usage: UAnonIBE -e -r ID1...IDn [-in fname] [-out fname]
        -d [-in fname] [-out fname]
```

Figure 2.4: Summary of the option list of the IBE command line utility

We also developed a security extension to Mozilla Thunderbird in order to integrate our IBE implementation with a popular open source e-mail client. This extension works as a bridge between the command line utility, used to encrypt and decrypt messages, and the e-mail client. It provides an easy to use interface which guides the user through all the required steps to encrypt or decrypt a message.

In order to encrypt a email, the user clicks on the "encrypt" icon in the composer window toolbar. A green icon on the bottom right corner of the same window confirms to the user that the email is going to be encrypted before it is sent to the recipients. Then the user can go on typing the message and press the "send" button as usual once the email is ready. At this point the plugin sends the message to the standard input of the command line encryption utility, specifying the recipient's identities as parameter. The encrypted message is returned and read by the plugin as the output of the command line tool.

We believe that this small prototype demonstrates how the use of identity-based encryption provides an incentive for the use of strong cryptography among the general public due to its intrinsic ease of use compared to traditional public-key encryption.

Chapter 3

Practical Public-Key Deniable Encryption

Public-key deniable encryption allows the sender and/or the receiver of an encrypted message to open the ciphertext is such a way as to reveal a different version of the plaintext than the one originally used in producing the ciphertext, without the need of any shared secret between them.

Deniable encryption, as introduced in [CDNO97], can be divided into two types: *plan-ahead* encryption and *ad-hoc* encryption. Plan-ahead deniable encryption requires the sender to choose both the real and fake messages at the time of encryption. Ad-hoc deniable encryption, on the other hand, allows the sender to encrypt a real message, and the fake message can be chosen at the time of coercion (from the entire message space). Currently, the only known constructions for ad-hoc deniable encryption correspond to the public-key setting and achieve sender-deniability.[1] Such constructions are expensive in the sense that they require one (public-key) ciphertext per message bit [CDNO97, KKK08, Ibr09a], or allow several bits to be transmitted in a single ciphertext, but the ciphertext size is still exponential in the number of message bits transmitted [Ibr09a]. Because our goal is to seek a practical solution, we concentrate on plan-ahead deniability.

Considering that certain encryption algorithms are well-known, widely accepted and used in practice, it would be desirable to use such algorithms, if possible, instead of custom-built schemes to achieve wider adoption of deniable encryption. Since the conventional encryption schemes are not deniable, the existing deniable encryption schemes do not use standard tools.[2] Our goal, however, is to explore what deniability properties can be

[1] The work of Ibrahim [Ibr09b] is an exception; see Chapter 1 for more information.

[2] One of the schemes given in Klonowski et al. [KKK08] is built using ElGamal, but requires the receiver to share her private key with the sender, see Chapter 1 for more information.

achieved with conventional encryption schemes which are in use today.

The original motivation for deniable encryption stems from the scenario where an adversary captures communications and forces the sender and/or receiver to open the ciphertext. We, however, view one of the most prominent uses of deniable encryption the case where encrypted contents are stored on a computer (e.g., a laptop) and the adversary forces the owner of the machine to unlock the encrypted contents. In this case, receiver deniability is desired and the entire disk contents can be examined by the adversary. This means that the only information that can stay hidden from the adversary is the secrets the machine owner can remember, but does not reveal to the coercer. Note that existing deniable encryption schemes are not necessarily satisfactory in this context, as to be able to open a ciphertext to a value different from what it encrypts, one might have to store additional information associated with that ciphertext (e.g., data used in generating it). Once an adversary obtains access to this additional information stored on the disk, the adversary is likely to be able to use it to uncover the original message. We first describe our solution using the conventional model of deniable encryption, and treat this case in section 6.2.

Existing public-key constructions are either flexible (i.e., ad-hoc), but inefficient (i.e., can communicate only a very small number of bits per ciphertext) or are able to communicate several secret bits and use standard tools, but require the recipient to give up her private key. Furthermore, all suitable solutions are sender-deniable and therefore are not applicable to the problem of deniable storage. The motivation of the work detailed in this chapter comes from a usability perspective: can we design an efficient solution that does not require revealing the receiver's private key? Can we achieve receiver-deniability without using mediated settings?

Following the work of Klonowski et al. [KKK08] , we investigate communication of hidden messages by means of an additional channel. That is, a ciphertext always encodes a fake message and can additionally contain a real hidden message. Furthermore, the assumption that the sender and the receiver share a secret appears powerful, and we investigate what exactly can be achieved without this assumption.

The contribution of this chapter is the definition of the first *sender-and-receiver* deniable encryption scheme. The scheme is efficient, practical and is obtained from standard tools.

Part of this chapter is the result of a joint work with Prof. Giuseppe Ateniese and Prof. Marina Blanton.

3.1 Background and Definitions

The original definitions of deniable encryption given in [CDNO97] are stated in terms of computational indistinguishability of views associated with real and fake messages. In

particular, in ad-hoc encryption, the encryption algorithm will take a real message m_r and produce a ciphertext. The so-called *faking algorithm* allows the ciphertext to be opened to reveal a fake message m_f. Then the security requirements are such that the communication between the sender and receiver when transmitting a real message in place of a fake message is indistinguishable with the communication when the fake message is being sent. Additionally, the deniability property requires that the view of the adversary that includes communication of encryption of m_r and opening of the corresponding ciphertext to m_f is indistinguishable from communication of encryption of m_f and its opening. This view accounts for all random choices revealed by the faking algorithm. Sender-deniability means that the sender reveals its random choices/keys, and receiver deniability means that the receiver opens its random choices/keys. Plan-ahead deniable encryption is defined similarly with the exception that the fake message is determined at the time of encryption.

In our case, we define deniable encryption as the ability to hide the presence of the real message. That is, the view of the adversary when transmitting a fake message together with a hidden real message should be indistinguishable from the case when no hidden message is included.

Definition 3.1.1. *A (plan-ahead) deniable public-key encryption scheme consists of the following four efficient algorithms:*

- Setup : *a probabilistic algorithm that, on input a security parameter* 1^κ*, outputs* (pk, sk)*, where* pk *is the public key and* sk *is the private key.*
- Enc : *a probabilistic algorithm that, on input* pk *and messages* m_f*,* m_r*, outputs a ciphertext* $C(m_f; m_r)$*. We explicitly denote by* r *the random choices made by this algorithm.*
- Dec : *a deterministic algorithm that, on input* (pk, sk) *and ciphertext* C*, outputs messages* m_r *and* m_f *encoded in the ciphertext.*
- Open : *a deterministic algorithm that on input a ciphertext* C *and public key* pk*, outputs private information* PI *that opens the encryption to message* m_f*.*

In the above, what constitutes the private input in the opening algorithm depends on what party is coerced into opening the encryption. If such a party is the sender, PI can consist of random choices r used in ciphertext generation. If the receiver is opening the ciphertext, PI can consist of the private key sk. Finally, if both of them are to open the ciphertext, the information PI available to the adversary will consist of the union of the sender's and receiver's information. When the Open algorithm is invoked by the sender, we denote it by Open_S, and when it is invoked by the receiver, it is denoted by Open_R.

Let $\mathcal{D} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Open}_S, \mathsf{Open}_R)$ be a deniable public-key encryption scheme, where $(pk, sk) \xleftarrow{R} \mathsf{Setup}(1^\kappa)$. The security properties required by $\mathcal{D}$ are inspired by the

security definition in [CDNO97], adapted to our setting. However, instead of using definitions based on computational indistinguishability of the view of the adversary, we provide a rigorous definition of (public-key) deniability in terms of interactive experiments. An informal but intuitive description of the security properties is provided next, followed by the formal definition of deniability.

Let $C(m_f; m_r)$ denote the ciphertext associated with encrypting a fake and real messages, and $C(m_f; -)$ denote the ciphertext formed by embedding only one message m_f into the encryption. Then $\mathcal{D}$ must satisfy the following properties (informal):

Correctness: given a ciphertext, the decryption algorithm outputs the same messages as the message that were used to form the ciphertext using the encryption algorithm.

Security: for any messages m_f and m_r, random variables $C(m_f; m_r)$ and $C(m_f; -)$ are computationally indistinguishable.

Deniability: we state this property separately for different notions of deniability.

sender-deniability: the random variables $(C(m_f;\ m_r),\ \mathsf{Open}_S(C(m_f;\ m_r), pk),\ m_f)$ and $(C(m_f; -),\ PI,\ m_f)$, where PI corresponds to the random choices used in forming $C(m_f; -)$ that allow the ciphertext to be opened, are computationally indistinguishable.

receiver-deniability: the random variables $(C(m_f; m_r),\ \mathsf{Open}_R(C(m_f; m_r), pk),\ m_f)$ and $(C(m_f; -),\ PI,\ m_f)$, where PI allows to open the ciphertext (e.g., can be the private key sk), are computationally indistinguishable.

sender-and-receiver-deniability: the random variables $(C(m_f; m_r),\ \mathsf{Open}_S(C(m_f; m_r), pk),\ \mathsf{Open}_R(C(m_f; m_r), pk),\ m_f)$ and $(C(m_f; -),\ PI_1,\ PI_2,\ m_f)$ are computationally indistinguishable.

When discussing sender deniability, we assume that a coercer can reasonably well guess who the intended recipient of the transmission is (e.g., if the message is communicated via email, recipient identity can be deduced from the transmission). Thus, the sender has to use the true recipient's public key during the opening phase.

The deniability requirement of a deniable public-key encryption scheme is strictly stronger than the security requirement (i.e., any notion of deniability above implies security), and therefore we next formally formulate the deniability property and only prove our solution deniable. Unlike prior literature, we specify the deniability property using an interactive experiment to more precisely capture the power of the adversary.

We define the experiment for sender-deniability of deniable encryption scheme

$$\mathcal{D} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Open}_S)$$

as follows:

Experiment $\mathsf{SDen}_{\mathcal{A},\mathcal{D}}(\kappa)$

1. Set a system-wide table T initially empty.
2. Run $(pk, sk) \leftarrow \mathsf{Setup}(1^\kappa)$.
3. Adversary $\mathcal{A}$ is given pk and oracle access to $\mathsf{Open}_S(\cdot, pk)$ and to $\mathsf{Enc}(\cdot, \cdot, pk)$. Eventually $\mathcal{A}$ outputs a message pair (m_f, m_r), with $|m_f| = |m_r|$.
4. A random bit $b \stackrel{R}{\leftarrow} \{0, 1\}$ is drawn; if $b = 0$, $\mathcal{A}$ is given ciphertext $C(m_f; m_r)$, and if $b = 1$, $\mathcal{A}$ is given ciphertext $C(m_f; -)$.
5. Adversary $\mathcal{A}$ continues to have access to $\mathsf{Open}_S(\cdot, pk)$ and to $\mathsf{Enc}(\cdot, \cdot, pk)$.
6. $\mathcal{A}$ outputs bit b', and the experiment outputs 1 if and only if $b = b'$.

$\mathsf{Enc}(m_f, m_r, pk)$:

1. Compute $C(m_f; m_r)$.
2. Store the tuple $(m_f, m_r, C(m_f; m_r), r)$ in T, where r denotes the set of the random choices.
3. Output $C(m_f; m_r)$.

$\mathsf{Open}_S(C, pk)$:

1. If C is in T return r.
2. If C is the challenge ciphertext (either $C(m_f; m_r)$ or $C(m_f; \perp)$) return only the part in r used to encrypt m_f.
3. Otherwise return $\perp$.

Note that we allow the adversary to open any ciphertext of its choice including the challenge ciphertext (produced on a message of its choice).

A more detailed explanation of this definition is in order. The adversary $\mathcal{A}$ is given access to an opening oracle and an encryption oracle under the public key pk. Note that even though the definition is for public-key encryption schemes we still provide access to an encryption oracle to model the ability of the coercer to choose arbitrary messages (m_f, m_r), have them encrypted by the sender, and have the sender reveal the random choices used in the encryption.

Definition 3.1.2. *We say that a public-key encryption scheme $\mathcal{D}$ is* sender-deniable *if there exists a negligible function* negl *such that for any probabilistic polynomial time $\mathcal{A}$,* $\Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{D}}(\kappa) = 1] \leq 1/2 + \mathsf{negl}(\kappa)$.

The experiment for receiver-deniability of deniable encryption scheme

$$\mathcal{D} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Open}_R)$$

is define as follows:

Experiment $\mathsf{RDen}_{\mathcal{A},\mathcal{D}}(\kappa)$

1. Run $(pk, sk) \leftarrow \mathsf{Setup}(1^\kappa)$.
2. Adversary $\mathcal{A}$ is given pk and oracle access to $\mathsf{Open}_R(\cdot, pk)$ and eventually outputs a message pair (m_f, m_r), with $|m_f| = |m_r|$.
3. A random bit $b \xleftarrow{R} \{0, 1\}$ is drawn; if $b = 0$, $\mathcal{A}$ is given ciphertext $C(m_f; m_r)$, and if $b = 1$, $\mathcal{A}$ is given ciphertext $C(m_f; -)$.
4. Adversary $\mathcal{A}$ continues to have access to $\mathsf{Open}_R(\cdot, pk)$.
5. $\mathcal{A}$ outputs bit b', and the experiment outputs 1 if and only if $b = b'$.

Note that by allowing the adversary to have oracle access to Open_R, we make it very powerful. In particular, for existing receiver-deniable constructions, calling Open_R once will give the adversary access to the private decryption key sk. In practice, once a receiver is coerced into opening a ciphertext (and thus revealing the private key), the key pair should no longer be used and a new key will be generated. We, however, grant the adversary this level of power to show that, even after revealing the private key, the real message m_r remains hidden.

Definition 3.1.3. *We say that a public-key encryption scheme $\mathcal{D}$ is* receiver-deniable *if there exists a negligible function* negl *such that for any probabilistic polynomial time $\mathcal{A}$,* $\Pr[\mathsf{RDen}_{\mathcal{A},\mathcal{D}}(\kappa) = 1] \leq 1/2 + \mathsf{negl}(\kappa)$.

Additionally, we say that a public-key encryption scheme $\mathcal{D}$ is *sender-and-receiver* deniable if it satisfies the requirements for both sender deniability and receiver deniability. We emphasize that we are not considering adversaries with access to a decryption oracle.

3.2 Deniable Encryption from Standard Tools

3.2.1 False Start

Before presenting our solution, we briefly (and informally) describe one simple approach and show why it does not achieve the goal. Suppose receiver R has a public-private key pair (pk, sk), where pk is publicly known. Also suppose that the receiver sets up another key pair (pk', sk') without publicly advertising it. The receiver communicates pk' to sender S. When the sender wants to send a message m to R without the ability to deny it, it simply forms the ciphertext as $E_{pk}(m)$. When S wants to send a message without the ability of a coercer to have access to it, it sends $E_{pk}(E_{pk'}(m))$.

There are significant problems with this solution. First of all, if a coercer approaches the receiver and obtains sk, the decrypted message will appear as a random string rather than a meaningful plaintext, which would make it suspicious. Similarly, if the coercer approaches the sender and asks for the random choices used in generating the ciphertext, the fact that the message looks like a random string will be even more suspicious than in case of coercing the receiver. Furthermore, this approach requires both the sender and the receiver to share a (secret) key. And if the adversary is aware of the above way of forming transmissions (and we cannot assume that the approach will stay unknown), the coercer can request information about the second key, which makes the scheme completely non-deniable.

3.2.2 Preliminaries

Our general results utilize a secure encryption scheme satisfying certain standard security properties. Let $\mathcal{E} = (Gen, E, D)$ denote a public-key encryption scheme where Gen, E and D are polynomial-time algorithms defined in the standard way (in particular, Gen and E are probabilistic algorithms while D is deterministic). Given a public key pk generated by Gen, we will denote by $\mathcal{M}_{pk}$ and $\mathcal{C}_{pk}$ the message and ciphertext spaces defined by pk. In addition, for simplicity and without loss of generality, we assume that messages in $\mathcal{M}_{pk}$ and ciphertexts in $\mathcal{C}_{pk}$, respectively, have all the same size.

We next state the properties of an encryption scheme we rely on. Consider the standard IND-CPA game defined as follows:

Experiment $\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{E}}(\kappa)$

1. Run $(pk, sk) \leftarrow Gen(1^\kappa)$.
2. Adversary $\mathcal{A}$ is given pk and eventually outputs messages $m_0, m_1 \in \mathcal{M}_{pk}$ of its choice.

3. A random bit $b \xleftarrow{R} \{0,1\}$ is drawn and $E_{pk}(m_b)$ is returned to $\mathcal{A}$.

4. $\mathcal{A}$ outputs bit b', and the experiment outputs 1 if and only if $b = b'$.

Definition 3.2.1 (IND-CPA security)**.** *An encryption scheme $\mathcal{E} = (Gen, E, D)$ is said to have indistinguishable encryptions under chosen plaintext attack if there exists a negligible function* negl *such that for any probabilistic polynomial time $\mathcal{A}$,* $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{E}}(\kappa) = 1] \leq 1/2 + \mathsf{negl}(\kappa)$.

Definition 3.2.2 (Sampleable)**.** *We say that an encryption scheme $\mathcal{E} = (Gen, E, D)$ has sampleable ciphertext space if there exists a polynomial time (in κ) algorithm such that, on input pk (produced by $Gen(1^\kappa)$), can choose an element of $\mathcal{C}_{pk}$ with the same distribution the algorithm $E_{pk}(\cdot)$ produces on a fixed message m.*

We note that when the encryption scheme $\mathcal{E}$ is IND-CPA-secure, the above definition will imply that the distribution of ciphertexts $E_{pk}(m)$ is computationally indistinguishable from the distribution of ciphertexts $E_{pk}(m')$ for messages $m \neq m'$. Then we can sample the ciphertext space by creating $E_{pk}(m)$, and the view will be indistinguishable from the distribution of encryptions of other messages.

3.2.3 Our Solution

The main idea behind our solution is simple: a receiver's public key consists of two parts, where the first part is a public key and the second part could be another public key or a randomly chosen string. To achieve deniability, the second half of the key is claimed to be random. Let $\mathcal{E} = (Gen, E, D)$ denote a IND-CPA-secure encryption scheme. Let us also denote the domain of valid public-private key pairs output by $Gen(1^\kappa)$ as $\mathcal{K}(\kappa)$ and the set of all possible public keys output by $Gen(1^\kappa)$ as $\mathcal{PK}(\kappa)$.

We first present our construction and show it to be a sender-deniable encryption scheme, and then proceed with describing and proving receiver deniability (thus, it has sender-and-receiver deniability).

Setup : On input a security parameter 1^κ, a user generates a key pair $(pk_1, sk_1) \leftarrow Gen(1^\kappa)$ and also either generates a second key pair $(pk_2, sk_2) \leftarrow Gen(1^\kappa)$ or chooses a random value from $\mathcal{PK}(\kappa)$. The public key of the user is then $pk = (k_1, k_2)$, where $k_1 = pk_1$ and k_2 is either pk_2 or the random value. The user's corresponding private key is $sk = (s_1, s_2)$, where $s_1 = sk_1$ and s_2 is either sk_2 or the empty string (denoted by $\perp$).

The public key $pk = (k_1, k_2)$ is disseminated. If such keys get registered or certified, the certifying authority will request a proof of knowledge of the private key corresponding to k_1, but not to k_2. For simplicity, we will also employ a public redundancy function $\mathsf{red}(\cdot)$ that maps elements from a generic message space to a redundancy space which is a subset of $\mathcal{M}_{k_2}$.

Enc : Given a public key $pk = (k_1, k_2)$, the sender forms a ciphertext as follows, depending on whether deniability is used or not. To deniably transmit a message m along with an innocent fake message m_f, the sender forms the encryption as

$$\mathsf{Enc}(pk, m_f, m) = (c_1, c_2) = (E_{k_1}(m_f), E_{k_2}(\mathsf{red}(m))).$$

To transmit a message m without the ability to deny it, the sender forms the encryption as

$$\mathsf{Enc}(pk; m) = (c_1, c_2) = (E_{k_1}(m), R),$$

where R is chosen according to the distribution defined by $\mathsf{Enc}(pk; \cdot)$ on $\mathcal{C}_{k_2}$.

Dec : Given a ciphertext (c_1, c_2), the recipient with the public key $pk = (k_1, k_2)$ and private key $sk = (s_1, s_2)$ proceeds as follows:

1. Set $m_1 = D_{s_1}(c_1)$.
2. If $s_2 \neq \perp$, compute $m_2 = D_{s_2}(c_2)$.
3. If m_2 is in the redundancy space (the codomain of the function $\mathsf{red}(\cdot)$) then return (m_1, m_2), otherwise return $(m_1, \perp)$.

Open_S : Suppose a coercer obtains a ciphertext (c_1, c_2) generated by the sender and requests its opening. Given public key $pk = (k_1, k_2)$, the sender opens the random choices made in generating c_1. The sender also claims that c_2 was chosen from $\mathcal{C}_{k_2}$ according to the distribution defined by $\mathsf{Enc}(pk; \cdot)$. Thus, the actual message the sender claims to have sent is $(m_1, \perp)$.

Theorem 3.2.3. *Assuming $\mathcal{E} = (Gen, E, D)$ is a* IND-CPA*-secure encryption scheme with security parameter κ, the scheme above is sender-deniable.*

Proof of Theorem 3.2.3. Given an IND-CPA secure scheme $\mathcal{E} = (Gen, E, D)$, we show that if adversary $\mathcal{A}$ wins in the *sender deniability* experiment with non-negligible probability then we can build an efficient algorithm $\mathcal{B}$ that wins in the IND-CPA experiment with non-negligible probability. Let $\Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{E}}(\kappa) = 1] - 1/2 = \delta$.

$\mathcal{B}$ engages in the IND-CPA game with a challenger, obtains the public key $\overline{pk}$ and uses it to setup the *sender deniability* experiment for $\mathcal{A}$ by constructing $pk = (k_1, k_2)$ where k_1 is generated using $Gen(1^\kappa)$ and $k_2 = \overline{pk}$ (i.e., $\mathcal{B}$ sends also k_2 to $\mathcal{A}$ without knowing the

corresponding private key). $\mathcal{B}$ responds to the $\mathsf{Enc}(m_i, m_j, pk)$ queries with $C(m_i\,;\ m_j)$. Let (r_i, r_j) be the randomness used by $\mathcal{B}$ to respond to the Enc query. $\mathcal{A}$ eventually outputs its choice for the pair (m_f, m_r). $\mathcal{B}$ outputs m_0, m_1 to the challenger where $m_0 = m_r$, and $m_1 = R$ is a random message from the message space such that $|R| = |m_r|$. The challenger picks a random bit b and returns $\bar{c} = E_{\overline{pk}}(m_b)$. $\mathcal{B}$ builds the challenge ciphertext for $\mathcal{A}$ as $\overline{C} = (c_1\,;\ \bar{c})$, where $c_1 = E_{k_2}(m_f)$. Let r be the randomness used by $\mathcal{B}$ to encrypt c_1. $\mathcal{B}$ responds to $\mathsf{Open}_S(C, pk)$ queries with r if $C = \overline{C}$, with (r_i, r_j) if $C = C(m_i, m_j)$, and with $\perp$ otherwise. Eventually $\mathcal{A}$ outputs b', and $\mathcal{B}$ outputs the same guess to the challenger.

It is not hard to see that $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{B},\mathcal{E}}(\kappa) = 1] = \Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{E}}(\kappa) = 1]$. That is, if $b = 0$, $\mathcal{B}$ receives the encryption of m_f and $\mathcal{A}$ receives the encryption of (m_f, m_r), in which case $\mathcal{B}$ answers the challenge correctly if and only if $\mathcal{A}$ answers its challenge correctly. If $b = 1$, $\mathcal{B}$ receives the encryption of R and $\mathcal{A}$ receives the encryption of (m_f, R), which is interpreted as $(m_f, -)$ since R is a random message. Therefore $\mathcal{B}$ guesses correctly if and only if $\mathcal{A}$ guesses correctly. Since $\mathcal{B}$ cannot guess correctly non-negligibly more than half of the times, $\delta \leq \mathsf{negl}(\kappa)$ for some negligible function negl. □

We next show how the above scheme achieves receiver deniability. The algorithms Setup, Enc, and Dec remain unchanged and we only provide a description of the Open algorithm. The main distinction between sender and receiver deniability in this scheme is that receiver deniability requires concealing the second private key s_2 (if present) and not revealing it to the coercer. In other words, s_2 must stay secret even upon coercion.

Open_R : Suppose a coercer obtains a ciphertext (c_1, c_2) that the recipient received. Given the recipient's public key $pk = (k_1, k_2)$, the recipient opens its private key as $sk = (s_1, \perp)$ such that s_1 matches k_1. In other words, the recipient claims that the second part of the public key was chosen uniformly at random from $\mathcal{PK}(\kappa)$ (here we assume w.l.o.g. that the public keys generated by $Gen(\cdot)$ follow the uniform distribution).

