UNIVERSITY OF CALIFORNIA,
IRVINE

**Decentralized Security Services**

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Nitesh Saxena

Dissertation Committee:
Professor Gene Tsudik and Professor Stanislaw Jarecki, Co-Chairs
Doctor Claude Castelluccia
Professor Alice Silverberg

Summer 2006

The dissertation of Nitesh Saxena
is approved and is acceptable in quality
and form for publication on microfilm:

_____

_____

_____

Committee Co-Chair

_____

Committee Co-Chair

University of California, Irvine
2006

*To the memory of my dear Nani (maternal grandmother), who passed away recently!*

# Contents

# List of Figures

# List of Tables

# Acknowledgments

This dissertation is made possible due to the immense support and help of some wonderful people. In the following, I attempt to acknowledge these people who directly or indirectly influenced me and my journey towards a Ph.D.

First of all, I am short of words to thank my advisors Gene Tsudik and Stanislaw (Stas) Jarecki. I remember when I first came to UC Irvine, I had only a very murky understanding of what research is. Gene is immensely responsible in presenting to me a new and clearer world of research and grooming me into a researcher that I am today. He has been a great mentor (undoubtedly the best one around!) and at times, a very nice friend. Stas has been a great technical advisor to me. I learnt most foundational cryptography from him. It is due to his guidance and during the countless discussion hours that he spent with me, that I learnt to write good crypto papers, and to deliver delectable presentations. Together, Gene and Stas, have proven to be great co-advisors, who provided ample (financial, technical, and moral) support that I needed, that a Ph.D student always needs.

I am grateful to Claude Castelluccia from INRIA, France for serving on my dissertation committee. It was an honor and a pleasure to work (and share office space) with him when he visited UC Irvine and also when I visited INRIA. I am also thankful to Alice Silverberg for serving on my dissertation committee. Alice's presence in the committee and her useful comments helped improve the quality of the dissertation and the defense talk.

I was fortunate enough to spend one summer as an intern at Nokia Research Center, Finland. I thank N. Asokan, my mentor there, who introduced me to the "real world" research and showed me how to do it. The collaborative work that I did at Nokia is included in this dissertation. I am thankful to my co-authors Jan-Erik Ekberg, Kari Kostiainen and N. Asokan for being such a great team and for such a nice result.

I would like to thank Jeong Hyun Yi (Samsung Research, Korea), a former Ph.D. student in our research group. Jeong and I did plenty of work together, some of which appears in both of our theses. I thank Jeong for the camaraderie that both of us greatly enjoyed.

I would also like to acknowledge some other people who inspired and influenced me in one way or the other. I thank Bruno Crispo at University of Trento, Italy; Robert Deng and Jianying Zhou at I2R, Singapore; Aurelien Francillon at INRIA, France; Luigi Mancini and his group at University of Rome, Italy; Valtteri Niemi and Kaisa Nyberg at Nokia Research Center, Finland; Dirk Westhoff at NEC Reserach, Germany.

I am also thankful to Nasir Memon and Stuart Steele at Polytechnic University (my would-be employer) for their support and patience during the preparation of my dissertation and guiding me to future research prospects in the meanwhile.

I also thank all our group members for their support and friendship. They include my seniors Xuhua Ding (who is now at SMU, Singapore) and Shouhuai Xu (who is now at UT San Antonio), my contemporaries Einar Mykletun and Maithili Narasimha, and my juniors Karim El Defrawy, Jihye Kim, Di Ma, Michael Sirivianos, John Solis, Claudio Soriente and Ersin Uzun. I also thank Jung Hee Cheon, who is a visitor from SNU, Korea and my current office-mate.

I thank all my friends who are (or were) in Irvine, Santa Barbara, India and elsewhere. The list is endless and I would not even attempt to include it here.

Last, but not least, a special thanks and regards to my parents back in India. This dissertation would never have been possible without their far-fetched vision, and their unfaltering affection, support and encouragement. I also thank my elder brother and my dear Bhabhi (sister-in-law) for their love and support, and for hosting my numerous visits to India during the Ph.D.

# Curriculum Vitae

## Nitesh Saxena

### EMPLOYMENT

Assistant Professor in the department of Computer and Information Science, at Polytechnic University, Brooklyn (to begin September 2006)

### RESEARCH INTERESTS

All aspects of information security with emphasis on network and distributed system security and applied cryptography.

### EDUCATION

- **University of California, Irvine**
  **Ph.D.** in Information and Computer Science
  September 2002 - August 2006
  **Advisors**: Dr. Gene Tsudik and Dr. Stanislaw Jarecki
- **University of California, Santa Barbara**
  **M.S.** in Computer Science
  September 2000 - March 2002
  **Advisor**: Dr. Alan Konheim
- **Indian Institute of Technology, Kharagpur, India**
  **B.S.** in Mathematics and Computing
  September 1996 - May 2000

## RESEARCH PROJECTS

- **Membership Control in Ad Hoc Groups**
  *Ph.D. Research,* ***UC Irvine****. (December 2002 - August 2006)*
  *Joint work with Gene Tsudik, Stanislaw Jarecki, and Jeong H. Yi*

- **Secure Pairing of Ad Hoc Devices Using a Human-Authenticated Channel**
  *Summer Internship Work,* ***Nokia Research Center****, Helsinki, Finland (June - September 2005)*
  *Joint Work with N. Asokan, Jan-Erik Ekberg, Stanislaw Jarecki, Kari Kostiainen*

- **Security and Privacy of Pervasive Devices**
  *Fall Internship Work,* ***INRIA*** *Rhone-Alpes, Grenoble, France (October - December 2005)*
  *Joint Work with Claude Castelluccia*

- **Cryptanalysis of the Schlüsselzusatz**
  *M.S. Thesis,* ***UC Santa Barabara****.(June 2001 - March 2002)*
  *Joint work with Alan Konheim.*

## INDUSTRY EXPERIENCE

- Intern at **INRIA Rhone-Alpes**, Grenoble, France (Oct - Dec 2005)
- Intern at **Nokia Research Center**, Helsinki, Finland (June - Sept 2005)
- Intern at **Automated Total Systems Solutions**, Santa Ana, California (Jul - Sept 2003)

## PUBLICATIONS

1. C. Castelluccia, N. Saxena, and J. H. Yi. Robust Self-Keying Mobile Ad-Hoc Networks. In *Elsevier Computer Networks Journal, to appear*
2. N. Saxena. Public Key Cryptography sans Certificates in Ad Hoc Networks. In *Applied Cryptography and Network Security (ACNS), June 2006.*.

3. N. Saxena, J. Ekberg, K. Kostiainen, and N. Asokan. Secure Device Pairing based on a Visual Channel. In *IEEE Symposium on Security and Privacy (ISP), Oakland, appeared as extended abstract (6 pages), May 2006.*

4. N. Saxena, G. Tsudik, and J. H. Yi. Efficient Node Admission for Short-lived Mobile Ad Hoc Networks. In *International Conference on Networking Protocols (ICNP), November 2005.*

5. C. Castelluccia, N. Saxena, and J. H. Yi. Self-configurable Key Pre-distribution in Mobile Ad Hoc Networks In *IFIP Networking Conference, May 2005.*

6. S. Jarecki and N. Saxena. Further Simplifications in Proactive RSA Signatures. In *Theory of Cryptography Conference (TCC)*, February 2005.

7. S. Jarecki, N. Saxena, and J. H. Yi. An Attack on the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, October 2004.

8. N. Saxena, G. Tsudik, and J. H. Yi. Identity-based Access Control for Ad Hoc Groups. In *International Conference on Information Security and Cryptology (ICISC)*, December 2004.

9. N. Saxena, G. Tsudik, and J. H. Yi. Access Control in Ad Hoc Groups. In *International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P)*, October 2004.

10. N. Saxena, G. Tsudik, and J. H. Yi. Experimenting with Admission Control in P2P. In *International Workshop on Advanced Developments in Software and Systems Security (WADIS)*, December 2003.

11. N. Saxena, G. Tsudik, and J. H. Yi. Admission Control in Peer-to-Peer: Design and Performance Evaluation. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, October 2003.

## HONORS AND AWARDS

- Nominated for ACM Dissertation Award by Information and Computer Science, UC Irvine, August 2006

- Dissertation Fellowship, Information and Computer Science, UC Irvine, Summer 2006
- Best Student Paper Award, Applied Cryptography and Network Security (ACNS) conference, June 2006

## PATENTS

- *Method and System of Improved Security in Wireless Environments Using Out-of-Band Channel Communication.* Pending patent application (no. PCT/IB05/003107) filed at Nokia Research Center, Helsinki, October 2005.

# Abstract of the Dissertation

Decentralized Security Services

by

Nitesh Saxena

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 2006

Professor Gene Tsudik and Professor Stanislaw Jarecki, Co-Chairs

Many security services, such as authentication and key management, rely on trusted third parties (TTPs) which provide a common root of trust, thereby enabling secure communication among all users. However, in many applications it is impractical to assume universally trusted TTPs. A broad class of such applications involves ad hoc groups, which include peer-to-peer systems (P2P), and mobile ad hoc networks (MANETs). Another class of applications which can benefit from avoidance of a single centrally trusted TTP are critical online services, such as certification, revocation and time-stamping. Centralized operation of such security services is undesirable because it leads to a single point of failure.

The security needs of both ad hoc groups and the critical online services motivate research on decentralized security services. This thesis investigates three decentralized security services: (1) Establishment of secure communication between two human-operated devices, without relying upon a TTP, (2) Distributed digital signature schemes that enable any set of $t + 1$ group members to sign messages on behalf of the group, even in the presence of at most $t$ faulty members, and (3) Secure membership management and secure communication in ad hoc groups.

The main research contributions of this thesis can be summarized as follows:

We show how to establish secure communication two devices over a short-range wireless communication channel, using messages that are visually authenticated by the users of these devices. Our work builds and improves upon a prior proposal called Seeing-is-Believing [MPR05]. We show how secure communication can be established based on a visual channel even on devices that have very limited displaying capabilities, in the most limiting case, on a device whose display consists of a cheap single light-source, such as an LED. Our proposal enables secure channel establishment between devices such as a Bluetooth cell phone and a headset, a WiFi cell phone and an access point, etc.

We also present a novel provably secure cryptographic protocol for establishing security between devices connected with a low-bandwidth (e.g. 20-bits) authenticated channel, e.g., the visually authenticated channel in the above application. Unlike the previous Diffie-Hellman-based protocols for this task [Vau05][LAN05][PVar], our protocol can be based on RSA encryption, and is hence suitable for settings where two devices have "asymmetric" computational capabilities (e.g., a PC and a cell phone, a cell phone and a headset). Moreover our protocol requires transmission of $(log_2(n))$ fewer bits on the authenticated channel than the previous protocols, where $n$ is total number of devices using the protocol.

Next, we focus upon distributed RSA signature schemes. We present a practical key-recover attack on a recently proposed distributed (proactive) RSA signature scheme [LL00]. In our attack, an admissible threshold of malicious players can completely recover the RSA secret signing key in the course of the lifetime of this scheme. Moreover, based on a corrected use of the above scheme, we propose a new efficient provably secure proactive RSA signature scheme. The new scheme offers a simpler alternative to the best previously known proactive RSA scheme [Rab98], and is applicable to build efficient decentralized online certification, revocation and time-stamping services.

Finally, we turn our attention to secure membership management and secure communication in ad hoc groups. Firstly, we propose an efficient protocol to securely extend an ad hoc group in a distributed manner. Compared to prior proposals [KZL$^+$01, KLX$^+$02, LZK$^+$02, NTY03], our protocol has minimal communication requirements, namely a single round of asynchronous communication Secondly, we present a scheme to speed-up secure communication in ad hoc groups. Compared to prior proposals [KZL$^+$01, KLX$^+$02, LZK$^+$02, NTY03], our scheme incurs lower communication and computation overhead involved in public-key based operations. Our scheme can also be viewed as a threshold-tolerant identity-based cryptosystem under standard (discrete logarithm based) assumptions.