Theorem 3.2.4. *Assuming $\mathcal{E} = (Gen, E, D)$ is a* $\mathsf{IND\text{-}CPA}$*-secure encryption scheme with security parameter κ, the scheme above is receiver-deniable.*

Proof of Theorem 3.2.4. Given an $\mathsf{IND\text{-}CPA}$ secure scheme $\mathcal{E} = (Gen, E, D)$, we show that if adversary $\mathcal{A}$ wins in the *receiver deniability* experiment with non-negligible probability then we can build an efficient algorithm $\mathcal{B}$ that wins in the $\mathsf{IND\text{-}CPA}$ experiment with non-negligible probability. Let $\Pr[\mathsf{RDen}_{\mathcal{A},\mathcal{E}}(\kappa) = 1] - 1/2 = \delta$.

$\mathcal{B}$ engages in the $\mathsf{IND\text{-}CPA}$ game with a challenger, obtains the public key $\overline{pk}$ and uses it to setup the *receiver deniability* experiment for $\mathcal{A}$ by constructing $pk = (k_1, k_2)$ where k_1 is generated using $Gen(1^\kappa)$ and $k_2 = \overline{pk}$ (i.e., $\mathcal{B}$ sends also k_2 to $\mathcal{A}$ without knowing the corresponding private key). Let s_1 be the private key associated with k_1. $\mathcal{B}$ responds to

any $\mathsf{Open}_R(C, pk)$ query with $sk = (s_1, \perp)$, regardless of whether C encrypts one messages or two. $\mathcal{A}$ eventually outputs its choice for the pair (m_f, m_r). $\mathcal{B}$ outputs m_0, m_1 to the challenger where $m_0 = m_r$, and $m_1 = R$ is a random message from the message space such that $|R| = |m_r|$. The challenger picks a random bit b and returns $\bar{c} = E_{\overline{pk}}(m_b)$. $\mathcal{B}$ constructs the challenge ciphertext for $\mathcal{A}$ as $C = (c_1\,;\,\bar{c})$, where $c_1 = E_{k_1}(m_f)$.

Eventually $\mathcal{A}$ outputs b', and $\mathcal{B}$ outputs the same guess to the challenger.

Clearly $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{B},\mathcal{E}}(\kappa) = 1] = \Pr[\mathsf{RDen}_{\mathcal{A},\mathcal{E}}(\kappa) = 1]$. Indeed, if $b = 1$, $\mathcal{B}$ receives the encryption of R and $\mathcal{A}$ receives the encryption $C(m_f;\, R)$, which is interpreted as $C(m_f;\, -)$ since R is a random message. Thus $\mathcal{B}$ answers the challenge correctly if and only if $\mathcal{A}$ answers its challenge correctly.

If $b = 0$, $\mathcal{B}$ receives the encryption of m_f and $\mathcal{A}$ receives the encryption of (m_f, m_r). $\mathsf{Open}_R(C, pk)$ returns $sk = (s_1, \perp)$ which exposes m_f. Thus even in this case $\mathcal{B}$ guesses correctly if and only if $\mathcal{A}$ guesses correctly. Since $\mathcal{B}$ cannot guess correctly non-negligibly more than half of the times, $\delta \leq \mathsf{negl}(\kappa)$ for some negligible function negl. □

We next show specific instantiations of our construction, which will also be useful in demonstrating extensions to our basic solution.

3.2.4 Instantiation based on ElGamal Encryption

Let G be a multiplicative group of order a prime q such that the discrete logarithm problem is hard in G. Let g be a generator of G. The redundancy function $\mathsf{red}(\cdot)$ now maps elements directly into a subset of G.

Setup: Choose $x_1 \xleftarrow{R} \mathbb{Z}_q^*$ and set $s_1 = x_1$, $k_1 = (G, q, g, g_1^x)$. The second half of the key is set to either $s_2 = \perp$, $k_2 = (G, q, g, R)$, where $R \xleftarrow{R} G$ or $s_2 = x_2$, $k_2 = (G, q, g, g^{x_2})$, where $x_2 \xleftarrow{R} \mathbb{Z}_q^*$. We abbreviate the public key as $pk = (G, q, g, h_1, h_2)$.

Enc: Given a key $pk = (G, q, g, h_1, h_2)$, encrypt m with deniability as $\mathsf{Enc}(pk; m_f, m) = (g^{r_1}, h_1^{r_1} m_f, g^{r_2}, h_2^{r_2}\mathsf{red}(m))$, where $r_1, r_2 \xleftarrow{R} \mathbb{Z}_q^*$ and $m_f \in G$; otherwise, encrypt as $\mathsf{Enc}(pk; m) = (g^{r_1}, h_1^{r_1} m_f,\, R_1, R_2)$, where $r_1 \xleftarrow{R} \mathbb{Z}_q^*$ and $R_1, R_2 \xleftarrow{R} G$.

Dec: Given ciphertext $(c_{11}, c_{12}, c_{21}, c_{22})$ and private key $sk = (s_1, s_2)$, set $m_1 = c_{12} c_{11}^{-s_1}$. If $s_2 \neq \perp$, set $m_2 = c_{22} c_{21}^{-s_2}$. If m_2 is not in the redundancy space, discard it.

Open_S: Given ciphertext $(c_{11}, c_{12}, c_{21}, c_{22})$ and public key $pk = (k_1, k_2)$, the sender opens the ciphertext by revealing r_1, such that $c_{11} = g^{r_1}$. This exposes m_1, and the sender claims

that values c_{21} and c_{22} were chosen uniformly at random from the group G.

Open_R: Given ciphertext $(c_{11}, c_{12}, c_{21}, c_{22})$ and public key $pk = (k_1, k_2)$, the recipient opens the private key $sk = (x_1, \perp)$, such that $g^{x_1} = h_1$. This exposes m_1.

Each ciphertext consists of four elements of G and cannot be shortened in a straightforward way while maintaining deniability. Note that, in the case of ElGamal in prime-order groups, the distribution induced by the encryption algorithm under a fixed public-key and a fixed message is the uniform one. Thus, sampling the ciphertext space is very efficient since we can just pick elements uniformly at random from the prime-order group without resorting to the encryption algorithm.

3.2.5 Instantiation based on IBE

We illustrate the case of Identity-Based Encryption (IBE) [Sha84] using Boneh-Franklin IBE scheme [BF03a]. The crucial difference between identity-based and other encryption scheme is that, since an identity is used as a public-key, a central authority with the master key must be involved in the generation of the corresponding private key. Each user is assigned two different identity strings. These strings can be derived from the user ID (such as the email address) for example as $pk = (k_1, k_2) = (ID||\text{“1”}, ID||\text{“2”})$, where $||$ denotes string concatenation. To ensure that the second key be optional, we use a standard oblivious transfer protocol during key retrieval, so that the central authority does not learn whether the user obtained the corresponding private key or not. OT protocols are well-studied and can be realized under standard assumptions (see for example [Rab81], [GIKM98]).

The IBE scheme utilizes bilinear maps which we define next. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of large prime order q. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ must satisfy the following properties:

1. *bilinear*: $e(g^a, h^b) = e(g, h)^{ab}$ for any $a, b \in \mathbb{Z}_q$ and $g, h \in \mathbb{G}_1$.
2. *non-degenerate*: if g generates $\mathbb{G}_1$, $e(g, g)$ generates $\mathbb{G}_2$.
3. *efficient*: there is an efficient algorithm to compute $e(g, h)$ for any $g, h \in \mathbb{G}_1$.

The original Boneh-Franklin IBE scheme is secure under the bilinear Diffie-Hellman (BDH) assumption in the random oracle model. Our deniable variant of their scheme is defined by the following algorithms:

SSetup: Setup a bilinear map $e, \mathbb{G}_1, \mathbb{G}_2, q$ and generator g of $\mathbb{G}_1$. Choose two cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some n. Choose a random master key $s \in Z_q$, set $h = g^s$, and publish public parameters $(q, \mathbb{G}_1, \mathbb{G}_2, e, g, h, n, H_1, H_2)$.

The message space is $\{0,1\}^n$, and the ciphertext space is $\mathbb{G}_1 \times \{0,1\}^n$. A system-wide redundancy function $\mathsf{red}(\cdot)$ is used that maps binary strings into a subset of $\{0,1\}^n$.

Setup: Given (k_1, k_2) and public parameters, the user computes $t_1 = H_1(k_1)$ and $t_2 = H_1(k_2)$. User obtains the first part of the secret key $s_1 = t_1^s$ from the central authority. To setup the second part, the user and the central authority engage in 1-out-of-2 OT protocol, at the end of which the user either learns the second private key $s_2 = t_2^s$ or a dummy (or empty) value in which case the user sets $s_2 = \perp$. The central authority learns nothing.

Enc: Given identities $pk = (k_1, k_2)$ and system public parameters, first compute $t_1 = H_1(k_1)$ and $t_2 = H_1(k_2)$. To encrypt with deniability, compute the ciphertext as $\mathsf{Enc}(pk;\, m_f, m) = (c_{11}, c_{12}, c_{21}, c_{22}) = (g^{r_1}, m_f \oplus H_2(e(t_1, h)^{r_1}), g^{r_2}, \mathsf{red}(m) \oplus H_2(e(t_2, h)^{r_2}))$, where $r_1, r_2 \xleftarrow{R} \mathbb{Z}_q$. Otherwise, encrypt as $\mathsf{Enc}(pk; m) = (c_{11}, c_{12}, c_{21}, c_{22}) = (g^{r_1}, m_f \oplus H_2(e(t_1, h)^{r_1}),\ R_1, R_2)$, where $r_1 \xleftarrow{R} \mathbb{Z}_q$, $R_1 \xleftarrow{R} \mathbb{G}_1$, and $R_2 \xleftarrow{R} \{0,1\}^n$.

Dec: Given ciphertext $(c_{11}, c_{12}, c_{21}, c_{22})$, public parameters, and private key $sk = (s_1, s_2)$, first compute $m_1 = c_{12} \oplus H_2(e(s_1, c_{11}))$. If $s_2 \neq \perp$, compute $m_2 = c_{22} \oplus H_2(e(s_2, c_{21})$. If m_2 is not in the redundancy space, discard it.

Open_S: Given ciphertext $(c_{11}, c_{12}, c_{21}, c_{22})$, public key $pk = (k_1, k_2)$, and public parameters, the sender reveals r_1 such that $c_{11} = g^{r_1}$. This allows the coercer to recover message m_1 using c_{21}, k_1, and public parameters. The sender claims that $(c_{21},\ c_{22})$ were chosen uniformly at random from $\mathbb{G}_1 \times \{0,1\}^n$.

Open_R: Given the ciphertext $(c_{11}, c_{12}, c_{21}, c_{22})$ and public-key $pk = (k_1, k_2)$, the recipient opens the private key $sk = (t_1^s, \perp)$, such that $t_1 = H_1(k_1)$ and $e(sk, g) = e(t_1, h)$. This exposes the message m_1.

Even in the case of our deniable variant of the Boneh-Franklin IBE scheme, sampling from the ciphertext space is straightforward and it is enough to choose ciphertexts uniformly at random from $\mathbb{G}_1 \times \{0,1\}^n$. This is because the first component of the ciphertext is uniformly distributed in $\mathbb{G}_1$, since all elements in $\mathbb{G}_1$ (except 1) have all the same order which is prime, and the second component of the ciphertext is uniformly distributed in $\{0,1\}^n$ since it is the result of the XOR between the message and the output of a random oracle (H_2) which is random and independent of the message.

3.3 Extensions

The scheme given in the previous section doubles the size of the ciphertext as compared to the underlying encryption scheme. In this section, we extend the base scheme to support

communication of several deniable messages in a single ciphertext and show how specific instantiations of our general solution can result in reducing the overhead associated with the general deniable encryption solution. In addition, we show a simple mechanism to reduce the space required by the sender to store the random choices associated with each encryption. Finally, we introduce *phishing* attacks against non-interactive deniable schemes and provide simple (albeit inefficient) solutions to mitigate them.

In this section we provide only high-level and informal descriptions. They are meant to provide an intuition and generic insights and are not rigorous. This section can be skipped with no notable inconvenience.

3.3.1 Increasing Message Space of Deniable Communication

The basic idea behind this extension is to enlarge the ciphertext to permit communication of several deniable messages. To do so, a user now registers more than two public keys, where the user knows the private key corresponding to the first public key only, and private keys associated with all other public keys may be unknown to the user (if they are known, they can be used for deniable communication). We use $pk = (k_0, k_1, \ldots, k_n)$ to denote such public keys. The private key is then of the form $sk = (s_0, s_1, \ldots, s_n)$, where sk_i can be $\perp$ for $i = 1, \ldots, n$.

It is not difficult to see that a ciphertext can now be formed as anything between $E(pk; m_f, m_1, \ldots, m_n) = (E_{k_0}(m_f), E_{k_1}(m_1), \ldots, E_{k_n}(m_n))$ and $E(pk; m) = (E_{k_0}(m), R_1, \ldots, R_n)$ and allows to deniably communicate up to k times as much information. Upon coercion, obviously, only the randomness corresponding to encryption with k_0 or private key s_0 are going to be opened.

This construction shortens the size of the ciphertext when properly instantiated. We provide an example next. Note that for clarity's sake we do not specify any redundancy function.

3.3.2 Shorter Ciphertext Size with ElGamal Encryption

Using ElGamal encryption to implement the above construction leads to 2 group elements per message. We, however, can reuse the first group element in encryptions of messages m_1 through m_n, as described by Bellare et al. in [BBS03]. That is, given public key $pk = (G, q, g, h_0, h_1, \ldots, h_n)$, the ciphertext is formed as $E(pk; m_f, m_1, \ldots, m_n) = (g^{r_0}, h_0^{r_0} m_f, g^{r_1}, h_1^{r_1} m_1, g^{r_1}, h_2^{r_1} m_2, \ldots, g^{r_1}, h_n^{r_1} m_n)$, such that g^{r_1} can be communicated only once. This results in $n + 2$ group elements to transmit $n + 1$ messages (n of which are deniable).

3.3.3 Space Savings with IBE

The above technique for shortening ElGamal ciphertext size also applies to the Boneh-Franklin IBE scheme. Additionally, the size of the public key can be reduced if all public keys (i.e., identities) of a user can be derived from a single identity string. For instance, as mentioned above, the public key can consists of a single id and n; the id is then used as k_0. Public keys k_i's for $i = 1, \ldots, n$ are set to $id||$"i". This results in no overhead in public parameters caused by the scheme.

3.3.4 Quasi-Stateless Sender

In deniable encryption, the sender is forced to keep state information regarding every message that is transmitted. This state information grows linearly in the number of messages and must be stored securely and reliably (it must be made available to the coercer upon request to avoid legal or unpleasant consequences).

Unfortunately, keeping state is a notoriously difficult problem in systems research. In addition, the information stored by the sender can be exposed and abused if the sender is ever compromised.

The coercer can ask the sender to open any message it sent at any moment. If the sender encrypts a message and then erases all the information used to encrypt it, such as the randomness or the associated plaintext, it is impossible to open it later upon request. Therefore some state information must be saved for every message sent. This state information must suffice to both open the encryption and to prove that it is a correct opening, i.e., just saving the plaintext associated with the ciphertext is not enough because it does not suffice to prove that it is the real plaintext when using a CPA-secure encryption scheme.

Providing message confidentiality with this added requirement is not trivial. Any adversary who can access the state information, can also obtain the plaintext of every message with no additional effort. This information grows linearly with the number of the sent messages.

A simple mechanism that is able to thwart all these potential problems is to embed the state information, encrypted with authenticated symmetric algorithm, into each ciphertext transmitted by the sender. In practice, the sender stores any relevant state information within every ciphertext and releases it whenever is questioned by the coercer on a specific message. The sender is the only party who has knowledge of the secret keys used to encrypt and authenticate the state information. These keys can be provided to the coercer upon request so that the sender does not have to be involved or interact with the coercer during the opening phase.

In this way, the secret key is the only information that the sender needs in order to open a

message and prove that the opening is correct. However, if the same key is used to encrypt the state information of all messages, once the coercer obtains it, it can open all of them without any further help of the sender; the sender cannot know which messages have been opened since no interaction between them is requested anymore. To avoid this, the sender may be tempted to choose a different secret key for each message. In this way, however, the number of the keys to store is equal to the number of sent messages and therefore not constant.

In the random oracle model it is trivial to generate different keys for each message using a random oracle and a message identifier. The encryption key used to protect the state information for a specific message is obtained from the output of the random oracle, given the message identifier as input.

3.3.5 Resilience against Phishing Attacks

In order to obtain receiver-deniability, the receiver must conceal the existence of the second private key s_2 (if present), so that it does not have to reveal it to the coercer. If the coercer determines the existence of s_2, the deniability of all messages encrypted under s_2 is compromised.

In practice, it may be possible for an adversary to obtain information about the existence of s_2. Since our scheme is a public-key cryptosystem, anybody can send encrypted messages to the receiver. A coercer can send a deniable message to the receiver asking it to perform a specific action that the coercer will be able to notice. In order to be successful, the coercer should request the receiver to perform an innocent looking task, such as opening a web page. The coercer will set-up an ad-hoc page so that it can be notified if the receiver opened it. By doing it, the receiver would prove to the coercer the existence of s_2. Even if we assume that receiver is smart enough to avoid performing any action the adversary might notice, the coercer can still try to find out the existence of s_2 by exploiting a specific vulnerability of the software used by the receiver to read the message. As an example, the email clients Outlook and Outlook Express were shown to have several vulnerabilities which allowed an adversary to execute arbitrary code on the user's machine (see, for example, [NISf] [NISe] [NISc] [NISa] [NISd]). Therefore we should assume that when our scheme is used in practice the adversary will be able to determine the existence of the second private key for some messages.

As far as we can ascertain, this type of phishing attack has not been considered before in the deniability literature and seemingly any existing plan-ahead public-key scheme is vulnerable to it.

As a trivial solution to this problem, instead of using the same keypair $pk = (k_1, k_2)$ and $sk = (s_1, s_2)$ for all encryptions, the receiver generates several independent key pairs

$pk_i = (k_1, k_{2,i})$ and $sk_i = (s_1, s_{2,i})$. Each message is encrypted using a different and independent public key. Even if the coercer establishes the existence of a specific private key $s_{2,i}$, it does not learn any information about any other $s_{2,j}$. Unfortunately, this trivial solution has a very high cost in terms of key distribution and imposes a limit on the number of messages which can be sent to a receiver.

As a first, incorrect attempt to solve this problem more efficiently, the receiver may try to use a forward-secure public key encryption scheme to obtain receiver deniability. A forward-secure public key encryption scheme [CHK07] protects secret keys from exposure by evolving the keys at regular periods of time. When a key evolves into a new key, the old secret key is deleted and cannot be reconstructed from the new secret key and therefore cannot be provided to the coercer. Unfortunately this solution does not suffice to provide receiver deniability since the existence of a private key at a specific time interval implies the existence of private keys corresponding to all previous time intervals.

A better solution is to use an identity-based encryption scheme (IBE) to "compress" the public keys into a single value in order to solve the key distribution issue. Roughly, the receiver acts as the IBE trusted authority (PKG) itself, divides the time in intervals, and generates an IBE key under its identity for each time interval. After this process, the recipient is supposed to delete any master secrets of the PKG and keep only the IBE keys corresponding to a randomly selected *subset* of all time intervals. The sender encrypts messages for the receiver with a key valid at a specific time, e.g., "*Username* || *Year* || *Month* || *Week*". If the coercer establishes the existence of the private key for a time interval, only the messages encrypted in that interval are compromised. The problem with this approach is that the secret key consists of several values, the number of which depends on the granularity desired. To avoid this, the receiver can claim to delete the master secret keys of the IBE trusted authority but keep a copy instead deniably encrypted locally, so that new IBE keys can be generated when needed.

A more elegant solution can be built using a forward secure hierarchical identity-based encryption scheme (FS-HIBE) [YFDL04]. A hierarchical IBE (HIBE) [CS02] [HL02] allows a PKG to delegate the generation of private keys to lower-level PKGs. A FS-HIBE scheme allows each user in the hierarchy to refresh its private key periodically while keeping the public key the same. Using a FS-HIBE scheme, each user constructs a single path *Username* $\rightarrow$ *Year* $\rightarrow$ *Month* $\rightarrow$ *Week* and have some secret key to evolve while others get deleted. In this way, similarly to the previous solution, the exposure of one secret key compromises only the messages encrypted in one time interval. The receiver can safely claim that the it does not own the secret keys corresponding to different time periods.

The existence of more efficient solutions is still an open problem. We do not investigate this further at this time. We just pointed out that phishing attacks are relevant in this context since they seemingly affect any plan-ahead public-key (i.e., *non-interactive*) deniable encryption scheme.

3.4 A Toy Example: DenGP

In order to explore the practicality of our solution, we designed and implemented a prototype application, DenGP. The purpose of this application is to encrypt deniable messages with the primary use of sending them by email. Therefore it is designed to take advantage of existing public key infrastructures such as PGP's web of trust [AR97]. However its implementation does not prevent it from being used for any other purpose where deniable encryption is required. In chapter 6 we illustrate a large application based on the deniable encryption schemes presented in this chapter.

3.4.1 High-level Description

DenGP is a command line tool designed to be accessible from both a text-based console and graphical applications. It basically generates a deniable message $C(m_f; m_r)$ and then encrypts and signs it producing an OpenPGP[CDF+07] compliant ciphertext. By encrypting and signing the messages, we achieve two additional goals: i) it is impossible for an adversary, by simply observing the ciphertext, to recognize if it correspond to a deniable message. Therefore, if the coercer didn't request an opening of a specific message, both sender and deniable can be assured that it doesn't even know if it is a deniable message. ii) It is impossible for an active adversary to tamper with the message, such as substituting the non-deniable and/or the deniable part, unless it requests the sender to surrend the private key used to sign the message.

In order to encode the ciphertext in OpenPGP format, we based DenGP on Gnu Privacy Guard [GPG] (also known as GPG) and the libgcrypt library [GPG]. GPG is a well known open source implementation of the OpenPGP standard defined by RFC4880 [CDF+07], and therefore is compatible with the proprietary encryption software PGP. It can be used to decrypt any PGP-encrypted or verify any PGP-signed messages and works on most Unix implementations and on Windows. Libgcrypt is the library used as foundation for the new GPG 2.0, and implements all the cryptographic routines and the multi-precision integer arithmetic used by GPG but not the whole OpenPGP standard.

We chose the OpenPGP standard because it is a well established, well known and widely used in practice. Its native support for encoding the ciphertext in Radix-64[CDF+07], a base64 encoding variant, makes it very useful for encrypting any type of content which can then be easily sent as the content of an email message. Our application uses the ElGamal encryption algorithm to construct a "deniable payload" described in this paper, that is then encrypted using GPG.

3.4.2 Comparison with PGP/GPG

PGP and GPG do not implement any form of deniable encryption. Neither of them would also work in a scenario where the sender may be asked to open an encrypted message: since no random choices are saved for security reasons, the sender will not be able to open any message after encryption.

Clearly the receiver can let the coercer open any message by giving up its private key. However GPG implements two options, `--show-session-key` and `--override-session-key`, that were designed for making key escrow unnecessary: the first option allows the receiver to extract the session key from the ciphertext, and therefore let the coercer open the symmetric encrypted part of the ciphertext. The coercer can use the second option to specify the session key. These options, however, may not satisfy the coercer, who may suspect that there is a hidden message in the asymmetric encrypted part of the ciphertext. Therefore the coercer may still request the receiver to give up its secret key rendering the two options useless.

3.4.3 Implementation Details

Given a public key $pk = (k_1, k_2, k_3)$ and a pair (m_f, m_r) the application generates the ciphertext c using the following algorithm:

1. $\mathsf{Enc}(pk, m_f, m_r) = (c_1, c_2) = (E_{k_1}(m_f), E_{k_2}(m_r))$. Let r_1 be the random choices made encrypting m_f
2. Generate c_3 as an OpenPGP-encoded encryption under k_3 with optional signature of (c_1, c_2) using GPG and save the random choices into r_2
3. Generate c_4 as an OpenPGP-encoded encryption of (r_1, r_2) using a symmetric algorithm and a one-time symmetric key k
4. Output $c = (c_3, c_4)$

To transmit a message m without the ability to deny it, the application set $(c_1, c_2) = (E_{k_1}(m), R \in \mathcal{C}_{k_2})$ where $\mathcal{C}_{k_2}$ is the ciphertext space for k_2.