# Chapter 1

# Introduction

*In this chapter, we motivate the need for providing robust and secure decentralized services. We consider decentralized security services in "two-party" and "multi-party" settings. These services include, in the two-party case, establishment of secure communication between two human-operated devices; in the "multi-party" case, distributed signatures, and secure membership management and secure communication in decentralized groups. We summarize the contributions of the thesis towards providing these services.*

## 1.1 Motivation

A security service has one or more of the following fundamental goals: access control, authentication, confidentiality, data integrity, availability, and non-repudiation. These goals can be achieved by various means. For example, confidentiality can be achieved by encryption, non-repudiation by digital signatures, etc. Traditionally, security services are supported by a fixed centralized infrastructure or a trusted third party (TTP) (and thus, we call them *centralized security services*). For example, in a public-key infrastructure (PKI), such services are provided with the help of certification authorities (CAs). The CAs issue signed certificates to users binding the identity of each user with its public key. Once the users have certified public keys (and corresponding private keys), they can securely communicate with each other using encryption, digital signing and key establishment mechanisms. Note that the CAs need to be mutually trusted by the communicating users. Other exam-

ples of centralized security services, based on symmetric-key cryptography, include Kerberos [MNSS87], RADIUS [RRSW97], S/Key [Hal95] and so on. Undoubtedly, centralized security services are easily managed and are ideal for certain applications.

In contrast, in certain scenarios, a trusted centralized infrastructure or a trusted third party might be unavailable or impractical. We broadly classify such application scenarios in following two categories.

1. **Applications that lack a trusted centralized authority**

   There are various applications wherein it is undesirable to assume the existence of a trusted authority or a pre-existing infrastructure. Various peer-to-peer (P2P) systems and mobile ad hoc networks (MANETs) fall into this category. These applications allow an "ad hoc" group of nodes to be able to communicate with each other. However, the nodes might not necessarily have any prior-context (or shared secrets) with each other or a common CA(s) among them. (Imagine, for example, an ad hoc network consisting of two nodes, Alice's (Nokia) cell phone and Bob's (Compaq) PDA, or a (file-sharing) P2P system, where a user in the US wants to share its files with a user in Iraq.) Moreover, in these applications, requiring constant presence of a central fixed entity is not realistic. First, such an entity is a single point of failure. Second, it represents an attractive and high-payoff target for attacks. Third, topology changes due to mobility and node outages (especially in MANETs) may cause the central entity to be unreachable and thus unable to perform its duties in the parts of the network not connected to it.

   In the rest of the thesis, we collectively call such applications as *ad hoc groups*. Ad hoc groups have numerous proposed applications, not only in personal/commercial and defense-oriented settings, but also in rescue and disaster-recovery operations.

2. **Applications that can not afford centralized operation**

   Centralized operation in certain applications is undesirable because it is a security liability and it leads to a single point of failure. This is typically true for various internet security services such as online certification, revocation (e.g. OCSP [MAM+99]), digital time-stamping, etc.

The security needs of aforementioned applications, i.e., both ad hoc groups and the critical online security services, motivate research on *decentralized security services*. Note

that the ad hoc group applications are "naturally" decentralized while the critical online security services need to be "forced" to become decentralized.

## 1.2 Decentralized Security Services

In this section, we describe the decentralized security services that we consider in this thesis. The first security service is applicable in a two-party setting, while the rest in a group setting. In the rest of this chapter and throughout in the thesis, we use the terms group/network/system and member/node/player/user interchangeably.

### 1.2.1 Establishment of Secure Communication Between Two Human-Operated Devices

The popularity of short-range wireless technologies such as Bluetooth and Wireless Local Area Networking (WLAN) based on the IEEE 802.11 family of protocols is experiencing enormous growth. Newer technologies such as Wireless Universal Serial Bus[1] are around the corner and promise to be as popular. This rise in popularity of personal area networks (PANs) implies that an increasing proportion of the users of devices supporting short-range wireless communication are not technically savvy. Such users need very simple and intuitive methods for setting up their devices. Since wireless communication is easier to eavesdrop on and easier to manipulate, a common set up task is to bootstrap secure communication between the devices. In this thesis, we will use the term *device pairing* to refer to the operation of establishing secure communication between two devices over short-range wireless communication channels.[2]

Both security researchers and practitioners have been looking for intuitive techniques for ordinary users to be able to establish secure communication between their devices. Although the primary impetus comes from the need to secure short-range wireless communication, the issue of intuitive secure channel establishment is more generally applicable whenever ordinary users need to set up secure communication without the help of expert administrators or trusted third parties.

More concretely speaking, the problem of secure channel establishment is to enable

[1] `http://www.usb.org/developer/wusb`

[2] The term *pairing* was introduced in the context of Bluetooth devices. Other roughly synonymous terms include "bonding," and "imprinting".

two devices, which share no prior context or a TTP with each other, to agree upon a key that they can use to protect their subsequent communication. The association must be resistant to a man-in-the-middle adversary who tries to impersonate one or both of these devices in the process. The adversary is assumed to be capable of eavesdropping or modifying messages on the communication channel between the devices.

One approach to security establishment is to use an additional physically authenticatable channel, called an out-of-band (OOB) channel which is governed by humans, i.e., by the users operating these devices. The adversary is assumed to be incapable of modifying messages on the OOB channel, although it can eavesdrop on them. Examples of such OOB channels include, in device pairing applications, audio, visual, infra-red channels. In other applications (such as SSH or PGP), security association can be achieved, for example, using a postcard, fax, telephone call or voice message.

### 1.2.2 Distributed Signatures

Critical online services, such as certification, revocation and time-stamping, all require a centralized server (or a TTP) to sign certain documents as per the clients' requests. However, as mentioned before, assigning the signing operation to a single server is a security liability to server compromise and is a single point of failure. Therefore, we distribute the signing operation among a number of servers by a security service called "distributed signatures".

The idea of distributing a cryptosystem so as to secure it against corruption of some threshold of participating players is known as *threshold cryptography*. It was introduced in the proposals of Desmedt [Des87], Boyd [Boy89], Croft and Harris [CH89], and Desmedt and Frankel [DF90], which were based on Shamir's *polynomial secret-sharing* technique [Sha79].