The use of hybrid encryption would clearly improve the performance of our toy example, both from a ciphertext expansion and encryption time point of view for long messages. However, for the scope of our analysis, this optimization is not relevant. Details about deniable hybrid encryption are provided in chapter 6.

Quasi-stateless Sender

The public key encryption algorithm used by the application is ElGamal. We can therefore limit the amount of information needed to open a specific ciphertext to the random choices made during encryption. The ElGamal instantiation of our deniable scheme is described in Section 3.2.4. In particular, given $pk = (G, q, g, h_1, h_2)$, the encryption of m_f and m is

$$Enc(pk; m_f, m) = (g^{r_1}, h_1^{r_1} m_f, g^{r_2}, h_2^{r_2} m)$$

where $r_1, r_2 \xleftarrow{R} \mathbb{Z}_q^*$. Note that the sender can open a message for the coercer by just revealing r_1: the coercer can calculate g^{r_1} and compare it to the ciphertext to confirm that r_1 is the correct value, and can then calculate $h_1^{r_1}$ to extract the message m_f. The coercer can then verify that m_f is in the expected message space as further confirmation. By using ElGamal also for the GPG encryption, the GPG-encrypted message can be opened by revealing its random choices. Let r denote such random choices.

Thanks to this property, our application encrypts the values r and r_1 using a symmetric cipher and attaches them at the end of the deniable ciphertext. The application requires either a *password*. which is used to derive a one time key for the current message, or a one time key k. If a password is provided, the application calculates $k = H(c_3 \mid\mid password)$ where H is the SHA-256 hash function. The symmetric algorithm used to encrypt (r, r_1) is AES_{256} employed in a variant of CFB mode.

It is clear that the ciphertext c_4, which hides the values r and r_1, introduces a new single point of failure for the confidentiality of the message when it is sent as non-deniable: an adversary who can decrypt c_4 automatically obtains m_f. For this reason, our implementation uses AES with the longest supported key. The overhead introduced by a longer key is negligible [DR99] in terms of time compared to the public-key encryptions performed by the application, and there is no added space requirement for the ciphertext since the block size for AES is independent from the key size. As usual, great care must be taken in choosing a hard to guess password [NISb], which in practical applications is the real single point of failure.

Ciphertext Expansion and Performances Evaluation

Using the default key length, i.e. 2048 bit, each encryption can carry up to 255 bytes of non-deniable (fake) message and up to 255 bytes of deniable (real) message. The content of GPG's payload (the deniable encryption) is four circa 2048-bit integers and some metadata such as field length and field types, for a total of 1130 bytes. The size of the GPG encryption without signature for this deniable payloads is, on average, 1734 bytes. This includes an ElGamal encryption of a session key, and therefore two 2048-bit integers g^r and $m \cdot y^r$, a symmetric encryption of the deniable payload under the session key, a message

authentication code and some metadata such as the algorithm identifiers and the recipient's public key identifier. Adding a 2048-bit RSA signature with the default key size brings the total size to 2037 bytes.

Thanks to the optimization described in Section 3.3.2 and to a different encoding of the value $m_r \cdot y^{r_1}$, we can reduce the size of the ciphertext by a non-negligible amount. More specifically, by reusing the value g^{r_1} for the encryption of both m_f and m_r, the ciphertext size is reduced to 1437 bytes for encryption only and 1738 bytes for encryption and RSA signature. The overhead introduced by including the encryption of r_1 and r_2 along with c_3 is minimal: the implementation of ElGamal in libgcrypt allows the use of a small exponent r for ElGamal, i.e. 337-bits long using a 2048-bits key. This length is obtained by multiplying the appropriate entry from Wiener's table [vOW96] on subgroup sizes matching with a specific field size by 3/2. Therefore just $2 \cdot 337$ bits need to be encrypted. The encryption mode defined in the OpenPGP specifications [CDF+07] is a variant of the CFB mode. OpenPGP CFB mode uses an initialization vector (IV) of all zeros, and prefixes the plaintext with a full block of the symmetric encryption algorithm plus two bytes of random data, such that the last two bytes of the prefix match the first two bytes. It does a CFB resynchronization after encrypting blocksize-plus-two bytes. Thus, for an algorithm with a block size of 16 bytes (128 bits), the IV is 18 bytes long, and octets seventeen and eighteen replicate octets fifteen and sixteen. Those extra two octets are an easy check for a correct key.

Our implementation uses the AES symmetric algorithm, which has a block size of 16 bytes (128 bits). The message to be encrypted has a size of 104 bytes. A complete OpenPGP symmetric-encrypted message also contains metadata, such as the string to key specifier (used to convert passphrase strings into symmetric-key encryption/decryption keys), the ID for the encryption algorithm, length headers etc. The total length of the encrypted values (r_1, r_2) is then 133 bytes. The default output format of our application is, as for GPG, a binary file. By specifying the `--armor` option the output is converted to an OpenPGP armored file, which is a Radix-64 encoded file with PGP headers and a checksum. In this way the content of the encrypted file can be simply pasted into an email and sent safely. The drawback of this solution, of course, is an increase in the message size.

The table below compares a standard OpenPGP encryption with and without signature for both our application and GPG. The plaintext is a 255-byte random string and all the keys used are 2048-bit long. The file encoding is OpenPGP binary.

Non deniable encryption	858 bytes
Non deniable encryption + sign (RSA)	1160 bytes
Deniable encryption	1570 bytes
Deniable encryption + sign (RSA)	1871 bytes

Clearly the size of the deniable encrypted ciphertext is greater than the size of the non-deniable encrypted ciphertext, since more information is contained in the ciphertext.

We also measured the time needed to encrypt and decrypt a message using our application and we compared it with GPG. All test were performed on a 2.53GHz Intel Core 2 Duo with 4GB of RAM under Mac OS X 10.5.6. The version of GPG used is 2.0.11. The compiler is gcc 4.0.1.

The table below summarizes our readings for encryption:

Non deniable encryption	110 ms
Non deniable encryption + sign (RSA)	221 ms
Deniable encryption	178 ms
Deniable encryption + sign (RSA)	290 ms

The table below reports the time needed to decrypt a message:

Non deniable decryption	121 ms
Non deniable decryption + verify (RSA)	124 ms
Deniable decryption	174 ms
Deniable decryption + verify (RSA)	178 ms

For the deniable application, by decryption we mean the decryption of both the fake and real message. The time required by our application to encrypt or decrypt a message is comparable to the time needed by GPG to perform the same action. We think that in the described scenario the added time would not be even noticeable by the user.

Chapter 4

Improving the Privacy and Flexibility of PKIs: StemCerts

Despite the high degree of security that can be offered and the wide availability of high quality open source software implementations for the creation, the management, and the use of digital certificates, Public Key Infrastructures (PKIs) have not enjoyed the widespread diffusion that security experts had envisioned after the introduction of public key cryptography. We believe that there are many reasons for this lack of practical success, including a relatively high cost, in terms of time and money, for obtaining high security grade certificates from external Certification Authorities, a complete lack of flexibility of the certificates with respect to changing and evolving user needs, and the practical need for on-line verification of revocation lists for higher security applications.

This research was motivated by the idea that there are several practical reasons for this limited acceptance and deployment. In order to address some of the weak points of digital certificates, we started to study the notion of StemCerts, which partially overcomes most of the above mentioned shortcomings by allowing the owner of a certificate to modify parts of it without interacting with the Certification Authority that issued it. In the first phase of our research [CG06] we based our design on Chameleon Hash functions [KR00], basically by following the idea of document signature delegation that was originally introduced in [ACdMT05]. Our first proof-of-concept prototype solved most of the problems, though at the cost of introducing some major incompatibilities with existing X.509 v3 software. Subsequently, we developed a second design that maintains all the advantages over standard certificates brought by StemCerts. Furthermore, it guarantees compatibility with the current applications without the need to make any modification to them, albeit with some limitations.

Part of this chapter appeared in "G. Chiola and P. Gasti, StemCerts: Customizable X.509

v3 Certificates for Higher Security, Flexibility, and Convenience", PRISE 2006, Rome [CG06], and in "G. Chiola and P. Gasti, StemCerts-2: Pairs of X.509 v3 Certificates for Greater Security, Flexibility and Convenience", 6th Annual IEEE Consumer Communications & Networking Conference, 2009, Las Vegas (NV) [CG09].

4.1 Chameleon Hash Functions

A chameleon hash function [KR00] is a trapdoor collision-resistant hash function: without knowledge of the trapdoor information, a Chameleon Hash function has the same characteristics of any cryptographic Hash function, such as pre-image and collision-resistance. However, collisions and second pre-images can be easily computed once the trapdoor is known.

A simple construction of a chameleon signature is presented in [KR00]. A potential recipient chooses and publishes a regular discrete logarithm-based public key $y = g^x$, where g is the generator of a cyclic group G and x is the secret key. Then, a user who wishes to sign message m can compute the Chameleon hash value $h = y^m g^r$, where r is an auxiliary integer chosen uniformly at random by the signer. The message m must be a short binary message that has value smaller than the order of the group G when interpreted as the binary expansion of a non-negative integer. However, in order to extend the scheme to arbitrary length messages it is sufficient to first hash the long message using a regular, cryptographic hash function. Notice that if the recipient forges the signature, and two pairs (m, r) and (m', r') become known to the signer (e.g. during a dispute), the signer can recover the secret key x of the recipient from $h = g^m y^r = g^{m'} y^{r'}$, giving $x = \frac{m-m'}{r'-r}$.

This is a highly undesirable outcome for several applications, so in [AdM04] Ateniese and de Medeiros propose a scheme which provide secret key exposure freeness in case of forgery.

Chameleon Hash Functions Without Key Exposure

Ateniese and de Medeiros introduce in [AdM04] the following scheme, on which StemCerts is based.

Key generation: choose a safe prime p of bit length k, and a generator g of the subgroup of quadratic residues $\mathbb{QR}_p$ of $\mathbb{Z}_p^*$. The owner of the certificate picks the secret key x at random from $\mathbb{Z}_q^*$, and its public key is computed as $(g, y = g^x)$.

Hashing scheme: To commit a value m, it is sufficient to choose random values $(r, s) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^*$, and compute

$$e = H(m, r); \text{ and } \text{Hash}(m, r, s) = r - (y^e g^s \bmod p) \bmod q.$$

where H is a collision-resistant hash function, mapping arbitrary-length bit strings to strings of fixed length l.

Collision finding: Let C denote the output of the Chameleon Hash on input the triple (m, r, s). A collision (m', r', s') for a random m' can be found by computing r' and s' such that:

$$e' = H(m', r'); \text{ and } C = r' - (y^{e'} g^{s'} \bmod p) \bmod q.$$

The owner of the certificate chooses a random value m', a random value k' in $\mathbb{Z}_q^*$, and computes $r' = C + (g^{k'} \bmod p) \bmod q$, $e' = H(m', r')$ and $s' = k' - e'x \bmod q$.

4.2 StemCerts

StemCerts lets a certification authority (CA) choose which fields are editable by the owner of a certificate. The available fields are *Distinguished Name*, *Alternative Name* and *Validity Period*. The CA, by signing a certificate which allows the user to modify its *Distinguished Name* or its *Alternative Name*, the certificate can be used pseudonymously. In our prototype implementation we decided to let the owner change a field called "username," which is a free text field. It should contain the information needed to identify a user, such as its name, e-mail address, DNS name, and so on. StemCerts allow the user to set the validity period of a certificate. In this way, the certificate is considered not valid until the user decides to "enable" it.

The main goal of our prototype implementation was to keep the StemCerts structure as similar as possible to the original X.509 v3 structure. This should provide a smooth transition between the current X.509 certificates and our proposed StemCerts. To achieve this result, we added three new extensions: `CHAM_KEY`, which contains the public key for the chameleon hash; `USERNAME`, which specifies the user's identity; and `VALIDITY`, which defines a period in which the certificate can be used [CG06].

`CHAM_KEY` is composed by four fields: p, q, g, and y. They represent the values which defines the public part of the Chameleon key as previously described. These values are chosen by the user when it generates the certificate request, together with the value of x (also needed to calculate y) which must be known only to the user. The values representing the key are chosen by the user and are added to the certificate signing request. When the CA receives a certification request, it is – of course – free to remove any of the editable fields that possibly violate its policy before signing the StemCert. In order to sign the request, The

CA substitutes the values contained in the fields of the `USERNAME` and `VALIDITY` extensions with two publicly known values, then computes the signature of the obtained request as for a standard X.509 v3 signature. In this way the structure of the certificate (the kind and number of editable extensions contained in it), the value of the public Chameleon key and the values of the hash over the editable fields are signed and cannot be modified by anyone. Now the owner of the signed certificate can start using it. To change, as an example, the username value, it has to include a string in the appropriate field of the `USERNAME` extension, then it has to calculate an opportune value for the chameleon hash field.

Even if the certificate is used anonymously, the CA can still know the identity of the owner of the StemCert, because the serial number on it is immutable. In case of dispute, a subject that received a StemCert can show it in Court to a judge who has the power to force the CA to reveal the true identity of the owner. In our model, all the certificate recipients do not need to be completely trusted. They only need to trust the CA which is involved, because it is the only entity which can disclose the certificate owner's real identity. In this way, whoever receives a StemCert can trust the user who sent it even if it uses a pseudonym.

In order to demonstrate the viability of StemCerts we implemented a first proof-of-concept prototype that allowed the creation, the successive customization, and the use of of certificates in the context of a client-server web connection. Our prototype incorporated OpenSSL 0.9.8a library routines for cryptographic functions and X.509 certificate handling. We added the appropriate Chameleon function extensions to OpenSSL code and modified the routines that generate and verify certificates according to our above discussed scheme. The performance of the implemented algorithm were good, generally adding a negligible delay to the handling of standard X.509 certificates, apart for the operations required to generate a new key 1024-bit long [CG06]. Being a probabilistic algorithm, the measured key generation times where quite different for repeated measurements, ranging from two to about ten seconds on our test machine. Considering that this operation must be done only once for each certificate, which is assumed to be quite long-lived, it should not be a concern.

The main concern that still remained after the implementation of this prototype was the need for using a patched OpenSSL library in order to be able to handle our new certificates, thus making them not usable by legacy applications. This observation led us to develop the second version of our scheme.

4.2.1 Signing a Certification Request

The generation of a certificate request for a StemCert is quite similar to that for a standard X.509 certification request. The user needs to generate a Chameleon key of appropriate

length, and then include it in the appropriate extension. After that, it can choose which of the changeable fields are to be included. The *id* and *username* (or *validity*) values contained in the `USERNAME` (resp. `VALIDITY`) extension included in the certification request must be set to a predefined value, in order to clarify that the certificate obtained just after signing the request should not be used. The value of the *hash* field of the two extensions must be set to a non-standard value, namely the hash value computed over the respective fields as described above.

Once the CA receives the certification request it examines it. The CA can clearly choose to remove any of the changeable fields that possibly violate its predefined policy before signing the StemCert. The last check is done by the CA over the *id* value. If it is the predefined value, the StemCert can be signed.

The CA substitutes the values contained in the fields *id*, r, s, and *username* (or *validity*) for the `USERNAME` (`VALIDITY`) extension with standard values, and computes the signature value of the obtained request as for a standard X.509 v3 signature. In this way the structure of the certificate (the editable extensions contained), the value of the public Chameleon key and the values of the hash over the editable fields are signed and cannot be modified by anyone.

If the owner can change the username value, it can choose which string to include in the StemCert just before using it. She can use it as a pseudonymous certificate, hiding its true identity, or in a privacy-aware way, publishing only the personal information that are strictly needed for the particular transaction that implies the use of the certificate. In either ways, it simply puts a new value in the username field, then calculates r and s such that the Chameleon hash function output is equal to the (immutable) hash value contained in the extension.

Analogously, if the owner can change the validity value, it can enable the certificate at its will. The most effective way of using this feature is to keep the StemCert invalid apart for the short periods of time when the owner decides to use it. Right before using the StemCert, it can change the validity field value, making the certificate valid for a short period. Before and after that period, the certificate cannot be validated. On the other hand, a subject who receives a certificate valid for a short period of time, can demonstrate to a third party that there has been interaction between the two users in the period of time specified by the certificate itself.

The use of this field permits to minimize the use of certificate revocation lists: if, with the current standards, the data contained in a certificate are not valid any more (e.g., change of e-mail address), the owner must ask the CA to revoke it and possibly issue a new one, otherwise whoever has a copy of the certificate must consider the obsolete data as valid. Using a changeable validity field, nobody can consider a certificate valid after its usage. The user who makes it valid, implicitly confirms the correctness of the information

contained. However, the specified period must be included in the (longer) one defined by the CA which signed the certificate.

If the certificate contains two (or more) editable extensions, the *id* field of each of them must have the same value. Otherwise, an attacker who knew the values of two editable certificates released from the same user, could build a third certificate containing one extension from the first and one from the second. The *id* is a random number of adequate length (in our prototype implementation it is a 128 bit value).

If another user wants to verify such a certificate, it has to substitute the values contained in the editable extensions as done by the CA during the signing operation. Then it can verify the standard signature value over the modified certificate. If it is valid, the user verifies that the Chameleon Hash value matches to the information contained in the certificate, and whether the *id* values of all the extensions are equal.

4.2.2 StemCerts-2 Design Goals

We designed StemCerts-2 with four goals in mind, after the experience maturated with StemCerts. We wanted to build a certification scheme which can enhance the users' privacy, strengthen the security of the infrastructure, and which is "as compatible as possible" with the existing certification scheme in order to allow gradual and smooth diffusion of our proposed scheme without changing current technology. We were also interested in allowing efficient implementations of StemCerts-2 on smartcards for higher security in mobile applications.

StemCerts-2 is not a strongly anonymous certificate scheme, since it does not provide unlinkability. We show a scheme which provides some form of privacy according to the principle of "limited disclosure": the amount of data being provided for a given purpose should be minimal – not taken into account by current X.509 certificate infrastructure [FB02]. We merely addressed the problem of sensitive information leak in order to defend against, for example, on-line shops which want to gather personal data to send customers unsolicited bulk email, or services which want to build users' profiles for their own reasons.

The second goal was achieved by allowing the certificate owner to change the validity period of the certificate. The validity period can thus be tailored to the single transaction needs, thereby improving the security of current certificates even without consulting revocation lists. The validity period which can be specified by the user must be shorter than and included in the period indicated by the CA.

The third goal was achieved by minimizing the changes to the current X.509 standard certificates. Therefore, despite some limitations, StemCerts-2 can be employed in most of the currently available software applications while requiring no modifications. We also

developed a patch for the OpenSSL library in order to allow a verifier to take advantage of all the expressive power of our scheme. In this way we provide a simple path for a complete migration towards this new kind of certificate.

StemCerts-2 conveys credentials by using a set of two X.509 certificates, called *master* certificate and *slave* certificate. StemCerts-2 allows the owner of the *master* certificate to generate one-time and/or pseudonymous *slave* certificates without the need to interact online with the Certification Authority. The CA can decide which fields of the *slave* certificate the owner is free to change, and which fields must keep the value that was assigned by the CA in the *master* certificate.

As long as the owner of the certificate is the only entity who owns the private key that corresponds to the *master* certificate, she is the only one who can renew and update already elapsed, one-time certificates. The security of this key can be enforced by a proper smart card implementation, thus allowing the use of StemCerts-2 also on untrusted computer platforms, albeit with some limitations.

We believe that these features may help the common user overcome what is seen as the major drawbacks in the current X.509 standard. Our proposed implementation can provide digital certificates with the flexibility that is likely needed by current applications. We implemented the scheme as a small patch to the OpenSSL library, and tested it in a real-world scenario. The results of our experimental implementation demonstrate that the scheme is practical and efficient. Moreover, we devised what we claim is a realistic transition path from current legacy software to our proposed scheme that would allow the two versions of certificates to coexist. We feel that this latter result is the main original contribution of our work.

Indeed, StemCerts-2 proved to be compatible with current networking applications, such as popular web servers and browsers which use X.509 v3 digital certificates. Hence, we claim it would not be difficult to integrate this kind of certificate, even in legacy systems. It is up to the Certification Authority and the user's system administrator to decide which certificates can be used with old applications and which need to be verified by software supporting StemCerts-2.

4.2.3 StemCerts-2 Structure

Our second scheme is based on a pair of X.509 v3 certificates. The first one – called the *master* certificate – is released by a CA, while the second one – called the *slave* certificate – is generated and signed by the owner of the *master* certificate. The signature computed over the *slave* certificate is verified using the public key in the *master* certificate. In this way the user can create many *slave* certificates at its will, without further interaction with the CA that issued the *master* certificate.

Of course, the values contained in the *slave* certificate must be subject to some restrictions. The user can choose a custom value only for a subset of the fields contained in an X.509 v3 certificate, and the subset of changeable fields is chosen by the CA and stated in the *master* certificate. The user does not need to interact on-line with the CA to create a new *slave* certificate, so that StemCerts-2 can be easily supported by portable devices such as smart cards. Note that only the legitimate owner of the *master* certificate can create valid *slave* certificates, as long as the secrecy of private key associated with the public key contained in the *master* certificate is kept.

This scheme may seem similar to "Proxy Certificates" [TWE+04], but it differs from that in many ways. First of all, the aim of our scheme is completely different from the aim of Proxy Certificates. Our main objective is to add flexibility and enhance user's privacy when using X.509 certificates, while proxy certificates are mostly intended for delegation and single sign-on. Moreover, while the *master* and *slave* certificates are intended to be used by the same entity, Proxy Certificates are generally employed by different entities. In our scheme, issuing a *slave* certificate means to specify additional information, while in Proxy Certificates means to delegate a subset of the entity's rights. Proxy Certificates are designed to be used principally in grid environments, while StemCerts are intended to be adopted by end users. These characteristics leads also to different implementation details: we propose three extensions which has very few similarities with the extensions introduced by Proxy Certificates.

In our prototype implementation, the CA can choose which fields are editable in the *slave* certificate, among the following three fields: the *Common Name*; the *Validity Period*; the *Subject Alternative Name*. This setting is maintained in a X.509 v3 certificate extension included in the *master* certificate. All the other fields, apart from the public key and the certificate issuer – which must indicate the *master* certificate – must take the same value of the corresponding field in the *master* certificate.

The *master* certificate must include the standard *BasicConstraints* extension [FB02]. This extension identifies whether the owner of the certificate is a CA and how deep a certification path may exist through that CA. Indeed the *master* certificate is a CA certificate, so the *CA* field must be set to "true". The *pathLenConstraint* field must be set to "zero" to indicate that only an end-entity certificate may follow in the path. In this way, a subject can use a *master* certificate only to sign end-entity certificates (in this case the *slave* certificates), and cannot be used to sign other CA certificates. Our StemCerts-2 implementation also specify the structure for two new extensions, one of which is to be included in the *master* certificate while the other should be found in the *slave* certificate.

4.2.4 Capabilities Extension

The *master* certificate contains an extension, called *capabilities*. It is composed by three boolean fields, and is defined by the following ASN.1 [ASN97] structure:

```
capabilities ::= SEQUENCE {
        CN   BOOLEAN,
        val  BOOLEAN,
        alt  BOOLEAN
}
```

The fields *CN*, *val*, and *alt*, refer to the *Common Name*, *Validity Period*, and the *Subject Alternative Name*, respectively. In a *master* certificate structure they are assigned a boolean value by the CA before the certificate is signed. When set to "true", they indicate that the value of the corresponding field in the *slave* certificate can be changed by the certificate's owner without restrictions. When set to "false", the corresponding field in the slave certificate must take the same value of the corresponding field in the *master* certificate. For every field set to "false" in the capability and for every other non changeable field, the StemCerts-2 verification software should check the consistency between the *slave* and the *master* certificates, and reject the *slave* certificate in case any difference is found.

By marking this extension as "non-critical" the CA can make the *master* certificate compatible with legacy applications. These applications will verify the certificate chain as specified in [FB02]. In this way, we provide a scheme which is immediately usable with the current software, at the cost of some limitations in terms of functionalities. More precisely, a *master*/*slave* pair will appear correct to a legacy verifier even if the *slave* certificate does not satisfy the specified restrictions. Such a "relaxed check" could provide adequate security for some less critical applications. However, there certainly are circumstances in which a higher level of security is required and this limitation cannot be tolerated. In those case, the CA can simply mark the extension as "critical", so that any application which does not recognize it must – according to [FB02] – reject the *master* certificate.

In order to simplify the coexistence with "usual" certificates we would recommend to use a new CA to sign StemCerts-2 *master* certificates. In this way system administrators who manage the public key infrastructure can decide not to use StemCerts-2 certificates in "legacy mode" (even if the extension is defined as "non-critical" by the CA that issues the master certificate) by simply not importing the CA root certificate used to sign the *master* certificates. On the other hand, by importing the new CA root certificate one would implicitly accept the StemCerts-2 scheme either with its full security in case the verification software is updated or with its reduced security in case the legacy X.509 v3

software is kept unchanged.