A $(t+1, n)$ threshold signature scheme [DF90] enables any subgroup of $t+1$ members in a group consisting of $n > t$ members, to collaboratively sign a message on behalf of that group. This is achieved by secret-sharing the signature key, e.g. the RSA secret key, among the group members, and allowing them to compute a signature on some message via a distributed protocol where the members use the shares of the signature key instead of the key itself. The scheme is said to be *$t$-secure* if any coalition of at most $t$ corrupt members is unable to forge a valid threshold signature on any message which honest members would not sign, and *$t$-robust* if honest group members can efficiently produce a

valid signature even in the presence of at most $t$ malicious members. To achieve $t$-security, a threshold signature scheme must in particular protect the secrecy of the signature key as long as no more than $t$ of the group members are corrupt.

A *proactive* signature scheme [HJJ+97a], based on techniques of proactive secret sharing [OY91, HJKY95], is a threshold signature scheme which remains secure and robust even if in every *time period*, called "share update interval", a possibly different set of $t$ group members is corrupted. This is achieved by the members periodically updating their shares of the secret signature key via a distributed share update protocol. Such an update protocol should destroy the correlation between secret shares learned by corrupted members in different time periods, so that the scheme can tolerate any number of corruptions throughout its lifetime as long as in any single time period the number of simultaneously corrupted members does not exceed $t$. A proactive signature scheme offers stronger security guarantee then a threshold scheme, especially in an application which might come under repeated attacks, such as a certification authority or a time-stamping service. Efficiency of a proactive signature scheme is very important in some applications, e.g., in a time-stamping service, or in the decentralized control of peer-to-peer groups, ad-hoc groups, or sensor networks [KZL+01, STY03]. An efficient proactive scheme for RSA signatures is especially important because RSA signatures are widely used in practice, and because verification of RSA signatures is several orders of magnitude faster than verification of other signatures.

### 1.2.3 Secure Membership Management and Secure Communication in Ad Hoc Groups

We consider dynamic ad hoc groups consisting of more than two nodes. As already pointed out previously, such groups have many well-known applications in military, commercial and personal settings as well as in emergency and rescue operations. However, lack of infrastructure and lack of centralized control make ad hoc groups inherently insecure, and therefore specialized security services are needed for their deployment. There are essentially three security-related problems in ad hoc groups:

- **Problem 1:** how to securely form the group,
- **Problem 2:** how to securely extend the group, and
- **Problem 3:** how to enable secure communication among the players.

In this thesis, our main focus is only on Problem 2 and 3 above. However, in our solutions, we also consider Problem 1, i.e., how to form the group in a distributed manner.

Next, we describe Problem 2 and 3 in detail. In short, Problem 2 requires a player to be securely added to the group whereby providing it with a credential; Problem 3 requires the players to be able to securely communicate with each other using their respective credentials.

**Problem 2.** We call Problem 2 *admission control*. Admission control is a fundamental security service in ad hoc groups; it is required to ascertain membership eligibility and to bootstrap other important security services, such as secure routing (e.g., [HPJ02, HJP02]) and secure group communication (e.g., [STW00b, STW00a]).

Due to the lack of a trusted authority in ad hoc groups, admission control must be performed in a decentralized manner, by the existing players. To achieve this, the natural technology to consider is threshold cryptography, as was first suggested by Zhou and Haas in their seminal proposal [ZH99]. The idea is to distribute the trust among the players of the group using threshold cryptography. This allows any set of $t+1$ players to admit a new player via a distributed admission control protocol.

Two features of ad hoc groups make admission control a very challenging problem. First, the devices in such groups often have very weak computational facilities and battery power. Second, the nodes usually function in an asynchronous (on/off) manner, often becoming temporarily unavailable. Therefore, an ideal admission control protocol must be efficient in terms of both computation and communication[3]. It must also involve minimal (ideally, *none* at all) interaction among the players of the group.

**Problem 3.** Once players join the group via their respective admission control protocols as described above, they need to securely communicate with each other. Based on the type of ad hoc group application, such secure communication might require confidentiality, authentication, non-repudiation, etc. Similarly, other advanced security functionalities, such as secure group communication [STW00b, STW00a], privacy-preserving authentication or secret handshakes [BDS+03, CJT04], multi-signatures [OMR01], aggregate signatures [BGLS03], and so on, might also be needed. The main challenge in providing such secure communication is to reduce the computation and communication overhead on the players.

[3]Communication is directly related to the consumption of battery power in mobile devices [BA03].

## 1.3 Design Goals and Challenges

In this section, we discuss the main design goals and challenges towards providing robust and secure decentralized services.

**Decentralization.** In the traditional client-server model, information is concentrated in centrally located servers and distributed through networks to clients that act primarily as user interface devices. Such centralized systems are ideal for some applications and tasks. For example, access rights and security are more easily managed in centralized systems. However, the topology of the centralized systems inevitably yields inefficiencies, bottlenecks, wasted resources, and single points of failure. A distributed signature application or a dynamic ad hoc group is a fully decentralized system where every entity is an equal participant. Clearly, the implementation of such applications is difficult because there is no centralized server with a global view of all the players in the group.

**Usability.** In security applications that involve human-intervention, usability is an important element. Mechanisms that are not friendly for the users to perform, are unlikely to be adopted in practice. Usability is certainly a primary design goal in device pairing applications, wherein ordinary users are involved.

**Self-Configurability.** Self-Configurability is a desirable feature in ad hoc groups, because they exhibit a lack of fixed infrastructure and possess no a-priori knowledge. That is, the group must be configured spontaneously by the players themselves.

**Robustness/Fault-Tolerance.** Fault-tolerant functioning of ad hoc groups is essential. They are faced with failures commonly associated with systems spanning multiple hosts and networks: disconnections, unreachability, partitions, and node failures. Such failures may be more pronounced in wireless than wired networks. Therefore, it would be desirable to continue operation among the still-connected players in the presence of failures. Similarly, a distributed signature application needs to be robust to both natural and malicious faults.

**Scalability.** P2P systems and MANETs may consist of a large number of nodes/players. Group size may continue to change as players leave and new players join. Thus, the security mechanisms have to scale with the group size.