4.2.5 Hashes Extension

With the *Hashes* extension, the CA can fix a set of values from which a user can insert at its will into a *slave* certificate. The *Hashes* extension contains a list of hash values and an identifier for each hash. Some or all of these identifiers are referred in the *slave* certificate, using the *pads* extension, which will be described later.

```
Hashes ::= SEQUENCE SIZE (1..MAX) OF Hash

Hash ::= SEQUENCE {
        hashID    INTEGER,
        hashValue BIT STRING
}
```

The values of the `hashValue` fields are calculated by the CA from some values provided by the user and some random values in this way:

- the user specifies some values for the fields it is allowed to edit, apart from the validity field
- for each value specified by the user, the CA generates a random number of adequate length
- the CA calculates a hash value of the data obtained by the concatenation of the value specified by the user and the random number generated in the previous step for each value provided by the user

The use of a random value is required to increase the (inherently) low entropy of these fields and to prevent an attacker from guessing their value.

With this extension, a user can choose from time to time if it wants to include a certified information in the slave certificate. If it wants to include a certified information, the user has to specify also the related random value and the hash identifier in the *Pads* extension. Otherwise it can use a value without specifying any random value or hash identifier. In this case the subject which receives the StemCerts cannot be assured of the validity of the user's claim about that values.

This extension should be marked critical only if also the *capabilities* extension is marked as critical.

4.2.6 Pads Extension

This extension is contained in the slave certificate. Its structure is the following:

```
Pads ::= SEQUENCE {
        CNPad     BIT STRING,
        CNHashID  INTEGER,
        subjAltNamePad BIT STRING,
        subjAltNameHash   INTEGER
}
```

When a user adds this extension to a *Slave* certificate, it specifies that some or all the fields it is able to modify are certified by the CA. In order to attest this certification, it has to specify both a hash identifier and the random padding value used by the CA to generate the hash. The identifier must match with the corresponding value contained in the *Hashes* extension of the *master* certificate.

After a verifier has checked the *master* and *slave* certificates with the standard procedure, it has to examine the validity of the fields for which there is a hash value and ID in the *Pads* extension. To do that, it applies the function used to put together the value to be hashed and its random padding value, it hashes the result and then checks the calculated hash value with the value in the master certificate corresponding to the same identifier. If the check is successful, the data inside the certificate were not modified by anyone or wrongfully altered by the owner of the certificate.

4.2.7 StemCerts-2 Usage

The steps needed to obtain a *master* certificate are similar to those needed to get a standard certificate from a CA. Depending on the purpose of the certificate requested, not all the user's personal information have to be included into that request. With StemCerts-2 a user can obtain a certificate which can be updated as her personal data change. The kind of certificate is obtained specifying, in the *Distinguished Name* field of the certification request, only a minimum amount of personal data. The certification authority should add a *Capabilities* extension which lets the user modify only the value of the *Subject Alternative Name*. Another option given to the user is to specify its name in the *Distinguished Name* field of the CSR submitted to the CA. In this way the *master* certificate will contain the user's name as *Distinguished Name* and the certificate policies will let the user update the other data (such as address, and so on) without any further interaction with the CA. With standard certificates the user needs to communicate everything that was initially specified in the CSR, even if it is not a requirement of a particular transaction. With StemCerts-2

the owner can choose to include only a minimum set of required data for that particular transaction.

Pseudonymous certification is obtained when the CA sets to "true" the *CN* field in the *Capabilities* extension of the *master* certificate. In this case the user is allowed to freely edit its identity at will, to enjoy all the benefits of a strong authentication without revealing its true identity during a particular transaction.

When a user wants to obtain a StemCerts-2 which can be enabled at its will, the CA has to include a *Capabilities* extension with the field *val* set to "true" in the *master* certificate. In this way whenever the owner needs to use a certificate to authenticate itself, a new *slave* certificate can be generated, with a new key-pair and a very short validity period. Short after its usage, the *slave* certificate can be discarded, as it is no longer valid. In this context, we think that revocation lists will have less importance for this kind of certificates, provided that the security of the private key used to sign slave certificates is guaranteed by proper measures, such as a smartcard based implementation. The options for master certificates are not necessarily disjoint: it is quite natural to have a certificate which can be activated by a user and which lets its update the identification data in it.

When verifying a *slave* certificate, a user needs to build a chain which is composed, at least, by three certificates: the CA root certificate, the owner *master* certificate (signed by the CA root certificate) and the *slave* certificate. Every time a user creates a new *slave* certificate, it should use a new key-pair in order to enhance the security of the transaction.

The X.509 certificate standard requires the use of revocation lists when verifying a certificate, due to their relatively long lifetime. This is an expensive task that is hardly ever implemented in legacy verification software, thus both violating the standard and reducing the overall level of security. Our proposed solution allows the owner to keep its certificate almost always "elapsed," and to "renew it" for a very small interval of time just before its use. With this customization, the user can create "one time certificates", which can virtually eliminate the need for consulting revocation lists (as long as the *master* certificate has not been revoked).

The *Hashes* and the corresponding *Pads* extension are intended to be used in a context in which the subject who receives the certificate does not trust anyone apart from the CA. In this way, the owner of the certificate can decide which values are to be put in the certificate. The authenticity of that data is verifiable, so in this case StemCerts works almost equivalently to a standard certificate. The obvious advantage is that the user can omit every field which is not strictly required in a given transaction.

When a user requests a new certificate, it can specify more than one value for each editable field. The CA will generate a random padding value and calculate the corresponding hash outcome for each of these fields. At this point the user can choose which of them is to be included in the slave certificate, and identify it with the right id. Even if the certificate is

used anonymously, without the *Pads* extension, the CA can still know the identity of the owner of the StemCerts-2, because the serial number on the master certificate is immutable. In case of dispute, a subject that received a StemCerts can show it in Court to a judge who has the power to force the CA to reveal the true identity of the owner. In this way, whoever receives a StemCerts-2 can trust the user who sent it even if it uses a pseudonym.

The option of having more than a value for the *Common Name* field can be exploited by users who are part of more than one group. At present, they need a certificate for each group the belong to, in order to certify their membership. With our scheme they can request a single *master* certificate, with the option of having a value to use as common name for each group it belongs to.

In order to demonstrate the viability of StemCerts we implemented a proof-of-concept prototype that allows the creation, the successive customization, and the use of certificates in the context of a client-server web connection.

4.3 Implementation

Our prototype implementation is based on OpenSSL 0.9.8a. We added the code needed to manage the *capabilities* extension to OpenSSL. We also modified the routines that verify X.509 certificates in order to implement the check of the restrictions imposed by the CA and expressed in the *master* certificate. In this way, any application which makes use of our patched library can immediately enjoy all the benefits of this new certification scheme.

To test the real-world usability of the certification scheme, we assembled a prototype system containing an Apache 2 web server and two client systems, the first based on a Linux, and the second based on a Microsoft Windows platform. We used a standard X.509 v3 certificate to authenticate the Server, and two StemCerts-2 certificates on the two clients using, in both cases, our modified version of OpenSSL to handle them.

The tests were made using two different web browsers: on the Linux system we used Opera 8.51, while on the Windows platform we used Microsoft Internet Explorer 6.

The server was configured to require a client-side certificate to give access to its pages. The same root certificate used to sign the *master* certificate was imported in the server's certificate list in order to verify the client-side certificates.

Our approach proved to be compatible with all the tested implementations. The user interface is easy to use. The owner of the *master* certificate uses a text-based interfaces which guides it through all the available options for obtaining a new slave certificate. The certificate is automatically added to either Opera or Internet Explorer's list of user certificates and can be used immediately. The overhead introduced by two client-side

certificate with a 1024-bit key is negligible compared to the network latencies.

4.3.1 Smartcard Implementation

The security of the private key associated with the master certificate is critical for the StemCert scheme to offer any advantage to the user. Therefore, a secure storage device must be adopted in case of mobile applications. It would be very convenient to have a device which allows the user to generate short lived slave certificate that could be used by mobile users on untrusted computers. This result can be achieved by using a smartcard for implementing such functionalities of StemCerts For our prototype implementation, we chose a chip card manufactured by Philips, the IBM JCop21id. This smartcard implements Java Card 2.2.1 [SUN00] and offers a cryptographic processor which can compute RSA with key-length up to 2048 bits and perform 3DES encryption and decryption.

In order to provide improve the security of our implementation, we generate the key-pair of the *master* certificate inside the smart card and we do not provide any command to read the corresponding private key. The smart card is programmed to generate a certificate signing request (CSR) for the *master* certificate which have to be signed by the CA. After uploading the signed certificate, the smart card is ready to release new *slave* certificates at user's will. Depending on the memory available on the smart card, our application can save several *slave* certificates. The user can select one of the saved *slaves* as default certificate to download it to a computer.

The smart card can operate in four different modes: non-authenticated, authenticated on partially-trusted environments, authenticated on trusted environments and administrator mode. These modes basically provide a minimum privilege environment for each function. When connected to the card reader, the smart card goes to the non-authenticated mode, in which the user can use only a subset of the available functions:

- **read master certificate**: this function returns the master certificate. The function can be safely executed without authentication because the certificate contains neither secret information nor sensitive user data;
- **read slave certificate**: returns the selected slave certificate;
- **enter one time PIN**: must be used in order to change the current state from non-authenticated to authenticated 1. It requests the user to enter its one-time PIN in order to switch to authenticated mode;
- **unlock PIN**: must be used to unlock the PIN after too many wrong tries.
- **enter PIN2**: same as **enter PIN**, but used to change the current state to authenticated 2

- **unlock PIN2**: same as **unlock PIN**, but for PIN2
- **enter PIN3**: same as **enter PIN**, but used to change the current state to authenticated 3
- **unlock PIN3**: same as **unlock PIN**, but for PIN3
- **logout**: puts the card in non-authenticated mode

The "enter PIN", "enter PIN2" and "enter PIN3" functions increases a counter every time a user enters a wrong PIN. When this counter reaches a predefined threshold, the smart card is automatically locked and does not accept any PIN. In order to unlock the card, the owner must provide the correct unlock code to the associated "unlock PIN function," which resets the PIN to a predefined value. While the PIN can be a short alphanumeric sequence that can easily remembered by the owner, we assume that the unlock code is chosen as a long random combination of characters so that an adversary can guess it correctly with only negligible probability. The smart card stores the retry counter in its EEPROM, so its value is kept when the card is disconnected from the reader.

The functions available in the first authenticated mode include all the same available in the non-authenticated mode plus the following:

- **select slave certificate**: selects one of the saved slave certificates chosen by the serial number specified as a parameter;
- **sign with slave**: this function signs the hash fingerprint provided as parameter using the private key of the selected slave certificate
- **get slave key**: returns, if allowed, the private key of the selected slave certificate
- **show slave certificates**: returns the list of the serial numbers of the saved slave certificates
- **get one time pins**: returns the next three one time pins which the user needs to entree in order to put the card in this state

The functions available in the second authenticated mode, which is intended for slave certificates management, include all the same available in the previous modes plus the following:

- **generate slave certificate**: this function returns a new slave certificate, saves it on the smart card and selects it. It takes as parameters the `common name`, the `subjectAltName` and the `validity period` of the certificate to be generated;

- **make private key exportable**: without calling this function on the selected slave certificate, its private key cannot be read in the first authenticated mode
- **delete slave**: this function lets the user delete a slave certificate when it is no more needed, in order to save storage space on the smart card

The functions available in the third authenticated mode, which is an administration mode which is used to manage the master certificate, are exactly the following:

- **generate new master CSR**: this function re-initializes the smart card, generates a new RSA key-pair which substitutes to the ones stored, and returns a new certificate signing request, which the user should send to the CA
- **import the signed certificate as master**: this function takes an X.509v3 master certificate as parameter, and stores it in the card as the new master certificate

We designed the smart card application so that it can be used in two different ways: a mode that is compatible with legacy applications but that must be used on at least a partially trusted system, and a more robust mode that requires some changes to the application architecture but can also be used on an adversary-controlled system. In order to be compatible with the current applications, the management of certificates can be left to the application. To do that the user can download the *slave* certificate's private key to its computer, after marking it as exportable. At this point the certificate can be imported into any application which supports PKCS#12 file format. In case it is used in a non-trusted environment, the private key of the slave certificate is likely to be compromised after some time. If the certificate is used for authentication, this can be a minor drawback if it was generated to expire in a few minutes. If the key is used to obtain secrecy, an attacker who has control over the system used to communicate could capture the encrypted traffic and decode it later using the key retrieved from the compromised computer. A safe way to use the smart card is to let it use the private key associated with the slave certificate without exporting it. The key can thus be used to gain access to resources without any risk of being compromised even in case the computer is not trusted.

4.4 Final Remarks about the CA

It should be noted that with StemCerts the CA has a new responsibility: in case of a pseudonymous certificate, it is up to the CA to keep the real identity of the certificate owner secret. In fact, if the real identity is not included in the *master* and *slave* certificates, the CA is the only subject (besides the owner itself) who actually knows who the *master* certificate was issued to.

Chapter 5

Inter-domain Authorization Certicates Made Easy

In the age of globalization, organizations have to collaborate to stay competitive so that they can concentrate on their core competencies. A collaboration happens in several forms, like outsourcing and workflow integration just to cite some. Collaboration is an agreement between two or more organizations to achieve a common goal and it can be short-term or long-term. To initiate a collaboration, the organizations share their users and resources. In this chapter we will refer to an organization allowing users from collaborator's domain to perform actions on its resources as *host domain*. A domain is said to be an *independent administrative domain* when the state of its users, resources, and their relations, is readily available within the domain. Change in state of an administrative domain happens when the need for change in access control arises. The internal change in state, at times, may be needed to communicate across collaborating domains. A mechanism that facilitates collaboration should ensure that the internal state changes of a domain should not always necessitate the change to be communicated to peer collaborating domains. That is, the mechanism should allow internal state changes in a host domain while keeping the cost of inter-domain communication for such state changes to a minimum.

As collaborators open up their resources for users from collaborating domains, off-line authentication of the users, and privacy, confidentiality, integrity, non-repudiation of communication between the domains becomes important. These properties can be easily achieved through a public-key infrastructure (PKI) like X.509. As shown in chapter 1, several other collaboration facilitating mechanisms exist that rely on X.509 for authentication and communication security but perform actual authorization decisions through other means. We are interested in finding a solution within X.509 specifications because this is the bare minimum common thing two diverse organizations would have.

The collaboration facilitating mechanism should also keep the modifications needed in intra-domain setup to a minimum in order to quickly gear up for collaboration. Also, such modifications should be reversible so that in case of unsuccessful collaboration, due to unforeseen reasons, the domain quickly reverts back to its pre-collaboration status. This property is very important for successful but ephemeral collaborations. It is paramount to maintain the functional autonomy of collaborating domains during the collaboration period. That is, the fact of being in collaboration state should not hinder a collaborating domain from performing a task that could be performed in its pre-collaboration state. Depending upon the type of collaboration facilitating mechanism, there is a cost associated with functional autonomy of a domain. The cost can be quantized in terms of the number of revocation of assertions (therefore, size of CRL and associated overheads) performed and communicated across domains. The functional autonomy should also allow the domain, from which users are accessing resources of collaborating domain, to degrade or amplify rights over collaborator's resources apart from resource owner doing the same.

Post collaboration, it is equally important to see how quickly a domain can fall-back to its pre-collaboration state. If the modifications to the pre-collaboration setup are kept to a minimum and non-intrusive, it is evident that post collaboration a domain can quickly fall back to its pre-collaboration state.

Having listed the expectations from a collaboration facilitating mechanism we should also note the fact about digital certificate around which we are building our mechanism. Digital certificates are static, off-line verifiable, cryptographic data structures. The static nature of certificates limits the later rights (authorizations/permissions) amplification or reduction and collaborators sharing resources may not always know the complete authorization requirements *a priori*. Off-line verifiability of certificate does guarantee the freshness of assertions made via that certificate. Despite these facts, digital certificates provide tangible assertions which can be relied upon with varying degree of trust and context under which they are used. Revocation or suspension of a single permission/right over a collaborating resource requires appropriate changes in permissions previously conferred on users participating from peer domain.

Capitalizing on the experience developed with StemCerts and StemCerts-2, we investigate to find whether it is possible to re-arrange permissible rights on a shared resource in an inter-domain collaboration so that the number of certificate revocations/issuance are minimized when rights are withdrawn/added. We could address this quandary by introducing two things in our mechanism: segregation of permissions/rights and hierarchy in flow of permission. We named the resulting collaboration facilitating mechanism *InterAC*. Our approach brings a substantial advantage, in terms of number of certificate revocations, autonomy, manageability, compared to existing solutions. These benefits, however, come with a modest added computational cost.

Part of this chapter appeared in "G. Chiola, P. Gasti, L.V. Mancini and V. Patil, Resource

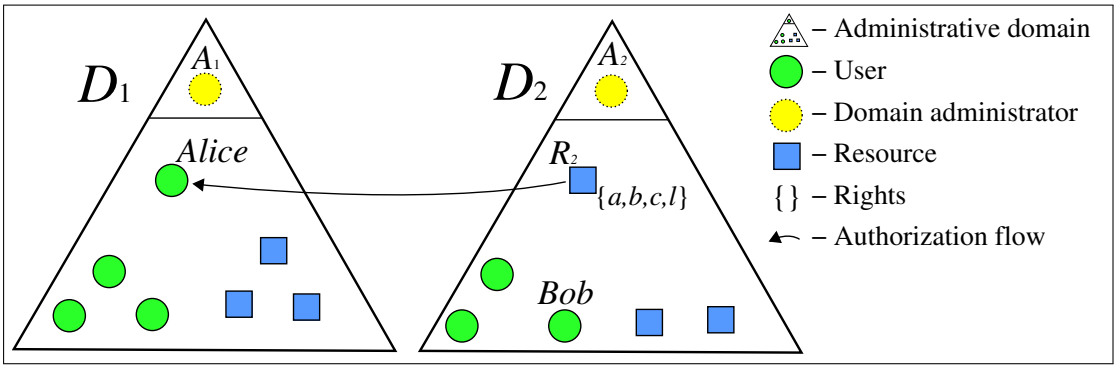

Figure 5.1: Flow of authorization using direct authorization certificate

Management with X.509 Inter-domain Authorization Certificates (InterAC)", EuroPKI 2009, The Sixth European Workshop on Public Key Services, Applications and Infrastructures, Pisa, Italy, 2009 [CGPM09].

5.1 Need for Hierarchy and Segregation of Rights

Before we introduce our proposal, we would like to underline the need for hierarchy in authorization flow and segregation of rights. In a collaboration realized solely using digital certificates, a collaborator sharing its resource will issue a certificate, to users from peer domain, containing appropriate permissible rights on the resource. Assume two collaborating domains D_1 and D_2, where D_2 is offering its resource R_2 for collaboration for user *Alice* from peer domain D_1. Domain administrator of D_2 issues a digital certificate containing permissions $\{a, b, c, d\}$ to *Alice*. When *Alice* needs to access resource R_2, it simply makes an access request along with the authorization certificate. The flow of authorization from R_2 to *Alice* due to the authorization certificate is shown in Figure 5.1. Domain D_2 starts incurring collaboration cost (affecting functional autonomy) when there is a state change in its domain. For example, D_2 needs to withdraw permission d over its resource R_2. This change requires revocation and re-issuance of certificate to *Alice*. The cost is directly proportional to the number of collaborating users having access to resource R_2. Imagine D_2 sharing several other of its resources with D_1 and most of these resources having some permissions that are frequently enabled/suspended. Introduction of a new permission will either require re-issuance of certificates or issuance of separate certificates containing new permission.

To reduce the collaboration costs to participating domains and to retain their functional autonomy we propose a novel approach in which we segregate the permissions on collaborating resource into static and dynamic sets. Static permission are those permission that

are less likely to be withdrawn by resource administrator for a relatively longer period as compared to dynamic permissions that may be suspended (temporarily or permanently) frequently. We also introduce a level of indirection in the flow of authorization flowing from resource to its collaborating users. In the next section, we give the details of our approach with the help of a running example.

5.2 Dynamic Inter-domain Authorization via X.509

In this section, we explain our collaboration facilitating mechanism *InterAC*. *InterAC* is purely within X.509 specifications. Under *InterAC* we have designed a *non-critical* extension to X.509 digital certificate which we detail in section 5.6. Through this extension we allow segregation of permissions on a collaborating resource. *InterAC* uses its type of certificates for the following three purposes:

- for subject binding,
- to define an ACL over resource, and
- as a collaboration agreement.

Using such certificates, *InterAC* allows a domain administrator to initiate collaboration and define the flow of authorization over its resource from collaborating users from peer domain. In Fig. 5.2 one such flow of authorization from resource R_2 to user *Alice* is shown. In short, to access a resource from a collaborating domain, a user need to compose a chain of certificates that proves a valid flow of authorization from the resource to the user. The chain composition and evaluation algorithm is explained in Section 5.4. Let us explain the syntax of *InterAC* certificate and semantics behind it.

Let us denote the *InterAC* digital certificate as CERTIFICATE. Let D_1 and D_2 be two independent administrative domains willing to collaborate. That is, for example, D_2 agreeing to share its resources with the users from domain D_1 as shown in Fig. 5.2. To share resource R_2, A_2 – domain administrator of D_2 – issues a special type of certificate to R_2 that R_2 will use as an ACL for requests coming from collaborating users. The short hand notation of this CERTIFICATE is: $R_2 \longrightarrow R_2 \boxed{\{a,b,c\},\{l,m,n\}}$. Where, $\{a,b,c\}$ are the set of static permissions and $\{l,m,n\}$ are the set of dynamic permissions. As a next step, A_2 takes ownership of this resource: $R_2 \longrightarrow A_2 \boxed{\{a,b,c\},\{l,m,n\}}$.

To initiate a collaboration with domain D_1, A_2 confers rights on R_2 to A_1 – the domain administrator for D_1. Therefore, $A_2 \longrightarrow A_1 \boxed{\{a,b,c\},\{l\}}$. In turn, A_1 issues a CERTIFICATE to *Alice* so that *Alice* contributes to collaboration: $A_1 \longrightarrow$ *Alice* $\boxed{\{*\},\{*\}}$. The

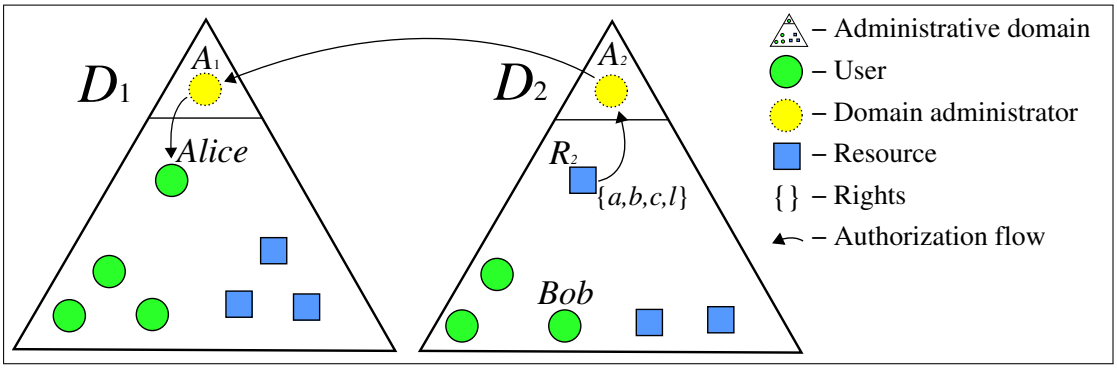

Figure 5.2: Flow of authorization using indirect authorization certificate

wild character in permission set has a special meaning under *InterAC*. It signifies that the decision to grant permissions to the subject of the certificate has been deferred or, in other words, the subject of the certificate can perform all possible actions provided that the subject comes up with a valid proof (certificate chain showing flow of authorization from resource to the requester). Intuitively, Alice can perform $\{a, b, c, l\}$ rights on resource R_2 with the above set of certificates. The computation of effective rights of a requester are done by taking a positional intersection[1] over permissions present in the certificates used for composition of proof. This indirect flow of authorization (hierarchy) from the resource to *Alice* allows each intermediate principal to decide the actual set of permissions *Alice* will have over resource R_2, at any given time. In the following section we shall see how this introduction of hierarchy and segregation of permissions into static and dynamic set contributes to autonomy and manageability of collaboration.

It is interesting to note that as there is no mention of exact rights *Alice* has been given, the same certificate can be used by *Alice* to participate in collaborations with other domains. Since the extension is *non-critical*, the certificate can be used as an identification certificate by *Alice* for purposes other than collaboration. As no rights are specified inside *Alice*'s certificate, privacy violations do not happen when this authorization certificate is used for just authentication purpose.