**Non-Interactivity.** Because of the dynamic nature of most ad hoc group applications, nodes operate in an asynchronous (on/off) manner, often becoming temporarily unavailable. Since we cannot assume synchronous communication (which requires a common global clock), the security mechanisms must involve minimal interaction among the nodes. Interaction may be direct among nodes, or, indirect, through a third party, e.g. a new node wanting to join the group. The ideal mechanism must minimize both direct and indirect interaction.

## 1.4 Summary of the Contributions

The contributions of this thesis are summarized as follows.

- **Secure Device Pairing Using a Visual Channel**

  [This work appeared in [SEKA06]]

  We focus upon how to implement device pairing using a manually authenticated OOB channel. In particular, we focus upon a visual OOB channel. Our work builds upon a prior proposal by McCune et al. [MPR05], a system called Seeing-is-Believing (SiB). The SiB visual channel consists of one device displaying the hash of its public key in the form of a two-dimensional barcode, and the other device reading this information using a photo camera. Strong mutual authentication in SiB requires running two separate unilateral authentication steps.

  In this thesis, we show how strong mutual authentication can be achieved even with a unidirectional visual channel, where SiB could provide only a weaker property referred to as *presence*. This could help reduce the SiB execution time and improve usability. By adopting recently proposed improved pairing protocols (see below for one such protocol that we propose in this thesis), we propose how visual channel authentication can be used even on devices that have very limited displaying capabilities, in the most limiting case, on a device whose display consists of a cheap single light-source, such as an LED. We also describe a new video codec that may be used to improve execution time of pairing in limited display devices, and can be used for other applications besides pairing.

- **Authenticated Key Agreement Using Short Authenticated Strings**

[This work is under submission [JS06]]

Recently, Vaudenay introduced a class of authentication protocols based on Short Authenticated Strings (SAS) [Vau05]. Such protocols allow authenticating messages of arbitrary length sent on insecure channels, between devices which can additionally send a very short, e.g. 20 bit long, *authenticated* message to each other. In general, these protocols can be used for authenticating arbitrary long messages. However, their primary application is to enable SAS-based authenticated key agreement (SAS-AKA) between any two devices with no reliance on key pre-distribution or a public-key infrastructure. The protocol applies to device pairing (as we discussed above), SSH, PGP, etc. The two subsequent proposals [LAN05, PVar] reduced the round complexity from four to three rounds in the SAS-based "message cross-authentication" (SAS-MCA) protocol proposed in [Vau05], where authentication is achieved for messages flowing bi-directionally between the two devices. Either of these 3-round SAS-MCA protocols implies a three-round Diffie-Hellman-based SAS-AKA [PVar].

We propose a three-round *encryption-based* SAS-AKA protocol (which is also a SAS-MCA protocol). The cost of our protocol, in the random oracle model, is a single public key encryption for one party and a decryption operation for the other, and therefore our encryption-based SAS-AKA protocol is a useful alternative to the Diffie-Hellman-based protocols especially in settings where the two devices have different computational powers, e.g. a PC and a cell-phone, a cell-phone and a headset, etc. Additionally, by proving security of our SAS-AKA protocol directly (instead of the modular approach used by [PVar]), we also show a better exact security guarantee for the encryption-based SAS-AKA compared to what was shown for the Diffie-Hellman SAS-AKA's. This improvement implies same security guarantees on a SAS channel with a bandwidth reduced by $log_2(n)$ bits, where $n$ is the total number of devices using the protocol.

- **Distributed Signatures**

  [This work appeared separately in [JSY04] and [JS05]]

  In this thesis, we focus upon RSA-based distributed (threshold/proactive) signatures. Recently, Luo, et al. [LL00] proposed a new proactive RSA scheme (called URSA) applied to access control in mobile ad hoc networks. The URSA proactive RSA is

assumed secure in the presence of an adversary who simultaneously corrupts at most a threshold of players in the lifetime of the scheme.

In this thesis, we first show an attack on this URSA proactive RSA scheme, wherein an admissible threshold of malicious players can completely recover the RSA secret signing key in the course of the lifetime of this scheme. Our attack stems from the fact that the threshold signature protocol which is a part of this proactive RSA scheme leaks some seemingly innocuous information about the secret key.

We then carry on to construct a new (provably secure) proactive RSA scheme based on the URSA proactive RSA. The new scheme offers a simpler alternative to the best previously known proactive RSA scheme given by Rabin [Rab98], itself a simplification over the previous schemes given by Frankel et al. The new scheme is conceptually simple because all the sharing and proactive re-sharing of the RSA secret key is done modulo a *prime*, while the reconstruction of the RSA signature is done using an equation over *integers*. Efficiency-wise the new scheme gives a factor of 2 improvement in speed and share size in the general case, and almost a factor of 4 improvement for the common RSA public exponents 3, 17, or 65537, over the scheme of Rabin. [Rab98]. The new proactive RSA is applicable to build online certification, revocation and time-stamping services.

- **Secure Membership Management and Secure Communication in Ad Hoc Groups**

  A straight-forward approach to solve the Problem 1 and Problem 2, as described in section 1.2.3, is by employing threshold cryptography to distribute the role of the certification authority among the players of the group. Each player is issued a signed certificate and a secret value (which we call a *secret share* of the group secret) by this distributed certification authority. The certificates are used by the players to securely communicate with each other, as is done traditionally. The secret share is used by a player to admit new players in the future. Starting with the seminal proposal by Zhou and Haas [ZH99], this approach was adopted by various subsequent proposals [KZL$^+$01, KLX$^+$02, LZK$^+$02, NTY03, STY03, STY04]. However, this approach has two main drawbacks. First, the distributed protocols to provide certificates and secret shares to new players, are prohibitively expensive (especially for highly dynamic environments), both in terms of computation and (especially) communication, since they

require a set of existing players to be fully connected with each other at all times. Recall that in many ad hoc groups, connectivity can be unstable and the devices are weak or resource-limited. Second, traditional solutions for secure communication (e.g., Diffie-Hellman key exchange) using certificates are also not feasible for many ad hoc groups as they incur high communication overhead.

In this thesis, we make following contributions.

– **Decentralized Admission Protocol**

  [This collaborative work first appeared in [STY05], and is also included in the dissertation by Jeong Hyun Yi [Yi05]]

  We first show that there is no need for node-specific certificates in ad hoc groups; in other words, only secret shares are sufficient. This implies that during the admission protocol, the need for certificate issuance (and thus distributed signing) is obviated. Second, we present a secure, efficient and a fully non-interactive admission control protocol to provide new secret shares. Our protocol requires only a single round of asynchronous communication and is developed using secret sharing techniques based on bi-variate polynomials.

– **Efficient Secure Communication**

  [This work appeared in [Sax06]]

  We show how to perform necessary public key operations without node-specific certificates in ad hoc groups. These operations include pair-wise key establishment, signing, and encryption. We achieve this by using Feldman's verifiable polynomial secret sharing (VSS) as a key distribution scheme and treating the *secret shares as the private keys*. Unlike the standard public key cryptography, where entities have independent private/public key pairs, in the proposed scheme the private keys are *related* (they are points on a polynomial of degree $t$) and each public key can be computed from the public VSS information and node identifier. We show that such related keys can still be securely used for standard signature and encryption operations (using resp. Schnorr signatures and ElGamal encryption) and for pairwise key establishment, as long as there are no more that $t$ collusions/corruptions in the system.

  The proposed usage of shares as private keys can also be viewed as a threshold-tolerant identity-based cryptosystem under standard (discrete logarithm based)

assumptions.

## 1.5 Relevant Applications

We believe that this thesis can be useful in various application domains, some that exist today and many more that are emerging. We discuss some of the applications in the following.

- **Personal Communication.** Short-range, such as Bluetooth and WiFi communication of personal devices is becoming increasingly popular and ubiquitous. Examples include communication between a bluetooth cell phone and a headset, a WiFi laptop and an access point, etc. Such communication is routinely performed not only in homes and offices, but also in commercial settings. The secure device pairing problem that we propose to solve in this thesis is essential to secure such personal communication. In addition, our SAS-AKA protocol is more generally applicable to securely initialize any two personal devices. For example, it can be used for public-key authentication in SSH or PGP sessions, by using a postcard, fax, telephone call or voice message.

- **Critical On-line Security Services.** On-line security services, such as a certification, revocation and digital time-stamping, have been considered by both researchers and practitioners. In this thesis, we propose building such fault-resilient and robust security services using distributed RSA signing. Note that RSA is a trusted standard for digital signing, widely used in practice today and also has efficient verification.

- **Military Mobile Ad Hoc Networks.** Since its introduction as a communication medium, wireless technology found broad application in the military environments. Because of dramatic advances in wireless technology and the capabilities associated with small computing devices, the demand for advanced mechanisms to employ wireless technology in the battlefield continues to grow. Secure communication is critical in the battlefield where the network infrastructure is vulnerable to various attacks and compromises. Conventional centralized solutions break down when security servers are destroyed. Thus, distributed security solutions that we propose are needed for securely distinguishing trusted from untrusted computers.

- **Private P2P File Sharing.** One fairly obvious application that we envision is in the domain of private P2P groups formed atop wide-open (essentially public) P2P systems, such as Kazaa, Morpheus, and Gnutella. In fact, a recent article in the Time Magazine [Ham03] examines the popular trend of creating so-called *"Darknets"* [BEPW02] – secure private groups within Kazaa and Morpheus – in order to escape the intensified crackdown on music and other content sharing.

- **Wireless Sensor Networks.** A wireless sensor network (WSN) is a mesh network of small sensor nodes communicating among themselves using RF communication, and deployed in large scale (from tens to thousands) to sense the physical world. Sensor nodes can be imagined as small computers, extremely basic in terms of their interfaces, with weak memory, and computation and battery power. Generally, the information sensed by the sensor nodes is routed to a central computer, called a base station (BS), which is governed by the user (or the administrator). Because of the nature of various proposed applications of WSNs, the sensor nodes must be able to securely communicate with each other and the administrator must be able to securely and easily manage the nodes. Some solutions that we propose in this thesis incur low computation, power and memory overhead, can be used to provide critical security services in WSNs.[4]

- **Self-organizing Transportation Management.** Another example of a somewhat futuristic (yet viable and emerging) application is in the form of a distributed and self-organizing transportation management and control system, wherein a number of GPS-equipped vehicles form an ad hoc group and communicate among themselves regarding traffic conditions in order to obtain advance notice of traffic jams and choices of optimal routes [Theml]. In such a system, secure decentralized services that we consider are certainly essential.

- **Distributed File/Storage Systems**: Storage systems have evolved into storage area networks (SANs) where hosts and disks are interconnected using IP. WAN customers often entrust security services such as data integrity and confidentiality to

[4] Although our solutions are based on public-key cryptography, they can be employed in sensor networks – it has been shown that most sensor nodes are sufficiently capable of performing public-key operations [GKS04].

third-party service providers. A secure group communication system can provide security as well as reliable message delivery on SANs independent of third-parties.

- **Collaborative Workspaces**: Collaborative and distributed workspaces require secure and reliable group communication. For example, Collabnet [Col] implements web-based work spaces that support collaborative programming, revision control, issue tracking and mailing list management. Since collaborators are often distributed, security is clearly a major concern.

## 1.6 Thesis Organization

We divide the thesis into two parts: Part I and Part II. Part I describes security services for a two-party setting and consists of Chapters 3 and 4. Part II describes security services for multi-party setting and consists of Chapters 6, 7 and 8.

The chapters are organized as follows.

- First, in Chapter 2, we introduce the cryptographic preliminaries that we utilize in our various constructions in the rest of the thesis.

- In Chapter 3, we show how to implement secure device pairing between two devices based on a visual channel. McCune, et al. [MPR05] proposed that one device displays the hash of its public key in the form of a barcode, and the other device reads it using a camera. Mutual authentication requires switching the roles of the devices and repeating the above process in the reverse direction. We show how strong mutual authentication can be achieved even with a unidirectional visual channel, without having to switch device roles. By adopting recently proposed improved pairing protocols, we propose how visual channel authentication can be used even on devices that have very limited displaying capabilities, such as a single blinking LED.