[1]Though the permissions in static and dynamic sets is treated similarly, i.e., a simple intersection across respective sets in the CERTIFICATE chain, it is important that across all the participating collaborative domains the CERTIFICATES should be issued with a consistent position of permission set – static permission set followed by dynamic permission set.

5.3 Bringing Autonomy and Manageability to Collaborators

Continuing with the setup shown in Fig. 5.2 we will introduce more scenarios to explain utility of *InterAC* towards autonomy and manageability. Let us begin with an example of effective rights computation for user *Alice* with a sample scenario.

In the following D_2's domain administrator A_2 is preparing R_2 for collaboration with $\{a, b, c\}$ permissions.

$$R_2 \longrightarrow A_2 \boxed{\{a, b, c\}, \{\ \}}$$

where, $\boxed{\{a, b, c\},\ \{\ \}}$ is the authorization string in the CERTIFICATE used as an ACL on resource R_2. Note that D_2 has abstained from conferring permission c over resource R_2 to its collaborator. Let the following be the CERTIFICATE denoting the collaboration agreement between domain D_1 and D_2, where domain D_2 is offering its resource to the users from domain D_1;

$$A_2 \longrightarrow A_1 \boxed{\{a, b\}, \{*\}}$$

In a slight modification to the setup in domain D_1, we introduce two roles G_1 and G_2 and let *Alice* be part of G_1, for time being. Therefore;

$$A_1 \longrightarrow G_1 \boxed{\{a\},\ \{*\}}$$
$$A_1 \longrightarrow G_2 \boxed{\{b\},\ \{*\}}$$
$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}}$$

Therefore, *Alice* constructs the following CERTIFICATE chain to access R_2

$$R_2 \longrightarrow A_2 \boxed{\{a, b, c\},\ \{\ \}}$$
$$A_2 \longrightarrow A_1 \boxed{\{a, b\},\ \{*\}}$$
$$A_1 \longrightarrow G_1 \boxed{\{a\},\ \{*\}}$$
$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}}$$

And the effective permissions at the disposal of user *Alice* are $\{a\}$, upon positional intersection of permissions present in the CERTIFICATE chain;

$$\textsf{static}\ permissions = \{a, b, c\} \cap \{a, b\} \cap \{a\} \cap \{*\} = \{a\}$$
$$and,\ \textsf{dynamic}\ permissions = \{\ \} \cap \{*\} \cap \{*\} \cap \{*\} = \{\ \}$$

Should A_1 decide *Alice* to avail permission b, G_2 issues the following to *Alice*

$$G_2 \longrightarrow \mathit{Alice} \boxed{\{b\},\ \{*\}}$$

Having shown the CERTIFICATE chain construction and evaluation of effective permissions due to the chain, we would like to show you how each principal in the authorization hierarchy can amplify or degrade effective rights of *Alice*.

5.3.1 Rights Amplification

Rights (alternatively referred as permissions or authorizations) can be amplified, that is, extra permissions can be added to the existing permission-set, by either resource controller or collaboration administrators (on either side of the collaboration). The only condition for rights amplification is the entity performing amplification operation itself should have the permission to be amplified. Following are the three instances of rights amplification that amplify the rights of *Alice*.

By resource controller. Resource controller is the fundamental authority to decide the actual set of permissions possible over the resource. Let m be a new permission that resource controller wants to make available to its collaborators. To do so, the resource controller updates its ACL (CERTIFICATE) with the following.

$$R_2 \longrightarrow A_2 \boxed{\{a, b, c\},\ \{m\}}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing R_2 in domain D_2 becomes;

$$\begin{aligned} R_2 &\longrightarrow A_2 \boxed{\{a, b, c\},\ \{m\}} \\ A_2 &\longrightarrow A_1 \boxed{\{a, b\},\ \{*\}} \\ A_1 &\longrightarrow G_1 \boxed{\{a\},\ \{*\}} \\ G_1 &\longrightarrow \mathit{Alice} \boxed{\{*\},\ \{*\}} \end{aligned}$$

And the effective permissions at the disposal of *Alice* are $\{a, m\}$, because;

$$\begin{aligned} \textsf{static}\ \mathit{permissions} &= \{a, b, c\} \cap \{a, b\} \cap \{a\} \cap \{*\} = \{a\} \\ \mathit{and},\ \textsf{dynamic}\ \mathit{permissions} &= \{m\} \cap \{*\} \cap \{*\} \cap \{*\} = \{m\} \end{aligned}$$

By host domain administrator. A_1 can perform rights amplification for *Alice* by issuing the following CERTIFICATE.

$$A_1 \longrightarrow G_1 \boxed{\{a, b\},\ \{*\}}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing R_2 becomes;

$$\begin{aligned} R_2 &\longrightarrow A_2 \boxed{\{a,b,c\},\ \{\ \}} \\ A_2 &\longrightarrow A_1 \boxed{\{a,b\},\ \{*\}} \\ A_1 &\longrightarrow G_1 \boxed{\{a,b\},\ \{*\}} \\ G_1 &\longrightarrow \mathit{Alice} \boxed{\{*\},\ \{*\}} \end{aligned}$$

And the effective permissions at the disposal of *Alice* are $\{a,b\}$, because;

$$\begin{aligned} \textsf{static}\ \mathit{permissions} &= \{a,b,c\} \cap \{a,b\} \cap \{a,b\} \cap \{*\} = \{a,b\} \\ \mathit{and},\ \textsf{dynamic}\ \mathit{permissions} &= \{\ \} \cap \{*\} \cap \{*\} \cap \{*\} = \{\ \} \end{aligned}$$

By peer domain administrator. A_2 can perform rights amplification for the users from its collaborating domain by issuing the following CERTIFICATE

$$A_2 \longrightarrow A_1 \boxed{\{a,b,c\},\ \{*\}}$$

Of course, this amplification will not be reflected in domain D_1 until D_1 does further rights amplification.

5.3.2 Rights Degradation

In this sub-section we show rights degradation, which is similar to rights amplification but the permissions will be removed from the existing set of permissions available to the principal that is performing rights degradation. Before proceeding to the examples of rights degradation let us bring back the CERTIFICATE states to the pre-amplification steps performed in previous sub-section.

By resource controller. As mentioned before, resource controller is the fundamental authority to decide the actual set of permissions possible over the resource. Let a be the permission that resource controller wants to make unavailable to its collaborators. To do so, the resource controller updates its ACL (CERTIFICATE) with the following.

$$R_2 \longrightarrow A_2 \boxed{\{b,c\},\ \{\ \}}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing the resource

R_2 becomes;

$$R_2 \longrightarrow A_2 \boxed{\{b,c\},\ \{\ \}}$$
$$A_2 \longrightarrow A_1 \boxed{\{a,b\},\ \{*\}}$$
$$A_1 \longrightarrow G_1 \boxed{\{a\},\ \{*\}}$$
$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}}$$

And the effective permissions at the disposal of *Alice* are { }, because;

$$\text{static } permissions = \{b,c\} \cap \{a,b\} \cap \{a\} \cap \{*\} = \{\ \}$$
$$and, \text{ dynamic } permissions = \{\ \} \cap \{*\} \cap \{*\} \cap \{*\} = \{\ \}$$

By host domain administrator. A_1 can perform rights degradation for *Alice* by issuing the following CERTIFICATE

$$A_1 \longrightarrow G_1 \boxed{\{\ \},\ \{*\}}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing R_2 becomes;

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\},\ \{\ \}}$$
$$A_2 \longrightarrow A_1 \boxed{\{a,b\},\ \{*\}}$$
$$A_1 \longrightarrow G_1 \boxed{\{\ \},\ \{*\}}$$
$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}}$$

And the effective permissions at the disposal of *Alice* are { }, because;

$$\text{static } permissions = \{a,b,c\} \cap \{a,b\} \cap \{\ \} \cap \{*\} = \{\ \}$$
$$and, \text{ dynamic } permissions = \{\ \} \cap \{*\} \cap \{*\} \cap \{*\} = \{\ \}$$

By peer domain administrator. A_2 can make use of rights degradation facility to achieve an important aspect required in collaboration – temporary suspension of collaboration. To do so, A_2 issues the following CERTIFICATE

$$A_2 \longrightarrow A_1 \boxed{\{b\},\ \{\ \}}$$

Therefore, the authorization proof (CERTIFICATE chain) by *Alice* for accessing R_2 is;

$$R_2 \longrightarrow A_2 \boxed{\{a,b,c\},\ \{*\}}$$
$$A_2 \longrightarrow A_1 \boxed{\{b\},\ \{\ \}}$$
$$A_1 \longrightarrow G_1 \boxed{\{a\},\ \{*\}}$$
$$G_1 \longrightarrow Alice \boxed{\{*\},\ \{*\}}$$

And the effective permissions at the disposal of *Alice* are { }, because;

$$\textsf{static}\ permissions = \{a,b,c\} \cap \{b\} \cap \{a\} \cap \{*\} = \{\ \}$$
$$and,\ \textsf{dynamic}\ permissions = \{*\} \cap \{\ \} \cap \{*\} \cap \{*\} = \{\ \}$$

Several combinations of rights amplification and degradation can be engineered by resource controller and corresponding domain administrator, independently or collectively to achieve desired effects in the availability of permissions to the users from collaborating domain.

5.3.3 Rights Suspension

This operation is a special instance of rights degradation. The resource controller can take down the resource temporarily for various reasons by issuing the following CERTIFICATE.

$$R_2 \longrightarrow A_2 \boxed{\{\ \},\ \{\ \}}$$

Based on the internal dynamics (state changes) of the domain sharing resources, domain administrators can roughly estimate life expectancy (certificate validity period) of CERTIFICATES at different hierarchy levels. We assume that domain administrators issue/revoke *InterAC* certificates to users and resources. The domain administrators are also responsible to initiate the collaboration (by issuing authorization certificate to peer domain administrator). We also assume that the semantics of permissions embedded inside the *InterAC* certificate issued for collaboration initiation is agreed upon. PKI Resource Query Protocol (PRPQ) [PRP] is a promising utility for seamless, dynamic integration of resources across independent administrative domains.

5.4 Chain Composition and Evaluation

In this section we provide an algorithm to compute a valid CERTIFICATE chain. We assume that the users of a collaborative domain have been made available with the set of

CERTIFICATES that affect the permission-set of the user. The onus of authorization proof generation is on the requester of the resource. We continue referring to principals (R_2, A_1, *Alice*, etc.) from the scenarios presented in previous sections.

Composition of certificate chain: Authorization proof construction (performed by requester)

certificate validation – Discard CERTIFICATES whose validity has expired or stand revoked.

Filter certificates – Include CERTIFICATES containing the permission for which request is being made in its authorization string. Discard CERTIFICATES with $\{\ \}, \{\ \}$ in its authorization string (i.e., empty static and dynamic permission-sets).

Construct directed graph – For each principal (issuer or subject of a certificate) add a vertex to the graph. For each CERTIFICATE put a directed edge originating in the "issuer" vertex and ending in "subject" vertex.

Find path – Find all possible paths starting in the vertex denoted by the principal "resource controller" (i.e., R_2) and terminating in the vertex denoted by the principal "requester" (i.e., *Alice*)

Purge paths – Discard paths in which the positional intersection of the permission under consideration leads to an empty set

If no paths are left after **Purge paths** step, a valid authorization proof is not available.

Evaluation of certificate chain: Authorization proof verification (performed by verifier)

certificate validation – Check CERTIFICATES in authorization proofs for their validity and revocation status.

Intersection – Take positional intersection over the authorization strings present in the CERTIFICATES of the authorization proofs.

Access will be granted with effective permissions evaluated upon positional intersection.

5.5 Comparative Analysis

In this section we would like to compare our mechanism with a mechanism that does not treat permissions as we do. The closest contender of such an approach is the X.509

attribute certificate framework defined in [ITU05] that provides the foundation upon which the Privilege Management Infrastructure (PMI) can be built. This framework has been the most commonly used approach to realize inter-domain authorizations.

The PMI approach for inter-domain authorization has following shortcomings: i) size of ARL (attribute revocation list) keeps on growing as the number of collaborating domains of a host domain go on increasing when the state of collaborating domain changes. ii) each collaborating domain of a host domain necessitates issuance of attribute certificates to the users of host domain – collaboration-specific certificates that expire upon completion of collaboration and may be added to ARL. iii) such an approach of embedding exact set of permission-set into user's certificate leaves no scope for later rights amplification or degradation. iv) and, lack of functional autonomy and manageability.

Intuitively, under *InterAC* the number of users in peer domain do not *proportionally* influence the cost of any operation performed towards collaboration. That is, the cost to establish/break a collaboration or to do rights amplification/degradation/suspension is constant.

Table 5.1 compares our approach with the traditional PMI approach using the example discussed in section 5.2, as a test-bed; where n is the number of collaborating users and h is the length of CERTIFICATE chain or depth of authorization flow hierarchy. We assume that A_1 already issued the appropriate certificates to its users and that there has been no previous interaction between domains D_1 and D_2. The comparison also assumes that the "push" model is adopted for the PMI [FH02]. The computational cost introduced by *InterAC* on resource R_2 is greater than the computational cost in traditional PMI. This is because, in PMI the authorizations are asserted in one or few attribute certificates, while in our approach the authorizations must be calculated by positional intersection of the authorizations contained in the CERTIFICATE chain. The actual computational overhead is given in section 5.6. We feel the cost overhead is justifiable given the numerous advantages our mechanism brings in for collaboration.

5.5.1 *InterAC* in Perspective of Policy-based Mechanisms

To facilitate collaboration among independent administrative domains, two other distinct research tracks exist: i) policy-based languages (e.g., [BFIK99, Sec05, XAC05, Web]) that allow to capture collaboration requirements, and ii) extensions to RBAC model (e.g., [CTWS02, CDD+04, LN99, JBBG04, LMW02a, LMW+02b]). These approaches have more expressive power as compared to *InterAC*. We say so because *InterAC* does not provide a language to capture context-aware decisions, neither it provides fancy constructs like separation-of-duty as under RBAC family. We refrained from devising an accompanying language in our proposal because all the above mentioned policy/model-based proposals

	InterAC	**PMI-based mechanisms**
Cost of collaboration initiation	$O(1)$ Issuance of certificate by a domain administrator to peer domain's administrator. It is assumed that collaboration-independent *InterAC* certificates have been already issued in participating domains.	$O(n)$ [a] Since the authorization certificates are specific to a collaboration, new certificates need to be issued each time a new collaboration is initiated.
Cost of incoming authorization request verification	$O(h)$ [b]	$O(1)$
Cost of rights amplification or degradation	$O(1)$	$O(n)$
Cost of rights suspension	$O(1)$	$O(n)$ - by revoking all user certs $O(1)$ - by updating the resource ACL, which also disables the access to the resource for users in host domain
Cost to revert to pre-collaboration state	$O(1)$	$O(n)$

[a] n – number of participating users from a collaborating domain
[b] h – authorization hierarchy or length of the CERTIFICATE chain

Table 5.1: *InterAC* vs. PMI-based mechanisms *w.r.t.* certificate issuance / verification / revocation cost to a collaborating domain

face interoperability issues. We observe that the minimum common that the administrative domains willing to collaborate have is a PKI (digital certificates). *InterAC* provides the basic requirements of collaboration purely through non-intrusive certificate extension. The policy/model-based mechanisms for collaboration use digital certificates as assertions and take access control decisions based on such set of assertions and plausibly other contexts. The *InterAC* certificates can also be used as assertions thus enriching the higher level policy/model-based approaches.

5.6 Implementation Details

The ASN.1 notation for our X.509v3 certificate extension is depicted in Fig. 5.3. `Static`

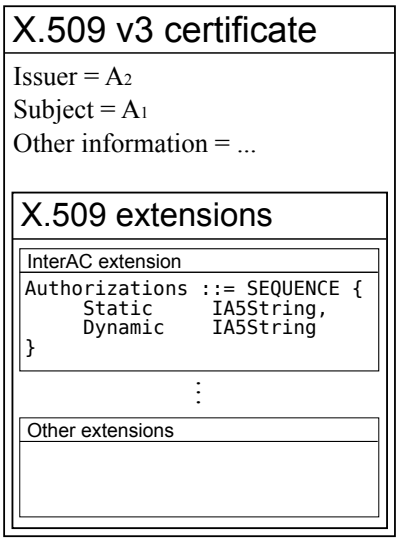

Figure 5.3: Sample X.509v3 certificate with the *InterAC* extension

and `Dynamic` are two strings which hold the permissions that are less prone (non-volatile) and more prone (volatile) to frequent modifications, respectively. Effective permissions of a certificate's subject are captured in two distinct sets. Depending upon the requirements, authorization delegation authority may include either a "$*$" or a " " (null) or a comma-separated list of permissions in any of the set. For example, `static = a,b,c` means that the set of non-volatile permissions for the certificate's subject are $\{a, b, c\}$, where a, b and c represent three different permissions.

5.6.1 Performance Analysis

Our prototype implementation [GP06] is implemented in C (gcc 4.1.2) on a Pentium III (933MHz processor with 512MB RAM) hardware running GNU/Linux (kernel 2.6). The cryptographic primitives are supported by OpenSSL library (0.9.8e). The certificates used for measuring performance results are generated with 1024-bit RSA public keys. An inter-domain authorization request consists of a well-formed sequence of digital certificates as a proof of credentials. The resource controller verifies such certificate chains before granting access. The algorithm to perform verification is given in Section 5.4. The performance results of our approach against PMI-based approach is summarized in the graph shown in Fig. 5.4.

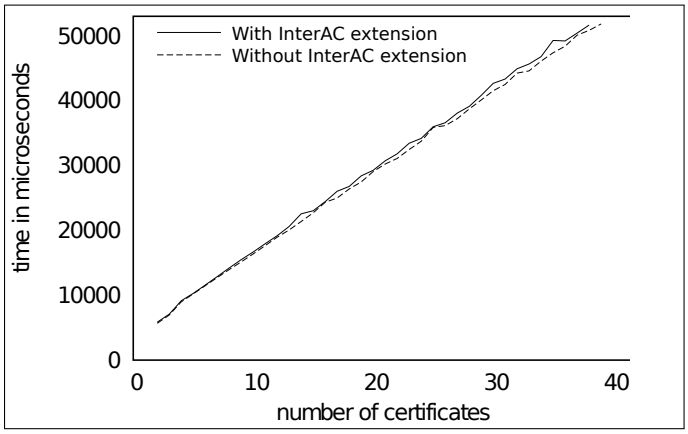

Figure 5.4: Time to evaluate certificates with different types (*InterAC* and non-*InterAC*) of authorization extensions

The graph is plotted for two different authorization proof chains consisting certificates with different extension types: i) typical authorization extension (i.e., without segregated permission-sets), and ii) our extension. The slight increase in the computational cost for our approach is justifiable by the benefits it provides. Taking a closer look at the difference between the two values we observe that it is around 1% on an average. For chains with realistic length (i.e., composed of 15 certificates or less), the actual computational cost overhead is around 0.5 ms in our operating environment.

Chapter 6

Practical Deniability: DenFS Deniable Cloud Storage

The goal of cloud computing is to provide CPU, storage, network bandwidth, and virtually unlimited scalability at low cost. In addition, cloud services provide full availability, i.e., users can access their data from any connected machine. Cloud users do not need to worry about backups or unexpected costs: if a component fails, it is the provider's responsibility to replace it and make the data available using replicas [Mir08]. Armbrust et al. identify three aspects which characterize cloud computing [AFG+09]: 1. the illusion of infinite computing resources available on demand, and therefore the elimination of the need to plan ahead for provisioning; 2. the elimination of an up-front commitment by users; and 3. the ability to pay for use of computing resources on a short-term basis as needed. Cloud computing is said to maximize benefits of scale [AFG+09][Vou08].

Amazon's Simple Storage Service (S3) [Ama] is among the best-known storage provider, and is used both by other service providers to store customer's data, and by end users. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data. This interface is specifically designed to be easy to integrate into new and existing applications.

There are several cloud storage providers which rely on Amazon S3 to store user's data, and construct service with high value on top of it. A notable example is Dropbox [Dro]. Dropbox is software that syncs files online and across different computers. Documents are also available through a web interface, which is also accessible from devices. Dropbox provides an easy interface for sharing documents with other users. A copy of the shared documents is stored in the cloud, and Dropbox takes care of synchronizing the local copy of each user with the on-line copy. This provides a very powerful tool for collaboration: users don't suffer delays introduced by the network when viewing or editing documents,

since they always access the local copy. Updates are sent and received in background and users are quickly notified when the content of a shared file changes.

With cloud storage services, encryption becomes crucial to protect data privacy, at least because the cloud provider has full access to user's data. The goal of traditional encryption is to preserve privacy of data communication or storage in presence of passive adversaries who can observe the communication or access stored data. Deniable encryption, on the other hand, was introduced by Canetti et al. [CDNO97] as a mechanism that allows one to maintain data privacy in presence of active adversaries who have the ability to coerce data senders or recipients into opening observed ciphertexts. That is, assume that an adversary first observes a ciphertext and then approaches the sender and asks its to reveal all of the random choices and secret keys (if any) used in generating the ciphertext. This means that the corresponding plaintext will be exposed to the coercer. A deniable encryption scheme is then an encryption scheme with an additional property that allows the sender to open the ciphertext in such a way as to reveal a different version of the plaintext than the one originally used in producing the ciphertext. The above type of deniable encryption, where the ciphertext sender is coerced and opens the ciphertext, is called *sender-deniable* encryption. Similarly, *receiver-deniable* encryption can be defined as a deniable encryption scheme where ciphertext recipient is coerced into opening the ciphertext. Additionally, *sender-and-receiver deniable* schemes can be defined to combine both of the above properties. We refer to the plaintext intended to the recipient as the real message and the plaintext that the coercer sees after opening of the ciphertext as the fake message.

The main motivation of the work presented in this chapter comes from the desire to explore whether efficient deniable encryption is possible in practice. In particular, we are concerned with practical applications of deniable encryption and consider the following scenario as our main target functionality. An adversary obtains access to a laptop computer, the owner of which keeps the stored data encrypted. Some of the encrypted data is shared with one or more collaborating users, who exchange the updated files through the Internet. The adversary, who may have intercepted update messages, forces the owner to decrypt the data by revealing any key material and necessary auxiliary information. We would like the owner to have the ability to open the data in two different ways through a usable mechanism, e.g., by entering a password that will unlock the key material and decrypt the data. That is, the data owner can be in possession of two different passwords: one uncovers the true data stored on the laptop and the encrypted messages sent and received from other users, and another uncovers a different (and meaningful) version of the data stored on the disk.

The contributions of this chapter are as follows: We show how a sender-receiver deniable plan-ahead public-key encryption scheme can be built from a standard encryption scheme using it as a black box. Our construction has a high throughput (i.e., linear in the size of the ciphertext) for messages that can be deniably communicated. We also show instantia-

tions of deniable encryption using ElGamal and Boneh-Franklin Identity-Based Encryption (IBE) schemes. An essential component of this chapter is the design and implementation of DenFS, a distributed file system that allows a group of collaborators seeking deniability to view, edit and exchange documents as if they were working on a standard filesystem. Finally, we give extensions to our work that allow us to increase the bandwidth of the deniable channel and provide other efficiency improvements.

6.1 Deniable Cloud Storage and Deniable Multiparty Computation

The deniable scheme and the optimizations introduced in chapter 3 allow the construction of several deniable public-key protocols. In this section we give a hint of how our solution allows a group of interacting parties to maintain data privacy in presence of an adversary who can adaptively coerce them into opening observed ciphertexts. Our scheme lets the participants disclose to the coercer a plausible but non-sensitive transcript of their interaction, preserving the privacy of the real messages. We give two examples of as potential applications for our scheme: deniable cloud storage and secure deniable multiparty computation.

Deniable Cloud Storage: *cloud computing* is a powerful tool for improving productivity and ease of collaboration among users. Additional advantages are high availability and cost savings compared with traditional infrastructures. It is not surprising that there is a growing interest towards these technologies, although privacy and security concerns are still mostly an open problem (see, for example, [Pea09]).

On-line storage is one of the services often found "in the cloud". Many companies are offering practical storage and synchronization services at a very low cost [Ama][Goo] or even for free [ADr][Dro]. Thanks to these services, users can store their documents on line and easily synchronize them between a laptop and a desktop computer, or other users. We focus our discussion on Dropbox, since it easy to use; it integrates well with Windows, Mac OS, X and Linux; it offers 2 GB of space for free; and it is well-known among users. Dropbox comes in the form of a desktop software, which automatically synchronizes a local folder with an on line folder: every time a user adds, removes, or modifies a file in its local folder, these changes propagate to Dropbox server. If the user owns another machine running the Dropbox software, the changes are automatically propagated to that machine as well.

Dropbox lets users share their folders with other Dropbox users. Once other members are added to a shared folder, any change made to its contents by any of its members will instantaneously propagate to everybody who has access to that folder. In this section we

focus on folder sharing among several collaborating users, more specifically, a user shares one of its folders and all of the participants store files encrypted for other participants. New users can join the shared folder anytime.

Dropbox uses Amazon S3 to store users' files. Before sending the files to S3, Dropbox encrypts them using AES. Unfortunately, the encryption keys are stored on Dropbox servers, therefore Dropbox administrators have complete access to all users' files.

In order to protect their privacy, users can encrypt their files prior to sending them using publicly available encryption software, like the open source GPG [GPG]. Unfortunately GPG does not provide deniability: traditional encryption software is designed to protect against adversary without access to the decryption key, while in our model we assume that the coercer will be able to obtain access to the decryption key, either by legal – e.g., court order, subpoena [Ser] – or illegal means.

In order to solve this problem, a user can use a disk encryption tools with support for deniability, like TrueCrypt: it creates a TrueCrypt standard volume containing a TrueCrypt hidden volume [Hidb] and then it shares the encryption keys for both volumes with all members of the shared folder. This solution is clearly not acceptable in our scenario where we need access control: the owner of the shared folder may not want all users to be able to access portions of content encrypted by other members, unless explicitly specified. In order to overcome this, several hidden volumes can be created and shared using distinct secret keys. Unfortunately, if we let users share symmetric keys for access control then key management soon becomes a nightmare.

With our public-key deniable scheme we could provide a reasonable solution to the problems above. In particular, collaborating users publish their public keys that will be used by the owner of the shared folder in Dropbox to deniably encrypt files. Any participant that is approached by the coercer and forced to decrypt one or more shared files can just open the corresponding non-deniable (*fake*) files without revealing any sensitive information.

Secure Deniable Multiparty Computation: secure multiparty computation [Yao82] allows n players to compute an agreed function of their inputs in a secure way, where the correctness of the output and the privacy of the players' inputs are guaranteed, even if some players cheat. However, if a coercer approaches one or more players after the protocol is completed, it can force them to reveal their inputs and the computed value. In this scenario, the only way for a user to avoid exposing its input and results to the coercer is to avoid participating in the computation *tout court*.

By using our deniable encryption scheme to encapsulate the message exchanged between the players, secure and deniable multiparty computation schemes can be easily constructed: all players can agree on running two "concurrent" versions of the computation at the same time, the first one with, non-incriminating and non-sensitive inputs and second with the "correct inputs". A player can be forced by the coercer to disclose its inputs and the result

of the first – non-incriminating – computation. However, the coercer cannot distinguish the second computation from a random computation for which both inputs and output are not known. Clearly, the number of messages required to perform the deniable computation must be less than or equal to the number of messages required to perform the non-deniable computation.

In this chapter we focus on deniable cloud storage and we leave the detail of the secure deniable multiparty computation as an open problem.

6.2 DenFS - A Deniable Shared Filesystem

In order to explore the practicality of our solution, we designed and implemented a prototype application. The purpose of our prototype is to implement DenFS, a filesystem which transparently and deniably encrypts all its content, and stores it *in the cloud*. In this way we intend to provide one of the basic tools for coordinating a collaboration between users. One of the design goals of DenFS is to provide a seamless experience, allowing users to interact with objects stored in the deniable filesystem as if they were stored on a standard filesystem.

6.2.1 High-level Description

DenFS allows users to read, write, create or move any file or directory using standard tools, like a file manager or any command line program. Users can edit files from the deniable folder with their usual applications. The content of the deniable filesystem can be shared with an arbitrary number of users. Different users may have a different view of the deniable filesystem, since DenFS shows only the encrypted files for which the appropriate decryption key was provided at mount time. The synchronization between files and folders in DenFS and the cloud storage happens in background. The data is also stored locally in encrypted form. This works as a buffer, which hides all the delays introduced by the network when reading or writing a file and allows users to create, edit, and view files when they are not connected to the Internet. After reconnecting, the synchronization service takes care of updating both the local cache and the on-line repository. Synchronization conflicts are shown to the user who can decide how to deal with them. File updates propagate with a speed proportional to users' available bandwidth. There is no performance downside when a relatively large number of users share the same folder, because each modification is sent only once to the cloud and then pushed concurrently to each user.

We designed our prototype to be efficient, secure, and easy to use. To mount a DenFS filesystem, a user simply specifies the mount point, the backend directory, and a password.

The backend directory is the directory where encrypted files are stored. In order to synchronize files between users, the backend directory must be inside a Dropbox folder and must be shared. Each user's secret key is encrypted with a symmetric algorithm; the decryption password is prompted when mounting the filesystem. If a user does not enter the decryption password, it will only be able to encrypt new files. All filesystem operations, like reading or writing a file, are automatically mapped to the backend folder and to the cloud.

Before mounting the filesystem, each user must create a directory containing sensitive looking files which will be used to fill the non-deniable part of a deniable encryption. This directory is called *non-deniable pool*. DenFS can be mounted in one of the two modes of operation available: *deniable* and *non-deniable*. When mounting the filesystem in non-deniable mode, random data is used to fill the deniable part of the ciphertext. In deniable mode, the non-deniable part is filled with the encryption of a file selected from the *non-deniable pool*.

When the file system is mounted in non-deniable mode, reading, writing, or copying a file only involves the non-deniable part. When it is mounted in deniable mode, the deniable part is considered when available.

6.2.2 System architecture

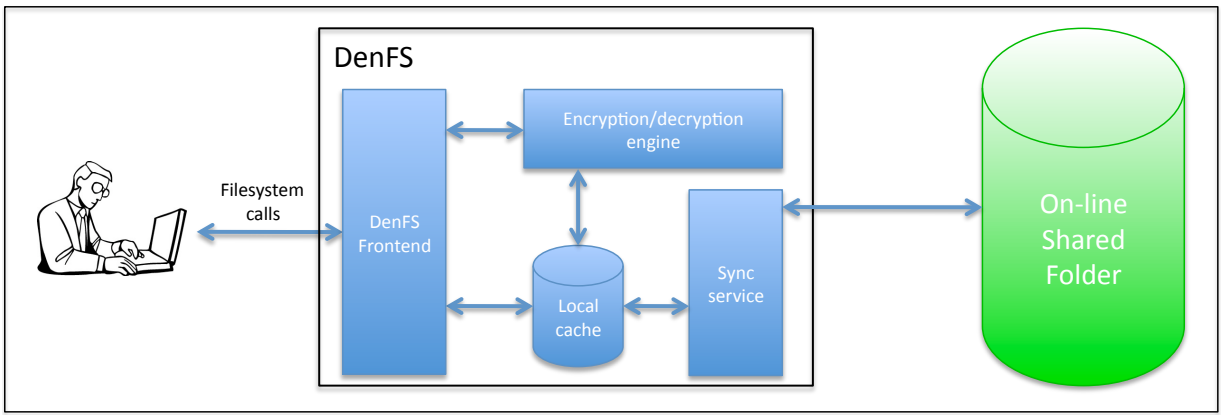

Figure 6.1: Operation of the DenFS filesystem. DenFS automatically encrypts and decrypts files and takes care of the synchronization with a shared folder stored in the cloud. DenFS is composed of a front-end which can directly access the local cache for reading and writing file date, permissions and ownership. The front-end relies on the cryptographic engine to read and write files from and to the local cache. A synchronization service constantly watches the local cache and receives updates from an on-line shared storage and keeps the content of both coherent.

DenFS is divided into several components. The architecture of DenFS is summarized in figure 6.1. DenFS front-end provides a standard interface for filesystem calls, such as open, read and write. We based our implementation on FUSE (Filesystem in Userspace). Fuse is composed of a library and a kernel module which allow non-privileged users to implement a fully functional filesystem in a userspace program [Fil]. FUSE is available for several operating systems, such as Linux, Solaris, FreeBSD, and Mac OS X. It is particularly useful for writing virtual filesystems, which do not actually store data themselves like regular filesystems, but act as a view or translation of an existing filesystem or storage device.

DenFS uses a local folder as a back-end for storing encrypted files. The system calls which deal with file permissions, access and modification time, and file ownership implemented by the front-end are simply forwarded to the back-end (local cache) which takes care of implementing them. All the other system calls are mediated by a cryptographic engine which implements our deniable encryption scheme. The cryptographic engine relies on the OpenSSL library. OpenSSL [Ope] is an actively developed, well know cryptographic library and SSL/TLS toolkit. It provides fast implementations of the ciphers, hash functions, and signature algorithms used in SSL.

When a network connection is available, the local cache is constantly synchronized with an on-line storage service and therefore with all users who share it. The on-line storage and synchronization is provided by Dropbox [Dro], which also manages conflicts and lets users recover deleted files.

6.2.3 Low-level Description

Our filesystem provides a transparent translation layer between the deniably encrypted local cache and the deniable folder. When a new file is added to the filesystem, an encrypted file with the same name is automatically created in the backend folder. For efficiency reasons, all files in the backend are encrypted using a hybrid encryption scheme. The asymmetric cipher used is RSA-OAEP and the symmetric algorithm is AES with a 128-bit key in counter mode. HMAC is used to preserve the integrity of files. The structure of the files in the backend folder is:

header length	*header*	*size*	*non-deniable data*	*deniable data*

The *header* field contains the symmetric keys for the *deniable* and *non-deniable* messages, the initialization vectors and the MAC tag. The fields *header length* and *size* are 32-bit little-endian unsigned integers and indicate the length of the header and the file size,

respectively. Both *header length* and *size* are expressed in bytes, therefore the maximum file size is 4GB. This, however, requires the underlying filesystem in the backend folder to allow the creation of a file slightly larger than 8GB to accommodate the deniable and non-deniable data and the other fields. Dropbox does not impose any limit on the file size, although the free account only allows users to upload up to 2GB of data and therefore up to about 1GB of deniably encrypted data.

The header contains the session keys used to encrypt the non-deniable and the deniable messages, the two initialization vectors, and the two MAC tags. In our prototype, the size of the key for the symmetric cipher is fixed to 128 bits. The default key size for the asymmetric cipher is 1024 bits and can vary from 1024 to 4096 bits. Users keep a local file which maps a directory on the deniable filesystem to a user's key. Adding a new file to a directory also means encrypting it under a specific key, which is chosen from the local map. If no key is specified for a folder, a new file stored in that folder is encrypted under the user's public key.

For every new encrypted file, the session key is saved in a buffer. This allows a user who writes a new file to be able to read it back even if it did not encrypt it under its public key. This feature allows the filesystem to mimic more closely the behavior of a standard filesystem in which, after writing a file, users can re-read it. By disabling this feature, a new file disappears after being saved if it was not encrypted under the creator's public key. We believe that this may be confusing for some users, therefore we suggest to leave it enabled. The key buffer is erased when the file system is unmounted. It is possible to set-up a timeout after which a session key expires and is removed from the buffer. The buffer is a 32-bit unsigned integer which represents the validity period in seconds. Setting the timeout to 0, disables this feature completely.

When the filesystem is used in non-deniable mode and a user creates a new file, our prototype writes a file with the same name in the backend. Then it picks a random session key, saves it in the key buffer and encrypts it. The ciphertext is stored in the file header. A write operation on the deniable folder translates to an encryption and a write operation in the associated file on the backend. The *size* field is updated accordingly. After that, the deniable part of the backend file is filled with random data and a random value in the ciphertext space is inserted in the header as a deniable encryption key.

When a user creates a new file on the filesystem mounted in deniable mode, a temporary file – *filename.den* – is created in a local temporary folder for efficiency reasons. Note that when writing a new file, the size of the file is not necessarily know so it is impossible to predict the offset for the encryption of the deniable data. The temporary file contains the encryption of the deniable part of the document. The session keys are chosen and stored similarly to the non-deniable case. Every write operation on the front-end translates in a write operation in the temporary deniable file. When the document is closed, a file from the deniable pool is encrypted and stored in the non-deniable part of the document. The

file is also removed from the non-deniable pool. The encryption of the deniable content is stored after the encryption of the file from the non-deniable pool.

For every open file, a data structure is created. The data structure contains, among other fields, the following information:

- `nden_ivec` and `den_ivec` contain the nounce and the counter for the symmetric encryption of the non-deniable and deniable data respectively
- `nden_ecount` and `den_ecount` contain the last output of the AES cipher used in CTR mode for the non-deniable and deniable encryption respectively
- `ndenkey` and `denkey` are the symmetric encryption keys
- `size` and `headsize` represent the size of the data and the size of the header respectively
- `den_fd` contains the file descriptor for the temporary deniable file, if any

Closing a file implies the destruction of such data structure.

When a stat request is made to the filesystem, the attributes of the file in the backend are reported, except from the file size, which is read from the appropriate field. The available space on the filesystem is reported as the available space in the filesystem which contains the backend directory. Hard links are not supported in the current implementation.

6.2.4 Performance Evaluation

We performed a combination of synthetic and real-world benchmarks in order to verify the performance impact of DenFS compared to a regular non-encrypted filesystem (Ext3). We used Bonnie++ [Bon], a filesystem benchmark tool, and the OpenOffice productivity suite to evaluate the performances of our prototype.

Bonnie++ performs several tests. We were interested in measuring the performances for sequential and random input/output. For large files performance, we measured sequential output performances writing a single character at a time, a whole block and rewriting one block at a time on a 3GB file. We also performed a sequential reading test, where we read one character at a time and then one block at a time. Finally we performed random seeks within a large files and measured how many seeks per second we were able to perform. Then we measured the overhead for small files by creating, reading and deleting 512,000 files in both sequential and random order.

The test machine is based on an Intel Core 2 Duo T9300 processor running at 2.6 GHz. The available RAM is 4GB and the hard disk is a 320GB, 5400 rpm Toshiba MK3252GSX.

Table 6.1: Sequential Output on a 3GB Dataset

	Per byte		**Block**		**Rewrite**	
	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU
Ext3	31657	39%	34455	5%	14432	2%
DenFS	22579	37%	34056	5%	14442	3%

Table 6.2: Sequential and Random Input on a 3GB Dataset

	Seq. per byte		**Seq. per block**		**Random seeks**	
	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU
Ext3	35108	40%	53882	3%	4137	6%
DenFS	35447	52%	51758	2%	143.2	0%

The operating system is Ubuntu Linux, the kernel version is 2.6.31. The key length for the asymmetric cipher was set to 1024 bits in all tests. The filesystem was mounted in deniable mode. The results for sequential and random access on a large file are summarized in tables 6.1 and 6.2. The results clearly show that when dealing with large files, all tests except random seeks are disk limited. In most cases, there is a small loss of performances introduced by DenFS which we believe is not going to be noticeable in everyday use. The most pronounced performance penalty is in the random read performances which decreases from about 4MB/s to 143KB/s.

The overhead introduced by DenFS is more evident when dealing with small files. Regular filesystems rely on large buffers to speed up these kinds of workload. Unfortunately, the performance of our prototype cannot be improved significantly when creating or reading small files, since one public key encryption or decryption is needed at every access. This explains the drop in measured performances shown in table 6.3 and table 6.4. Note that in our test DenFS was capable of deleting files ten to forty times faster than Ext3. The reason for our filesystem outperforming the underlying Ext3 is that files are not deleted immediately but put in a queue and processed in background. Deleted files are not shown on the filesystem, so the filesystem behaves as expected from an application point of view.

We investigated further to determine the amount of the overhead introduced by the cryptographic algorithms with small files. More specifically, we were interested to see how the file size impacted the read and write speed. Therefore, we created a large set of small files and we measured the read and write speed when copying the files from and to the deniable filesystem. The size of the files composing our testbed varied from 1KB to 128KB. The results are summarized in figure 6.3, which shows the average speed in MB/s for a given file

Table 6.3: Small Files Sequential Performances – 512K Files

	Create		**Read**		**Delete**	
	files/s	CPU	files/s	CPU	files/s	CPU
Ext3	45643	65%	551512	99%	1227	1%
DenFS	5459	76%	22212	85%	17059	15%

Table 6.4: Small Files Random Performances – 512K Files

	Create		**Read**		**Delete**	
	files/s	CPU	files/s	CPU	files/s	CPU
Ext3	19388	27%	502471	99%	561	0%
DenFS	4902	67%	25766	84%	22280	22%

size. With small files, the overhead introduced by our filesystem is non-negligible. When the file size increases, both write and read performance improve noticeably as expected.

We also performed a real-world test with OpenOffice 3.0.1 build 9379. In order to simulate access to complex documents, we created a 306 pages OpenOffice Writer document and a spreadsheet of an approximate size of 940KB. The spreadsheet contained several thousand cells and ten graphs, while the word processor document, which was all written using the same font, also contained some bitmap images. We tried to open, modify, and save them on both Ext3 and DenFS. A copy of both OpenOffice Writer and Calc was open in the background to get rid of the application loading time in the measurements. The reading and writing times were evaluated using a stopwatch. To measure the open time, the watch was started together with the pressure of the "Open" button in OpenOffice open dialog window, and was stopped when the document was fully displayed. For the save time, the watch was started together with the pressure of the "Save" button on the toolbar, and was stopped when the progress bar at the bottom indicating the saving process disappeared. The values shown on figure 6.2 represent the average of ten measurements. The performances of the two filesystems are virtually indistinguishable for a user. The overhead introduced by DenFS is negligible on the test system with OpenOffice.

We believe that the performances provided by DenFS are sufficient for the tasks it is designed for. Synthetic benchmarks confirmed that the weak point of DenFS is accessing several small files, where the overhead introduced by the public key operations is more noticeable. Our real world tests confirmed that this behavior is not noticeable when users create and edit documents and exchange them in encrypted form with their collaborators. We do not expect users to create or open small files on a deniable filesystem at a rate

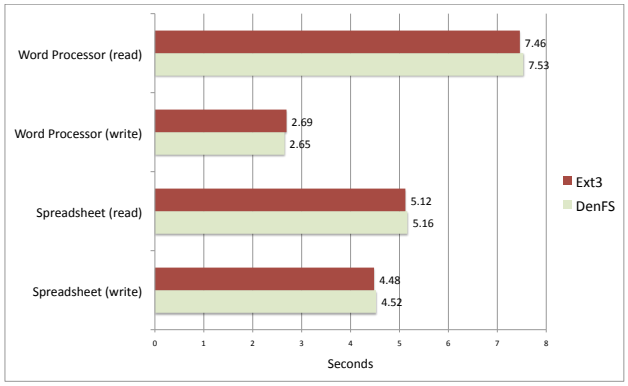

Figure 6.2: Performance comparison between Ext3 and DenFS when opening and saving a spreadsheet and a word processor document with OpenOffice 3.0.1. The reported values are measured in seconds.

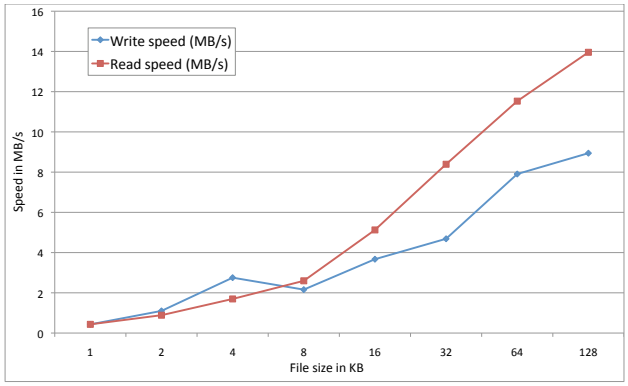

Figure 6.3: Read and write speed for small files. The size of the files used in our test varied from 1KB to 128KB.

that would possibly exceed the values measured in our synthetic benchmark. We believe that this chapter brings deniable encryption to practical use by introducing the concept of deniable cloud storage.

Chapter 7

Integrity in Network Coding Anonymous P2P Networks

Peer-to-peer (P2P) networks have been designed to support data exchange among interconnected peer entities. In a P2P network any peer acts both as a data requestor and a data provider. Recent studies (e.g. [CW07] and [BFW09]) have shown that the use of network coding as a routing technique may provide advantages in terms of throughput, delay, and fault tolerance in decentralized networks. However, pollution attacks (a denial of service attack where byzantine nodes inject corrupted packets in the network [BFW09]) can have more severe consequences when network coding is in use instead of standard routing. In fact, a corrupted packet injected by an attacker gets linearly combined with other packets and can thus pollute more packets along various routing paths. Thus, it is paramount to define network coding based networks that are robust against pollution attacks. In particular, it is important to augment network coding with efficient solutions for checking the integrity of data blocks.

This issue has been investigated in decentralized networks [KMB08] where a sort of block authentication can be assumed. Basically, blocks are signed with the private key of the producer and this allows an easy form of integrity check. However this approach does not easily apply to networks requiring anonymity.

In this chapter, we tackle the problem of integrity in an anonymous P2P network called PANGEA (P2P Architecture based on network coding Granting Extended Anonymity). PANGEA bases its anonymity on network coding. The anonymity has two consequences from an integrity perspective. First, anonymity of data producers prevents the use of digital signatures, since they would associate blocks with producers. Second, anonymity of data consumers makes it very difficult to accomplish distributed and collaborative integrity checks [CW07]. In general, PANGEA defines original and not previously analyzed

conditions for a scenario based on network coding.

The main contributions of this chapter are: (1) definition of an identification scheme for data blocks and (2) an efficient strategy for checking the integrity of packets, both for a decentralized anonymous storage network based on network coding. We support our claims with performance evaluation of a small prototype which implements the integrity strategies described in this chapter. Our idea is to abandon signature-based techniques altogether and use a simpler hash-based approach. We then extend this scheme to deal with blocks in batches, rather than one at a time, based on the reasonable assumption that faulty blocks are expected to be infrequent. This improves performance, but also requires techniques for discovering individual corrupted blocks in faulty batches.

Part of this chapter appeared in "P. Gasti, A. Merlo, G. Ciaccio and G. Chiola, On the Integrity of Network Coding-based Anonymous P2P File Sharing Networks", The 9th IEEE International Symposium on Network Computing and Applications (IEEE NCA10), Cambridge, MA, USA [GMCC10]

7.1 An Integrity Mechanism for PANGEA

Existing anonymous P2P network provide a satisfactory level of anonymity at the cost of high latency and poor bandwidth [SHGA06]. We designed a P2P network in the effort of overcoming these limitations and maintaining an adequate level of anonymity. To achieve this result, we rely on network coding techniques.

PANGEA is an anonymous P2P file sharing network. It is based on network coding for the packet exchange, and on an anonymity-preserving protocol for adding new documents and for finding existing resources. PANGEA allows peers to choose a level of anonymity when issuing file requests or introducing new contents. The details and analysis of the protocol are out of scope for this chapter; we rather limit ourselves to explaining the PANGEA structure in order to identify the issues arising from the integrity check requirement.

The first problem that emerged during the design of PANGEA was the integrity problem. None of the available solutions for network coding seems to provide a viable solution in our setup. More specifically, the anonymity requirement does not allow a straightforward use of homomorphic signatures: signature-based approaches would require each block to be signed by the content provider. Clearly, in our framework the content provider does not want to reveal its identity, and therefore only its public key should be known. Each user can generate a public/signing keypair and use it for signing any block of its choice. It is therefore possible for a peer to maliciously alter a block and then sign it with its own signing key. The corruption will be apparent only once the file is completely reconstructed, but by then several other possibly independent blocks would already be polluted [GR06]. Thanks

to the anonymity property of the system, keys are not associated with users. Therefore even if the corrupted block is found by other peers, the malicious peer can deny the possession of the signing key and claim that the block was corrupted somewhere else. It is possible that this problem can be solved associating a reputation with each public key. Only keys with a reputation higher than a chosen threshold would be accepted as valid signing keys. This solution has, however several drawbacks. Signature produced by a new user who is sharing one or more files with other peers cannot be verified reliably until a certain threshold for its reputation has been met. Another problem of this approach is that reputation values for a specific public key should be made available to all peers. Failing to do that would imply that a key which has a high reputation for some peers, has a low reputation for others. This would unnecessarily delay block verification when that key is involved. This global knowledge would require either the use of a P2P information exchange protocol (see e.g. [ADH05]), which is in turn susceptible to integrity attacks; alternatively, a centralized database with the association of a key and its reputation should be maintained.

We decided to focus on a different approach: we abandoned signature-based techniques altogether and used a simple hash-based approach instead. Each new file is divided into blocks of a predefined size. A hash is calculated using a collision-resistant homomorphic hash function on each block individually. Thanks to the collision-resistance of the hash function, different blocks have a different hash with overwhelming probability. We exploit this property to identify each block with its hash value. Blocks with the same content belonging to different files will have the same hash and therefore will appear as the same block to the system. Depending on the concrete instantiation of the network, this helps to increase either efficiency, because the same block does not have to be stored more than once even if it belongs to different files, or redundancy, since different blocks belonging to different files are implicitly replicated. We indicate the hash value for block b with ID_b.