- In Chapter 4, we present a new efficient encryption-based protocol for securely associating two devices. The protocol is based on short (e.g., 20-bit long) manually authenticated channels. Our protocol is a useful alternative to the prior Diffie-Hellman-based protocols, especially in settings where the two devices have different computational powers, e.g. a PC and a cell-phone, a cell-phone and a headset, etc.

- In Chapter 5, we provide a detailed background of threshold cryptography. The techniques that we describe in this chapter are used in our constructions in the following chapters.

- In Chapter 6, we focus upon RSA-based distributed (threshold/proactive) signatures. We first present an efficient attack on a recently proposed proactive RSA scheme [LKZ+04], in which an admissible threshold of malicious group members can completely recover the group RSA secret key in the course of the lifetime of this scheme. We then carry on to construct a new (provably secure) proactive RSA scheme based on a corrected use of the scheme of [LKZ+04]. The new scheme offers a simpler alternative to the best previously known proactive RSA scheme given by Tal Rabin.

- In Chapter 7, we present a secure, efficient and a fully non-interactive admission control protocol for ad hoc groups. Our work is focused on novel applications of non-interactive secret sharing techniques based on bi-variate polynomials

- In Chapter 8, we show how to perform necessary public key operations without node-specific certificates in ad hoc groups. These operations include pair-wise key establishment, signing, and encryption.

# Chapter 2

# Cryptographic Preliminaries

*This chapter describes some cryptographic preliminaries. Some of the preliminaries are used in our various constructions in the rest of the thesis; some others are used in the constructions of schemes that our proposals compare to.*

## 2.1 Cryptographic Background

**Cryptography** is the study of mathematical techniques used to achieve various goals in information security, such as confidentiality, data integrity, entity authentication, data origin authentication, non-repudiation, etc.

Among these, confidentiality can be achieved by symmetric key encryption or asymmetric (public) key encryption. Consider an encryption scheme consisting of the sets of encryption and decryption transformations $\{E_y : y \in \mathcal{K}\}$ and $\{D_x : x \in \mathcal{K}\}$, respectively, where $\mathcal{K}$ is the key space. An encryption scheme is said to be **symmetric-key** if for each associated encryption/decryption key pair $(y, x)$, it is computationally easy (and hard otherwise) to determine $x$ knowing only $y$, and to determine $y$ from $x$. An encryption scheme is said to be **public-key** if, for each associated encryption/decryption key pair $(y, x)$, one key $y$ (the public key) is made publicly available, while the other $x$ (the private key) is kept secret. For the scheme to be secure, it must be computationally infeasible to compute $x$ from $y$.

**Definition 2.1 (Public-Key Encryption Scheme).** *A **public-key encryption scheme***

*$\mathcal{E}$ is a triple of algorithms (*KeyGen*,* Enc*,* Dec*), where the following conditions hold:*

1. KeyGen *is the (randomized)* ***key generation*** *algorithm: on input of a random string, it outputs a pair $(pk, sk)$, such that $pk$ is the* public key *and $sk$ is the* private key *of the encryption scheme.*

2. Enc *is the (randomized)* ***encryption*** *algorithm: on input a plaintext* message $m$ *and the* public key $pk$*, it outputs ciphertext* c*, an encryption of a message $m$ under the public key $pk$, i.e.,* c=$\mathsf{Enc}_{pk}m$*.*

3. Dec *is the (deterministic)* ***decryption*** *algorithm: on input a* ciphertext $c$*, the* private key $sk$*, it outputs the corresponding plaintext* m=$\mathsf{Dec}_{sk}c$*, where* c=$\mathsf{Enc}_{pk}m$*. A ciphertext whose decryption does not belong to the message space, is said to be invalid.*

For an encryption scheme to remain secure even in the critical context where the messages are taken from a small set of plaintexts, the notion of indistinguishability [GM89] is used.

**Definition 2.2 (IND-CPA Security).** *Indistinguishability under chosen-plaintext attack,* IND-CPA*, is defined via the following game between an adversary and a challenger: A random key-pair $(pk, sk)$ is output by* KeyGen *on some security parameter. Adversary $\mathcal{A}$ on input $pk$ sends to the challenger $\mathcal{C}$ two messages $m_0$ and $m_1$ of his choice. $\mathcal{C}$ picks $b \in \{0, 1\}$ and sends back the ciphertext $c = Enc_{pk}(m_b)$. $\mathcal{A}$ responds with $\hat{b}$ and we say that $\mathcal{A}$ "wins" if $\hat{b} = b$. The adversary is free to perform encryptions on any number of adaptively chosen messages. An encryption scheme (for a given security parameter) is said to be $(T, \epsilon)$-*IND-CPA *if no adversary $\mathcal{A}$ running in time $T$ can win in this game with a probability greater than $1/2 + \epsilon$.*

**Definition 2.3 (IND-CCA Security).** *Indistinguishability under chosen-ciphertext attack,* IND-CCA*, is defined via the following game between an adversary and a challenger: A random key-pair $(pk, sk)$ is output by* KeyGen *on some security parameter. Adversary $\mathcal{A}$ on input $pk$ sends to the challenger $\mathcal{C}$ two messages $m_0$ and $m_1$ of his choice. $\mathcal{C}$ picks $b \in \{0, 1\}$ and sends back the ciphertext $c = Enc_{pk}(m_b)$. $\mathcal{A}$ responds with $\hat{b}$ and we say that $\mathcal{A}$ "wins" if $\hat{b} = b$. $\mathcal{A}$ is also given access to a decryption oracle, which returns* $\mathsf{Dec}(sk, c_i)$ *for any adaptively chosen ciphertexts $c_i$ chosen by $\mathcal{A}$, as long as for all $i$, $c_i \neq c$. An encryption scheme (for a given security parameter) is said to be $(T, \epsilon)$-*IND-CCA *if no adversary $\mathcal{A}$ running in time $T$ can win in this game with a probability greater than $1/2 + \epsilon$.*

Note that IND-CCA is a stronger security notion than IND-CPA. In other words, an IND-CCA-secure scheme is also IND-CPA-secure. IND-CPA-security is also called semantic security.