PANGEA allows each peer to share new files. Before sharing a new file, a peer divides the file into predefined-size blocks. Then it calculates the ID for each block using a homomorphic hash function and sends all the resulting block IDs to a special peer, called super-peer, together with the associated file name and other metadata, such as file type, tags etc. A super-peer is in charge of maintaining the association between a list of block identifiers and the file they represent. Peers willing to download a specific file retrieve the list of the IDs which compose such file from the super-peer. For each block they retrieve, the integrity is verified on the fly by calculating the hash value of the block and comparing it to the expected hash value. Corrupted blocks are identified immediately and do not pollute any other block. No producer-specific information is needed to verify a block.

Clearly peers must trust the super-peer to provide non corrupted packet IDs. We believe that this problem is much easier to solve compared to the trust problem associated with signature-based integrity techniques. Peers only need to trust one entity – the super-peer – instead of several independent entities – the signers. There is, however, another

difference between our approach and the signature-based approach: with our approach a content provider can either provide a corrupted file or associate incorrect metadata with a shared file (fakes). This problem, however, is common to several P2P file sharing systems [LKXR05] and is out of the scope of this chapter, which focuses primarily on the integrity of the blocks within the overlay network.

Network coding allows each node to linearly encode an arbitrary number of blocks into a single block. When a block is encoded, peers must attach the list of the IDs of the blocks used to produce that encoding. To verify an encoded block, a peer combines the list of IDs to obtain a single packet ID and then verifies the output of the homomorphic hash function on the aggregate block with the combined ID. In the next section we review the requirements for a collision-resistant homomorphic hash function suitable for our approach. Additionally we detail a concrete instantiation and provide some experimental results.

7.2 System Model

The main features of our system model are as follows:

- **Storage:** each peer can store received packets in a temporary buffer until it is ready to verify their integrity.
- **Communications:** peers can always reach one another via the underlying network, which is supposed not to introduce errors in the transmitted data. Network coding is used for encoding and routing individual packets in the network, as well as decoding packets once they have passed the integrity checks.
- **Secure channels:** each peer can run basic cryptographic primitives so that a possible eavesdropper cannot decode ciphertext or inject fake packets or alter in-flight packets in a communication channel between two peers. Therefore the integrity of each packet is completely under the responsibility of the sender.

The envisaged P2P system should provide some degree of anonymity to both the producer and the consumer of information. Due to this requirement, the integrity verification scheme should minimize any leakage of information about the producer of a verified block. Ideally, it should be even made impossible to determine whether two data blocks belonging to different files were produced by the same user. As already pointed out, this rules out signatures in general, and homomorphic signatures in particular (like, for instance, the scheme proposed in [BFW09]). Indeed, although a key in itself may not leak information

about its owners, signatures allows to determine whether two files have been signed by the same user (or, at least, by users who share a private key). Short-lived keys may not present such a drawback, but are at the same time unable to convey reputation information to the receiver, so they become virtually useless. Instead, it is much easier for a super-peer to build its reputation compared to a user's signing keypair, since we expect a super-peer to be a long-lived entity, well known by all peers, what is clearly not the case for most user's public keys.

7.3 Adversarial Model

The envisaged adversary has the following features:

- **Goal:** the goal of the adversary is to corrupt the content of some or all of the packets that flow through the network. Packets must be corrupted is such a way that legitimate peers cannot detect packet corruption until a significant portion of the file has been retrieved.

- **Compromise power:** the adversary can compromise any note at its will and fully control its behavior. When compromised, a node will reveal all its secret information. The adversary can modify only the packets that flow through compromised nodes.

- **Parameter generation:** the adversary cannot interfere with the generation of the the global parameters for the system.

- **Defense awareness:** the adversary is fully aware of any scheme or algorithm that the peers use to defend the integrity of the packets they exchange.

7.4 Homomorphic Hash for Network Coding

A major concern in systems that use Network coding is to avoid Byzantine nodes to introduce modified blocks in the network. This behavior, known as "pollution attack", allows an adversary to corrupt a large amount of blocks in the network, because errors introduced into a single block can propagate and corrupt multiple blocks making their way to the destination. In contrast to traditional routing schemes, network coding requires intermediate nodes to modify data blocks in transit. For this reason, standard hashing or signature schemes are not applicable and it is challenging to provide resilience to tampering by malicious nodes [KMB08].

Homomorphic hash and homomorphic signature algorithms provide a valuable tool for verifying packet integrity when network coding is used. A homomorphic hash algorithm suitable for our approach is defined by the following algorithms:

$\mathsf{Setup}(1^n)$. A probabilistic algorithm which, on input a security parameters 1^n, outputs the public parameter pk. The running time of the encryption, decryption, and the adversary algorithms are all measured as a function of a security parameter, which is expressed in unary notation.

$\mathsf{Hash}(pk, \mathsf{b})$. A deterministic algorithm which on input a public parameter pk and a block b, outputs a hash value ID. Given pk and b it must be computationally infeasible to find a block $\mathsf{b}' \neq \mathsf{b}$ such that $\mathsf{Hash}(pk, \mathsf{b}) = \mathsf{Hash}(pk, \mathsf{b}')$.

$\mathsf{Combine}(pk, \mathsf{b}_1, \ldots, \mathsf{b}_s, \mathsf{ID}_1, \ldots, \mathsf{ID}_{s'}), s \leq s'$. An algorithm which, on input a public parameter pk, a list of blocks $\mathsf{b}_1, \ldots, \mathsf{b}_s$ and a list of block IDs $\mathsf{ID}_1, \ldots, \mathsf{ID}_s$, outputs a new block $\mathsf{b}^{(s)}$ which is the linear combination of $\mathsf{ID}_1, \ldots, \mathsf{ID}_s$, and its hash $\mathsf{ID}^{(s)}$ derived from $\mathsf{ID}_1, \ldots, \mathsf{ID}_s$.

$\mathsf{Verify}(pk, \mathsf{b}, \mathsf{ID})$. An algorithm which, on input a public parameter pk, a block b and a hash ID outputs 1 if $\mathsf{Hash}(pk, \mathsf{b}) = \mathsf{ID}$, 0 otherwise.

While blocks have a fixed size, files are allowed to be of arbitrary size. Files are seen as an ordered sequence of blocks.

We use the global homomorphic hash algoritm proposed by Krohn et al. in [KFM04] for calculating the blocks ID and for integrity checks, since it produces relatively short hash output and allows batch verification for improved performances. The size of the hash values in Krohn et al. global homomorphic hashing scheme is independent of the encoding rate, although in PANGEA an encoded block carries the list of all the identifiers of the blocks it was composed with. Moreover, hash values can be verified by the receiver on the fly. By allowing all the nodes to share the global hash parameters, there is a single way to map a block b to $\mathsf{Hash}(\mathsf{b})$. As such, the hash generation on a set of blocks must be performed only once and the resulting values are well-defined. Since there may be different publishers for the same content, by using global hashing all copies look identical to the system.

7.4.1 The Hash Algorithm

In this section we briefly introduce the hash scheme of Krohn et al. [KFM04]. Informally, two random primes p and q and a vector $\mathbf{g}$ are chosen by a trusted party and shared among

peers. The trusted party can be the developer of the network coding library or one of the super-peers. The trust on this party is, however, limited since a malicious trusted party can generate collisions, which we assume can quickly be noticed. A collision is also the proof that such a malicious party exists and would lead to the re-generation of the public parameters.

Each file is divided into blocks, and each block is divided into several sub-blocks of size $|p|$. The ID of a block has the same size of a sub-block. To combine two blocks, a peer simply computes the sum of the corresponding sub-blocks; let $\mathsf{b}_1 = (b_{1,1}, \ldots, b_{1,\ell})$ and $\mathsf{b}_2 = (b_{2,1}, \ldots, b_{2,\ell})$. The combination of b_1 and b_2 is $\mathsf{b}^{(s)} = (b_{1,1} + b_{2,1}, \ldots, b_{1,\ell} + b_{2,\ell})$. The combination of the identifiers ID_1 of b_1 and ID_2 if b_2 is $\mathsf{ID}^{(s)} = \mathsf{ID}_1 \cdot \mathsf{ID}_2$. An encoded block is not corrupted if $\mathsf{ID}^{(s)}$ is equal to the hash calculated on the combined block $\mathsf{b}^{(s)}$.

More formally, the three algorithms composing the homomorphic hash function are defined as:

$\mathsf{Setup}(1^n, 1^m)$. Picks two random primes p and q such that $|p| = n$, $|q| = m$ and q divides $(p-1)$. Then it picks a vector $\mathbf{g} = (g_1, \ldots, g_\ell)$ composed of $\ell = \left\lceil \frac{|\mathsf{b}_j|}{m-1} \right\rceil$ random elements from $\mathbb{Z}_p$, all of order q. Outputs $pk = (p, q, \mathbf{g})$

$\mathsf{Hash}(pk, \mathsf{b})$. Divides each block b into ℓ sub-blocks $\mathsf{b}_1, \ldots, \mathsf{b}_\ell$ of size $|q| - 1$, then outputs a hash value ID calculated as:

$$\mathsf{Hash}(pk, \mathsf{b}) = \prod_{i=1}^{\ell} g_i^{\mathsf{b}_i} \pmod{p}$$

$\mathsf{Combine}(pk, \mathsf{b}_1, \ldots, \mathsf{b}_s, \mathsf{ID}_1, \ldots, \mathsf{ID}_s)$. divides each block b_i into ℓ sub-blocks $b_{i,1} \ldots, b_{i,\ell}$ calculates the combined block $\mathsf{b}^{(s)}$ as:

For j **in** $1 \ldots \ell$ **do**

$$\mathsf{b}_j^{(s)} = \textstyle\sum_{j=1}^{s} \mathsf{b}_{i,j} \pmod{q}$$

done

and its hash $\mathsf{ID}^{(s)}$ as

$$\mathsf{ID}^{(s)} = \prod_{i=1}^{\ell} \mathsf{ID}_i \pmod{p}$$

$\mathsf{Verify}(pk, \mathsf{b}, \mathsf{ID})$. outputs 1 if $\mathsf{Hash}(pk, \mathsf{b}) = \mathsf{ID}$, 0 otherwise.

Reasonable values for n and m are 1024 and 257 bits respectively. Our tests were performed with those values. Note that the knowledge of a tuple (i, j, x_i, x_j) such that $g_i^{x_i} = g_j^{x_j}$ for some $i \neq j$ suffices to efficiently compute arbitrary hash collisions. Therefore, all nodes must trust the third party in charge of generating the global parameters, which must be chosen accordingly to [KFM04].

7.4.2 Probabilistic Batch Verification

The homomorphic hash algorithm presented in [KFM04] allows a node to verify many blocks probabilistically and in batches in order to improve efficiency. The idea of improving verification performances by batching is not new and appeared for the first time in [BGR98]. Batch verification allows a peer to verify several block at once in order to improve aggregate verification speed. If the batch verifies successfully, then all blocks are correct with high probability, otherwise one or more blocks are corrupted. In order to perform a batch verification on a batch of size $\mathcal{B}$, a peer combines $\mathcal{B}$ blocks and their IDs using the Combine algorithm. Then it computes the homomorphic hash on the aggregate block and verifies whether it matches with the expected value obtained with the Combine algorithm.

Unfortunately, as pointed out by Gkantsidis and Rodriguez [GR06] and Yu et al. [YWRG08], an adversary can poison two or more blocks in a way that, when verified together in a batch, the resulting combined block is not affected by such modification. As an example, an adversary can corrupt blocks b_1 and b_2 by replacing them with $\mathsf{b}_1 + \varepsilon$ and $\mathsf{b}_2 - \varepsilon$. If the two blocks are verified separately, the verification would clearly fail, but if the receiver applies batch verification it will not detect the corruption. Both the cited papers provide a solution to this problem by multiplying each block by a random coefficient. With this modification the adversary can only succeed by guessing the random coefficient correctly, which happens only with negligible probability.

We implemented the homomorphic hash algorithm to measure its performance on our test platform, performing both stand-alone verification and batch verification and compared the results. We based our implementation on the OpenSSL library (version 0.9.8g) under Linux. Our test were performed on a Intel Core 2 Duo T9500 at 2.6GHz. Table 7.1 shows the performances of our algorithm on the test machine. We performed batch verification for a batch of 1000 blocks. The size of each block is 16KB, and the size of batch window is therefore slightly less than 16MB.

As our test confirmed, the cost of the composition of several blocks and their hash values is very low compared to the cost of calculating a single hash. Therefore the cost of verifying a batch is marginally higher than the cost of verifying a single block, and this holds even

Table 7.1: Performance evaluation of the hash algorithm. The batch size is 1000 blocks and the block size is 16KB.

Public parameter generation	1436 ms
Average time for calculating a hash	289 ms
Average speed for calculating a hash values	56 KB/s ($\sim$ 450Kbps)
Average time for composing 1000 hash values	2.9 ms
Average time for composing 1000 blocks	141 ms
Average time for verifying a batch	433 ms
Average speed for verifying a batch	37 MB/s ($\sim$ 300Mbps)

for rather large batches.

Batching clearly introduces a delay, since a newly received block cannot be used until a specific amount of data has been received and the whole batch verification is completed. How this delay affects network coding based systems was first studied by Gkantsidis and Rodriguez in [GR06]. They showed that a large batch window severely affects the performances of a content distribution network based on network coding, since explicit content requests pay a high delayed, proportional with the number of intermediate peers. This problem, of course, is mitigated when data traffic is mainly unidirectional. For instance, with content streaming or long downloads, the batching latency can be hidden after an initial "buffering" time; contents are then distributed in a pipeline fashion, and the batching does not increase the download time significantly.

7.4.3 Security Analysis

Given the adversary's compromise power and knowledge of the scheme, the scheme and usage mode in this paper effectively prevent an adversary from reaching its goal. By compromising a node, the adversary does not obtain any new secret information, since all nodes share the same information regarding packet verification. Moreover, even if the adversary subverts all but one nodes, the remaining legitimate peer will still be able to tell wether a packet has been polluted by any adversary-controlled node.

An adversary who has control over the generation of the global parameters for the homomorphic hash function can easily calculate collisions and therefore allow corrupt packets to propagate through the network. In this case a collisions implies that two or more blocks share the same block identifier. However, the **Parameter generation** constraint in our adversary model prevents this from happening. In practice this means that these global parameters must be chosen by a trusted third party. We point out that whenever the

party which generates the global parameters misbehaves, legitimate peers can quickly notice thanks to the availability of hash collisions in the network. The peers can react by discarding the global parameters and consider the third party as hostile. In order to be able to communicate again, they will have to find another trusted third party.

7.5 Batch Verification Strategies

As shown before, pollution attacks are the main reason for checking every packed encoded using network coding before accepting it. Verifying every block individually is computationally expensive, therefore we proposed to combine together several blocks and verify them at once to improve performances. Batching is an optimistic approach in which verification performances are greatly improved when the whole batch integrity is satisfied. Unfortunately if the verification of a batch fails, peers are left with no information about which block (or blocks) made the verification fail. Hence less efficient strategies must be applied. A peer can discard the whole batch without investigating further, but this approach does not make an efficient use of the resources since the bandwidth of the peer is limited and most of the blocks may not be corrupted. Therefore, when a batch fails a peer should try to keep as many non-corrupted blocks as possible to maximize bandwidth efficiency. The trivial way to find the non-corrupted blocks in a corrupted batch is to verify each block separately. This is very inefficient and has a cost which grows linearly with the number of blocks composing a batch. A more efficient strategy when the number of corrupted blocks is low is to perform a bisection search: the sub-batch is divided into two parts. Each part is verified as a new batch and is accepted if it is non-corrupted. The process is repeated on the part that does not verify correctly, until all corrupted blocks are found. When the number of corrupted blocks is small compared to the size of the batch this approach is quite efficient: it has a logarithmic complexity in the number of blocks composing a batch.

We studied how to provide a more efficient strategy for batch verification when a small number of blocks in a batch is corrupted. In PANGEA, peers can assume that the underlying communication layers provide reliable communication. Each peer is assumed to verify each block that is forwarded to other peers; therefore if a block is corrupted, it is so because of a malicious action. Moreover, the neighbor which forwarded it is the culprit of such corruption.

We assume that, since byzantine nodes can be detected as soon as they forward a corrupted packet, there will be a small number of them. If a batch does not verify correctly, we optimistically assume that only a subset of the neighbors are byzantine. To improve efficiency, when a batch fails we divide it in sub-batches; each sub-batch is composed of all the blocks coming from one neighbor. Sub-batches which verify correctly can be accepted

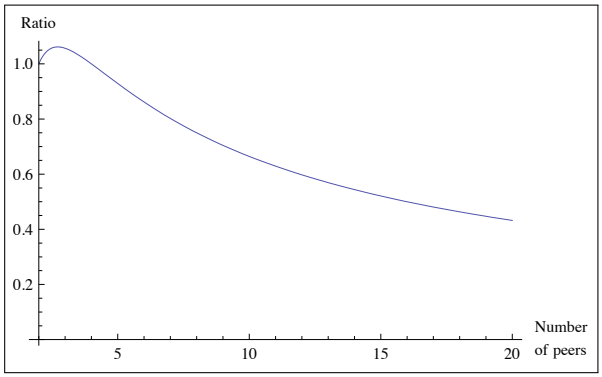

Figure 7.1: Bisection vs. naive partitioning. The graph shows the ratio of the expected number of verification with bisection and naive partitioning.

immediately; sub-batches which fail the verification can either be discarded completely or be verified with the bisection approach. However, discarding all blocks from a specific source may provide a good tradeoff between bandwidth efficiency and computational overhead. Once a node is discovered to be malicious, it can be removed from the neighbor list. This form of naive partitioning has a complexity which grows linearly with the number of neighbors.

A more efficient approach is to use the bisection technique at a sub-batch level instead of block level. Figure 7.1 shows the ratio between the expected number of verifications performed in the case of bisection and in the case of naive partitioning, varying the number of neighbors. When a node has a small number of neighbors, there is no much difference between the two approaches. The difference becomes more evident and justifies the adoption of this approach when the number of neighbors grow: when a node has eight neighbors, the expected number of verifications with neighbor-level bisection is 3/4 of the expected number of verification with naive partitioning. When the neighbors are sixteen or more, the bisection approach requires less than half verification compared to naive partitioning on average.

We designed another batch verification approach which relies on statistics on the previous behavior of neighbors, called ranked partitioning. With this approach a node keeps track of how many blocks it received from each neighbor. In case a batch verification fails, the node divides the batches in sub-batches according to the source of the blocks, and orders them by the ranking of the source. Then it verifies the sub-batches sequentially, starting from the one with the lowest ranking. We argue that a neighbor with a high ranking is less likely to forward corrupted blocks than a neighbor with a low ranking. A denial of service attack against a peer executed by exploiting the batch verification algorithm has a higher effectiveness if it is performed by the neighbor with the highest

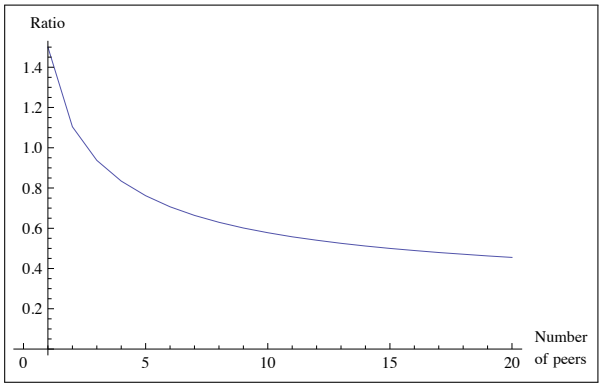

Figure 7.2: Ranked partitioning vs. bisection. The graph shows the ratio of the expected number of verification with bisection and ranked partitioning.

ranking. However, in order to be among the neighbors with highest ranking, an adversary must first provide a large contribution in terms of non-corrupted blocks, therefore making the attack less effective (the user has to pay a higher computational cost to find the byzantine neighbor, but received a large number of non-corrupted blocks from that very neighbor) and particularly expensive to implement. Figure 7.2 shows the performances of this approach compared to the bisection methods. The graph represents the ratio of the expected values of the ranking partitioning and the bisection approach varying the number of neighbors. When a node has a very small number of neighbors, the ranked partitioning approach is less efficient than the bisection approach. Increasing the number of neighbors, the advantage of this batching method over bisection becomes more evident. A peer with sixteen nodes performs, on average, about half verification operations using the ranked partitioning approach compared to the bisection approach.

Chapter 8

Conclusions

In this thesis we have discussed, analyzed and improved several different tools that can be used as powerful building blocks for systems which preserve users privacy.

We proposed UAnonIBE: the first IBE providing universal anonymity (thus key-privacy) and secure under the standard quadratic residuosity assumption. The efficiency and ciphertext expansion of this scheme are comparable to those of Cocks IBE. We showed that Cocks IBE and our anonymous variant are suitable in practice whenever hybrid encryption (KEM-DEM paradigm) is employed. We believe our scheme is valid alternatives to decidedly more expensive schemes introduced in [BGH07] (which, in addition, are anonymous but not universally anonymous). Incidentally, our results can be used to simplify existing PEKS constructions [CS07] and base their security on the standard quadratic residuosity assumption (which was left as an open problem in the area).

In normal circumstances encryption provides data privacy unless the adversary is powerful enough to force a user into opening ciphertexts. Furthermore, whenever the adversary can force the victim to open all communication and all data on the victim's machine, there is little hope for secrecy. Fortunately, here is where deniability comes in play. In this dissertation we have provided an efficient construction which is based on standard tools that brings deniable encryption to practical use. We developed the first efficient sender-and-receiver deniable *public-key* encryption scheme based on standard tools. Our scheme is efficient and provably secure, and introduces a minor overhead, in terms of space and time, for most practical purposes.

The focus of this thesis is then shifted to more practical matters. We introduced the notion of StemCerts, which provides users with flexible and privacy-aware digital certificates. We proposed two variants; the first one is not compatible with the X.509 standard, although minimal adaption of the verification and issuing software is needed; the second variant is compliant with the X.509 v3 specifications and it offers a hight expressive power.

We believe that these features can help common users to overcome what is seen as the major drawbacks in the current X.509 standard, such as the lack of flexibility and privacy. We implemented StemCerts as a small patch to the OpenSSL library, and tested it in a real-world scenario. The results of our experimental implementation demonstrate that the scheme is practical and efficient. Moreover, we devised what we claim is a realistic transition path from current legacy software to our proposed scheme that would allow the two versions of certificates to coexist. We feel that this latter result is the main original contribution of our work.

Moreover, StemCerts proved to be compatible with current networking applications, such as popular web servers and browsers which use X.509 v3 digital certificates. Hence, we claim it would not be difficult to integrate this kind of certificate, even in legacy systems. It is up to the Certification Authority and the user's system administrator to decide which certificates can be used with old applications and which need to be verified by software supporting StemCerts.

After that, we have argued that it is possible to realize flexible inter-domain authorizations within the X.509 specifications. Leveraging on the experience obtained when designing StemCerts, we have designed our new X.509 extension to help collaborators maintain their functional autonomy. Our extension allows users to participate in any collaboration initiated by its domain, thus reducing the number of certificates a user need to maintain, obviously reducing the size of the CRL. The feature of rights amplification and degradation was not possible under any other X.509-compatible approach. The ability to quickly initiate/break collaborations while maintaining domain's functional autonomy is specially very useful for ephemeral collaborations. The performance analysis of our implementation showed that the additional cost introduced by our proposal is usually negligible compared to the benefits *InterAC* offers. In RBAC framework, a role is a set of permissions. Therefore, the treatment we provide to permissions in our approach can be easily extended to roles when the collaborating domains have RBAC as their underlying access control framework. The static and dynamic permission sets can be further supplemented with an additional set whose members may carry semantics for context-aware, exception-tolerating authorization.

In this thesis we have discussed how to bring deniable encryption to practical use by introducing the concept of deniable cloud storage. Deniable cloud storage allows users to collaborate and store data in such a way that if an adversary coerces a user into opening all encrypted data, the information still stays secret. We have developed a prototype of a deniable file system, DenFS, which realizes this functionality. Through several tests, detailed in this work, we show how DenFS can provide high performances, low latencies and reasonable space overhead for most applications. We showed that user will not be able to notice any loss of performance when editing their documents stored on a DenFS volume in a common office scenario. We believe that this strongly supports our claim that

deniable cloud storage is a practical collaboration tool and that DenFS provides a viable implementation for that.

The last topic we discussed in this thesis is the integrity problem in network coding based anonymous P2P networks. A strong motivation for the adoption of network coding for P2P file sharing is its exploitation to obtain producer and/or consumer anonymity without significantly affecting the network performance. However, the adoption of network coding in P2P anonymous networks requires proper counter-measures to prevent (or at least mitigate) Denial of Service (DoS) attacks based on pollution.

To this regard, this dissertation provides two novel contributions: (1) a mechanism for identifying univocally any block on the anonymous P2P network and (2) an efficient integrity check based on Batch Verification. In particular, for the unique block identification we propose an algorithm for the computation of a unique hash ID based on the content of the block; for the batch integrity check we propose a verification working on grouped blocks. If the batch integrity check is satisfied then the calculation is extremely quick. In case the verification check fails, the issue of recognizing the sender of the corrupted block (among the neighbors of the peer) arises. To this regard, we define two strategies (bisection and ranked neighbor partitioning) for splitting the grouped blocks in sub-groups and performing parallel checking in order to recognize the malicious sender. Performance evaluation proves the efficiency of our solutions, especially in the case of relatively high number of peers connected. In particular, the ranked neighbor partitioning batch verification schema appears to be robust against pollution attacks, requiring very large efforts for the attacker to be able to significantly affect the efficiency of the integrity check and of the Network Encoding.

Bibliography

[ABC+05] M. Abdalla, M. Bellare, D. CatA.o, E. Kiltz, T. Kohno, T. Lange, J. MaloneLee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In *Advances in Cryptology, CRYPTO '05, volume 3621 of Lecture Notes in Computer Science*, pages 205–222. Springer-Verlag, 2005.

[ACdM05] G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable RFID Tags via Insubvertible Encryption. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 92–101, New York, NY, USA, 2005. ACM.

[ACdMT05] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings*, pages 159–177, 2005.

[ACL99] R. J. Anderson, B. Crispo, and Jong-Hyeon Lee. *The Global Internet Trust Register*. MIT Press, Cambridge, MA, USA, 1999.

[ADH05] A. Allavena, A. Demers, and J. E. Hopcroft. Correctness of a gossip based membership protocol. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 292–301, New York, NY, USA, 2005. ACM.

[AdM04] G. Ateniese and B. de Medeiros. On the key exposure problem in chameleon hashes. In *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, volume 3352 of *Lecture Notes in Computer Science*, pages 165–179. Springer, 2004.

[ADr] ADrive. Revolutionizing Online Storage & Backup. `http://www.adrive.com/`.

[AFG+09] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katzand, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[AG09] Giuseppe Ateniese and Paolo Gasti. Universally anonymous ibe based on the quadratic residuosity assumption. In *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, pages 32–47, 2009.

[Ama] Amazon Web Services: Amazon Simple Storage Service (Amazon S3). `http://aws.amazon.com/s3/`.

[And08] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2 edition, April 2008.

[AR97] A Abdul-Rahman. The PGP Trust Model. Available at `http://www.cs.ucl.ac.uk/staff/F.AbdulRahman/docs/` EDI-Forum, 1997.

[ASN97] ASN.1: Abstract Syntax Notation One, ITU - International Telecommunication Union, 1997.

[BBDP01] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Advances in Cryptology, ASIACRYPT '01, volume 2248 of Lecture Notes in Computer Science*, pages 566–582, London, UK, 2001. Springer-Verlag.

[BBS03] M. Bellare, A. Boldyreva, and J. Staddon. Multi-Recipient encryption schemes: Security notions and randomness re-use. In *Desmedt Y, ed. Proceedings of the Public Key Cryptography PKC 2003*, pages 85–99. Springer-Verlag, 2003.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *Advances in Cryptology, CRYPTO '04, volume 3152 of Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.

[BBW06] A. Barth, D. Boneh, and B. Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography and Data Security*, pages 52–64, 2006.

[BCOP04] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In *Advances in Cryptology, EUROCRYPT*

'04, volume 3027 of Lecture Notes in Computer Science, pages 506–522, Interlaken, Switzerland, 2004.

[BCP03] E. Bresson, D. Catalano, and D. Pointcheval. A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications. In *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, pages 37–54, 2003.

[BDR98] M. Bellare, D.Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology - CRYPTO98*, pages 26–45. Springer-Verlag, 1998.

[BF03a] D. Boneh and M. Franklin. Identity Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[BF03b] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal on Computing, volume 32-3*, pages 586–615, Philadelphia, PA, USA, 2003.

[BFIK99] M. Blaze, J. Feigenbaum, J. Ioannidis, and Angelos Keromytis. The KeyNote Trust-Management System Version 2. RFC 2704, IETF, 1999.

[BFMLS08] K. Bentahar, P. Farshim, J. Malone-Lee, and N. Smart. G.ric Constructions of Identity-Based and Certificateless KEMs. *Journal of Cryptology*, 21(2):178–199, April 2008.

[BFS98] M. Blaze, J. Feigenbaum, and Martin Strauss. Compliance Checking in the PolicyMaker Trust Management System. *Financial Cryptography, FC 1998*, LNCS 1465:254–274, 1998.

[BFW09] Dan Boneh, J. Freeman, D.and Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In *Irvine: Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography*, pages 68–87, Berlin, Heidelberg, 2009. Springer-Verlag.

[BG03] K. Bennett and C. Grothoff. GAP: Practical Anonymous Networking. In *Proc. of Workshop on Privacy Enhancing Technologies (PET 2003)*, March 2003.

[BGdMM05] L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-Resistant Storage via KeywordSearchable Encryption. In *Cryptology ePrint Archive, Report 2005/417*, 2005. Available at `http://eprint.iacr.org/2005/417`.

[BGH07] D. Boneh, C. Gentry, and M. Hamburg. Space-Efficient Identity Based Encryption Without Pairings. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 647–657, Washington, DC, USA, 2007. IEEE Computer Society.

[BGMW92] E. Brickell, D. Gordon, K. McCurley, and D. Wilson. Fast exponentiation with precomputation: algorithms and lower bounds. eurocrypt, 1992.

[BGR98] M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Advances in Cryptology – Eurocrypt 98*, pages 236–250. Springer-Verlag, 1998.

[Ble96] D. Bleichenbacher. Generating ElGamal Signatures Without Knowing the Secret Key. In *EUROCRYPT*, pages 10–18, 1996.

[Bon] Bonnie++: a free and open source filesystem benchmark. Available at . `http://www.coker.com.au/bonnie++/`.

[BR96] M. Bellare and P. Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *Advances in Cryptology - Eurocrypt 96*, pages 399–416. Springer-Verlag, 1996.

[BS99] M. Bellare and A. Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In *Advances in Cryptology - CRYPTO 99*, pages 519–536. Springer-Verlag, 1999.

[BW06] X. Boyen and B. Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In *Advances in Cryptology, CRYPTO '06, volume 4117 of Lecture Notes in Computer Science*, pages 290–307. Springer-Verlag, 2006.

[CAS04] CAS. *Community Authorization Service*. The Globus Alliance, `http://www.globus.org/grid_software/security/cas.php`, 2004.

[CDD+04] D. Chadwick, Theo Dimitrakos, Kerstin Kleese-Van Dam, Damian Mac Randal, B. Matthews, and A. Otenko. Multilayer privilege management for dynamic collaborative scientific communities. In *Workshop on Grid Security Practice and Experience, Oxford*, pages II 7–14, 2004.

[CDF+07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP Message Format. RFC 4880 (Proposed Standard), November 2007.

[CDNO97] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable Encryption. In *Adnvances in Cryptology – CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104, 1997.

[CG06] G. Chiola and P. Gasti. StemCerts: Customizable X.509 v3 Certificates for Higher Security, Flexibility, and Convenience. In *PRISE*, 2006.

[CG09] G. Chiola and P. Gasti. StemCerts-2: Pairs of X.509 v3 Certificates for Greater Security, Flexibility and Convenience. In *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, pages 1–7, 2009.

[CGPM09] G. Chiola, P. Gasti, V. Patil, and L. V. Mancini. Resource management with x.509 inter-domain authorization certificates (interac). In *Public Key Infrastructure, 6th European PKI Workshop: Theory and Practice, EuroPKI 2009, Pisa, Italy, September 11-12, 2009, Proceedings*, 2009.

[CHK07] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. *Journal of Cryptology*, 20(3):265–294, 2007.

[CHK+08] A. Czeskis, D. J. St. Hilaire, K. Koscher, S. D. Gribble, T. Kohno, and B. Schneier. Defeating Encrypted and Deniable File Systems: TrueCrypt v5.1a and the Case of the Tattling OS and Applications. In *3rd USENIX Workshop on Hot Topics in Security (HotSec '08)*, July 29, 2008.

[CO02] D. W. Chadwick and A. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. In *SACMAT '02: Proc. of ACM Symp. on Access Control Models & Tech.*, pages 135–140, 2002.

[Coc01] C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, London, UK, 2001. Springer-Verlag.

[CS98] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology, CRYPTO '98, volume 1462 of Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.

[CS02] C. Gentry Craig and A. Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pages 548–566, London, UK, 2002. Springer-Verlag.

[CS04] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM Journal on Computing, volume 33-1*, pages 167–226, Philadelphia, PA, USA, 2004.

[CS07] G. Di Crescenzo and V. Saraswat. Public Key Encryption with Searchable Keywords Based on Jacobi Symbols. In *Progress in Cryptology, INDOCRYPT '07, volume 3797 of Lecture Notes in Computer Science*, pages 282–296, Chennai, India, December 9-13, 2007.

[CSWH00] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval S ystem. In *Proc. of Designing Privacy Enhancing Technologies: Workshop on D esign Issues in Anonymity and Unobservability (PET)*, pages 46–66, July 2000.

[CTWS02] E. Cohen, R. K. Thomas, W. Winsborough, and D. Shands. Models for coalition-based access control (CBAC). In *SACMAT'02: Proc. of ACM Symp. on Access Control Models & Tech.*, pages 97–106, 2002.

[CW07] P. A. Chou and Y. Wu. Network coding for the internet and wireless networks. Technical report, Microsoft Research, June 2007.

[Dam90] I. Damgård. On the Randomness of Legendre and Jacobi Sequences. In *Advances in Cryptology, CRYPTO '88, volume 403 of Lecture Notes in Computer Science*, pages 163–172, London, UK, 1990. Springer-Verlag.

[DFM00] R. Dingledine, M. J. Freedman, and D. Molnar. The Free Haven Project: Distributed Anonymous Storage Service. In H. Federrath, editor, *Proc. of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability (PET)*. Springer-Verlag, LNCS 2009, July 2000.

[DMS04] R. Dingledine, N. Matthewson, and P. Syverson. Tor: The Second-G.ration Onion Router. In *Proc. of the 13th USENIX Security Symp.*, San Diego, CA, August 2004.

[DR99] J. Daemen and V. Rijmen. Aes proposal: Rijndael, 1999.

[Dro] Dropbox: Secure backup, sync and sharing made easy. Available at . `http://www.dropbox.com/`.

[FB02] W. Ford and M. S. Baum. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption,* 2^{nd} *Ed.* Prentice Hall, 2002.

[FH02] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. RFC 3281, IETF, 2002.

[Fil] Filesystem in Userspace . `http://fuse.sourceforge.net/`.

[FO99] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Advances in Cryptology, CRYPTO '99, volume 1666 of Lecture Notes in Computer Science*, pages 537–554, London, UK, 1999. Springer-Verlag.

[FSG+01] D. F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST Standard for Role-based Access Control. *ACM Trans. on Info. and Sys. Sec.*, 4(3):224–274, 2001.

[GA05] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, New York, NY, USA, 2005. ACM.

[Gam85] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[GGHL05] C. Grothoff, K. Grothoff, T. Horozov, and J.T. Lindgren. An Encoding for Censorship-Resistant Sharing. Technical report, GNUnet project,http://gnunet.org/download/ecrs.pdf, 2005.

[GIKM98] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 151–160, New York, NY, USA, 1998. ACM.

[GMCC10] P. Gasti, A. Merlo, G. Ciaccio, and G. Chiola. On the Integrity of Network Coding-based Anonymous P2P File Sharing Networks. In *Proceedings of The Nineth IEEE International Symposium on Networking Computing and Applications, NCA 2010*, Cambridge, Massachusetts, USA, 2010. IEEE.

[Goo] Google Storage. `http://www.google.com/accounts/PurchaseStorage`.

[GP06] P. Gasti and V. Patil. Interdomain Access Control, 2006. `http://www.disi.unige.it/person/GastiP/publications/interac/`.

[GPG] GPG. The GNU Privacy Guard. `http://www.gpg.org/`.

[GR06] C. Gkantsidis and P. Rodriguez. Cooperative security for network coding file distribution. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, 2006.

[Hal05] S. Halevi. A Sufficient Condition for Key-Privacy. In *Cryptology ePrint Archive, Report 2005/05*, 2005. Available at `http://eprint.iacr.org/2005/005`.

[Hida] Hidden Operating Systems with TrueCrypt . `http://www.truecrypt.org/docs/?s=hidden-operating-system`.

[Hidb] Hidden Volumes with TrueCrypt . `http://www.truecrypt.org/hiddenvolume`.

[HL02] J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 466–481, London, UK, 2002. Springer-Verlag.

[HMM+00] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid, and D. Naor. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. In *SP '00: Proc. of the IEEE Symp. on Security and Privacy*, pages 2–14, 2000.

[HT05] R. Hayashi and K. Tanaka. Universally Anonymizable Public-Key Encryption. In *Advences in Cryptologu, ASIACRYPT '05, volume 3788 of Lecture Notes in Computer Science*, pages 293–312, London, UK, 2005.

[Ibr09a] M. Ibrahim. A Method for Obtaining Deniable Public-key Encryption. *International Journal of Network Security*, 8(1):1–9, 2009.

[Ibr09b] M. Ibrahim. Receiver-deniable Public-key Encryption. *International Journal of Network Security*, 8(2):159–165, 2009.

[ITU05] ITU X.509 Recommendations. Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks, 2005.
`http://www.itu.int/rec/T-REC-X.509/en`.

[JBBG04] J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor. Access-control language for multidomain environments. *IEEE Internet Computing*, 8(6):40–50, 2004.

[JD96] D. Jonscher and K. R. Dittrich. Argos – Configurable Access Control System for Interoperable Environments. In *Proc. of the 9^{th} annual IFIP TC11 WG11.3 working conf. on Database security IX : status and prospects*, pages 43–60. Chapman & Hall Ltd., 1996.

[KD04] K. Kurosawa and Y. Desmedt. A New Paradigm of Hybrid Encryption Scheme. In *Advances in Cryptology, CRYPTO '04, volume 3152 of Lecture Notes in Computer Science*, pages 426–442, London, UK, 2004.

[KFM04] M. N. Krohn, M. J. Freedman, and D. Mazieres. On-the-fly verification of rateless erasure codes for efficient content distribution. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 226–240, 2004.

[KKK08] M. Klonowski, P. Kubiak, and M. Kutylowski. Practical Deniable Encryption. In *SPFSEM'08*, volume 4910 of *LNCS*, pages 599–609, 2008.

[KL07] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2007.

[KMB08] M. Kim, M. Médard, and J. Barros. Counteracting byzantine adversaries with network coding: An overhead analysis. *CoRR*, 2008.

[KPF01] M. H. Kang, J. S. Park, and J. N. Froscher. Access Control Mechanisms for Inter-organizational Workflow. In *SACMAT '01: Proc. of ACM Symp. on Access Control Models & Tech.*, pages 66–74, 2001.

[KR00] H. Krawczyk and T. Rabin. Chameleon signatures. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2000, San Diego, California, USA*. The Internet Society, 2000.

[KV04] T. Kohno and J. Viega. Doug whiting: Cwc: A high-performance conventional authenticated encryption mode. In *Proceedings of Fast Software Encryption 2004, LNCS vol 3017, Springer-Verlag*, pages 385–402, 2004.

[LKXR05] J. Liang, R. Kumar, Y. Xi, and K. W. Ross. Pollution in p2p file sharing systems. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA*, pages 1174–1185, 2005.

[LMW02a] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, 2002.

[LMW+02b] N. Li, J. C. Mitchell, W.H. Winsborough, K.E. Seamons, M. Halcrow, and J. Jacobson. RTML: A Role-based Trust-management Markup Language. Technical report, Purdue University, 2002.

[LN99] J. Linn and Magnus Nyström. Attribute certification: an enabling technology for delegation and role-based controls in distributed environments. In *RBAC*

'99: Proc. of the 4th ACM workshop on Role-based access control, pages 121–130, 1999.

[Mir08] N. Mirzaei. Cloud Computing. Technical report, Indiana University, 2008.

[Moo65] G. E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1965.

[MV04] D.A. McGrew and J. Viega. The galois/counter mode of operation (gcm), 2004.

[MW09] B. Meng and J. Wang. A Receiver Deniable Encryption Scheme. In *Proceedings of the International Symposium on Information Processing, ISIP 2009, Huangshan, China*, 2009.

[Nat00] National Institute of Standards and Technology. *FIPS PUB 186-2: Digital Signature Standard (DSS)*. National Institute for Standards and Technology, Gaithersburg, MD, USA, January 2000.

[NISa] NIST. Heap-based NIST.buffer overflow in Microsoft Outlook Express 6 and earlier, and Windows Mail for Vista, allows remote Network News Transfer Protocol (NNTP) servers to execute arbitrary code via long NNTP responses that trigger memory corruption. `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-3897`.

[NISb] NIST. The Case for Elliptic Curve Cryptography. Available at `http://www.nsa.gov/ia/industry/crypto_elliptic_curve.cfm`.

[NISc] NIST. Buffer overflow in the Advanced Search (Finder.exe) feature of Microsoft Outlook 2000, 2002, and 2003 allows user-assisted remote attackers to execute arbitrary code via a crafted Outlook Saved Searches (OSS) file that triggers memory corruption, aka "Microsoft Outlook Advanced Find Vulnerability". `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-0034`.

[NISd] NIST. Heap-based buffer overflow in Microsoft Outlook Express 6 and earlier, and Windows Mail for Vista, allows remote Network News Transfer Protocol (NNTP) servers to execute arbitrary code via long NNTP responses that trigger memory corruption. `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-3897`.

[NISe] NIST. Microsoft Outlook 2002 and 2003 allows user-assisted remote attackers to execute arbitrary code via a malformed VEVENT record in an .iCal meeting request or ICS file. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-0033.

[NISf] NIST. Stack-based buffer overflow in the Vector Graphics Rendering engine (vgx.dll), as used in Microsoft Outlook and Internet Explorer 6.0 on Windows XP SP2, and possibly other versions, allows remote attackers to execute arbitrary code via a Vector Markup Language (VML) file with a long fill parameter within a rect tag. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2006-4868.

[NPS01] D. Naccache, D. Pointcheval, and Jacques Stern. Twin signatures: an alternative to the hash-and-sign paradigm. In *ACM Conference on Computer and Communications Security*, pages 20–27, 2001.

[Ope] OpenSSL Project. http://www.openssl.org.

[Pea09] S. Pearson. Taking Account of Privacy when Designing Cloud Computing Services. In *CLOUD '09: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 44–52, Washington, DC, USA, 2009. IEEE Computer Society.

[PH09] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. http://dud.inf.tu-dresden.de/Anon_Terminology.shtml, December 2009. v0.32.

[Pho] Phonebook open source project - Protecting your On-Disk Privacy with Deniable Encryption. Available at . http://www.freenet.org.nz/python/phonebook/.

[POR05] PORTIA – Privacy, Obligations and Rights in Technology of Information Assessment, 2005. http://crypto.stanford.edu/portia.

[PRI05] PRIME – Privacy and Identity Management for Europe, 2005. http://www.prime-project.eu.org.

[PRP] PRPQ. OpenCA PKI Project. http://www.openca.org.

[PS04] V. Patil and R. K. Shyamasundar. Towards a Flexible Access Control Mechanism for E-Transactions. In *EGCDMAS '04: International Workshop on*

Electronic Government, and Commerce: Design, Modeling, Analysis and Security, pages 66–81. INSTICC, 2004.

[Rab81] M. Rabin. How to Exchange Secrets by Oblivious Transfer. Technical report, Harvard Aiken Computation Laboratory, 1981.

[Rab89] M. O. Rabin. Efficient Dispersal of Information for Security, Load BA.cing, and Fault Tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[Rog00] P. Rogaway. Ocb mode: Parallelizable authenticated encryption, 2000.

[RP02] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proc. of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.

[RR99] M. K. Reiter and A. D. Rubin. Anonymity loves company: Anonymous web transactions with crowds. *Communications of the ACM*, 42, 1999.

[RSA] RSA Labs. RSA ID. `http://www.rsa.com/`.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Rub] Rubberhose Project . `http://iq.org/~proff/rubberhose.org/`.

[Sco02] M. Scott. Authenticated ID-based Key Exchange and Remote Log-in With Insecure Token and PIN Number. In *Cryptology ePrint Archive, Report 2002/164*, 2002. Available at `http://eprint.iacr.org/2002/164`.

[Sec05] Security Assertion Markup Language, 2005. `http://www.oasis-open.org/committees/security`.

[Ser] G. S. Sergienko. Self Incrimination and Cryptographic Keys, February 1996. Available at. `http://tinyurl.com/yjjzwe5`.

[Sha] Shamus Software. The MIRACL library. Available at `http://www.shamus.ie`.

[Sha79] A. Shamir. How to share a secret. *Comm. of the ACM*, 22(11):612–613, 1979.

[Sha84] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO 84*, pages 47–53, 1984.

[SHGA06] H. Skogh, J. Haeggstrom, A. Ghodsi, and R. Ayani. Fast freenet: Improving freenet performance by preferential partition routing and file mesh propagation. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, page 9, Washington, DC, USA, 2006. IEEE Computer Society.

[Shi05] Shibboleth, 2005. `http://shibboleth.internet2.edu/`.

[Sho] V. Shoup. A Proposal for an ISO Standard for Public Key Encryption (Version 2.1), manuscript, December 20, 2001. Available at `http://www.shoup.net/papers/iso-2_1.pdf`.

[Sho00] V. Shoup. Using Hash Functions as a Hedge against Chosen Ciphertext Attack. In *Advances in Cryptology, EUROCRYPT '00, volume 1807 of Lecture Notes in Computer Science*, pages 275–288, 2000.

[SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairing. In *Symposium on Cryptography and Information Security (SCIS 2000), Okinawa, Japan*, 2000.

[Spi92] M. R. Spiegel. *Theory and Problems of Probability and Statistics*. McGraw-Hill, 1992.

[SUN00] SUN Microsystems Inc. JavaCard 2.1.1 API Specification, Runtime Environment (JCRE) Specification and Virtual Machine Specification, 2000.

[SYJS01] D. Shands, R. Yee, J. Jacobs, and E. J. Sebes. Secure Virtual Enclaves: Supporting Coalition use of Distributed Application Technologies. *ACM Trans. Inf. Syst. Secur.*, 4(2):103–133, 2001.

[TJM+99] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based Access Control for Widely Distributed Resources. In 8^{th} *USENIX Security Symp.*, pages 215–228, 1999.

[Tru] Truecrypt: Free Open-source Disk Encryption Software for Windows 7/Vista/XP, Mac OS X, and Linux - Available at . `http://www.truecrypt.org/`.

[TWE+04] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820 (Proposed Standard), 2004.

[Vou08] M. A. Vouk. Cloud computing: Issues, research and implementations. In *Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on*, pages 31–40, August 2008.

[vOW96] P. C. van Oorschot and M. J. Wiener. On diffie-hellman key agreement with short exponents. In *EUROCRYPT*, pages 332–343, 1996.

[Web] Web Services Security v1.1. OASIS standards. `http://www.oasis-open.org/specs/index.php#wssv1.1`.

[WO02] B. Wilcox-O'Hearn. Experiences Deploying a Large-Scale Emergent Network. In *Proc. of the 1st Int.l Peer To Peer Systems Workshop (IPTPS02)*, March 2002.

[WRC00] M. Waldman, A. Rubin, and L. Cranor. Publius: A Robust, Tamper-evident, Censorship-resistant and Source-anonymous Web Publishing System. In *Proc. of the 9th USENIX Security Symposium*, pages 59–72, August 2000.

[XAC05] XACML. *eXtensible Access Control Markup Language*. OASIS Std., `http://www.oasis-open.org/committees/xacml`, 2005.

[Yao82] A. C. Yao. Protocols for Secure Computations. In *Proceedings of the 21st Annual IEEE Symposium on the Foundations of Computer Science*, pages 160–164, 1982.

[YFDL04] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 354–363, New York, NY, USA, 2004. ACM.

[YWRG08] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, pages 1409–1417, 2008.