Other than confidentiality, public key cryptography can be used to provide authentication and non-repudiation using **digital signatures**. The purpose of a digital signature is to provide a means for a signer to bind its identity to a piece of information. A signature scheme consists of three components: a key generation algorithm, a signing algorithm, and a verification algorithm. Each entity should create a public/private key pair. With a private key, it can sign a message using a signing algorithm. The resulting signature *sig* can subsequently be verified using a public verification algorithm and the entity's public key $y$. Given a pair $(sig, y)$, the verification algorithm returns an answer "true" or "false" depending on whether the signature is authentic. Here is a formal definition of a signature scheme.

**Definition 2.4 (Signature Scheme).** *A **signature scheme** $\mathcal{S}$ is a triple of algorithms (*KeyGen*,* Sig*,* Ver*), where the following conditions hold:*

1. KeyGen *is the **key generation** (randomized) algorithm: on input of a random string, it outputs a pair $(pk, sk)$, such that $pk$ is the* public key *and $sk$ is the* private key *of the signature scheme.*

2. Sig *is the **signing** (randomized) algorithm: on input a* message $m$ *and the* private key $sk$*, it outputs* sig*, a signature of a message $m$ under the key $x$, i.e.,* sig=$\mathsf{Sig}_{sk}m$.

3. Ver *is the **verification** (deterministic) algorithm: on input a* message $m$*, the* public key $pk$*, and a string* sig*, it checks whether* sig *is a proper signature of $m$, i.e.,*

$$\mathsf{Ver}_{pk}(m, sig) = \begin{cases} true & \text{if sig} = \mathsf{Sig}_{sk}m \\ false & \text{if sig} \neq \mathsf{Sig}_{sk}m. \end{cases}$$

The standard notion for the security of a signature scheme is security against existential forgery on adaptively chosen message attack (CMA) [GMR84].

**Definition 2.5 (Existential Forgery under Chosen Message Attack).** *Existential forgery under chosen message attack,* CMA*, is defined as the following game between an adversary and a challenger. The challenger produces the public key, private key pair $(pk, sk)$*

*using* KeyGen. *The adversary adaptively chooses messages* $m_i$ *and sends it to the challenger, who signs* $m_i$ *with* $sk$*, i.e., computes* $sig_i = \mathsf{Sig}_{sk} m_i$*, and returns* $sig_i$ *to the adversary. Finally, the adversary outputs a pair* $(m, sig)$ *and wins if* $sig$ *is a valid signature on* $m$ *under* $sk$*, and* $m$ *is not equal to any of* $m_i$*.*

Symmetric-key and public-key cryptography have a number of complementary advantages. Current cryptographic systems exploit the strengths of each. An example will serve to illustrate.

Public-key encryption techniques may be used to establish a key for a symmetric-key system being used by communicating entities $A$ and $B$. In this scenario $A$ and $B$ can take advantage of the long term nature of the public/private keys of the public-key scheme and the performance efficiency of the symmetric-key scheme. Since data encryption is frequently the most time-consuming part of the encryption process, the public-key scheme used for key establishment is a small fraction of the total encryption process between $A$ and $B$. To date, the computational performance of public-key encryption is inferior to that of symmetric-key encryption. Consequently, the important points in practice are:

- public-key cryptography facilitates efficient digital signatures and key management
- symmetric-key cryptography is efficient for bulk data encryption and data integrity.

As long as there is a strong binding between the signer and the signer's public-key, the identity of the signer of a given message can be traced.

A **Public Key Infrastructure (PKI)** provides the means to bind public keys to their owners and helps in the distribution of public-keys in large heterogeneous networks. Public-keys are bound to their owners by **public key certificates (PKCs)**. These certificates contain information such as the owner's name and the associated public key and are issued by a reliable **Certification Authority (CA)**.

We define the components of a PKI below:

- **Public Key Certificate** - An electronic record that binds a public key to the owner of the public key and is signed by a trusted entity.
- **Certificate Revocation List (CRL)** - A list of certificates that have been revoked. The list is usually signed by the same entity that issued the certificates. Certificates

can be revoked for several reasons, for example, if the owner's private-key has been lost.

- **Certification Authority** - A trusted entity that issues and revokes public key certificates.

## 2.2 Cryptographic Assumptions

Some solutions that we propose in this thesis work in the standard discrete logarithm setting: $p, q$ are large primes s.t. $q$ divides $p-1$ and $g$ denotes a generator of subgroup $G_q$ of order $q$ in $\mathbb{Z}_p^*$. For definitional convenience we'll denote by $DL\text{-}INST(k)$ any set of instances of this discrete-log setting, i.e. of triples $(p, q, g)$ which satisfy the above constraints, but where $q$ is a $k$-bit prime and $p$ is $poly(k)$-bit prime, long enough to fend off known attacks on the discrete logarithm.

We call function $f$ *negligible* if for every polynomial $P(.)$, $f(k) \leq 1/P(k)$ for all sufficiently large k. We say that some event occurs with a negligible probability if the probability of this event is a negligible function of the security parameter k.

**Assumption 1 (Discrete Logarithm (DL) Assumption)** For every probabilistic polynomial time algorithm I, for every $(p, q, g)$ in $DL\text{-}INST(k)$, probability $Pr[x \leftarrow \mathbb{Z}_q; I(p, q, g, g^x) = x]$ is negligible.

**Assumption 2 (Computational Diffie-Hellman (CDH) Assumption)** For every probabilistic polynomial time algorithm I, for every $(p, q, g)$ in
$DL\text{-}INST(k)$, probability $Pr[x \leftarrow \mathbb{Z}_q; y \leftarrow \mathbb{Z}_q; I(p, q, g, g^x, g^y) = g^{xy}]$ is negligible.

**Assumption 3 (Square Computational Diffie-Hellman (SCDH) Assumption)** For every probabilistic polynomial time algorithm I, for every $(p, q, g)$ in $DL\text{-}INST(k)$, probability $Pr[x \leftarrow \mathbb{Z}_q; I(p, q, g, g^x) = g^{x^2}]$ is negligible.

Some of our solutions are based on the RSA assumption and the Strong RSA assumption. RSA is a public key cryptosystem, where the public key of a user is a pair $(N, e)$ and private key is $(p, q, d)$, such that $p$ and $q$ are two large primes, $N = pq$, and $ed = 1 \bmod [(p-1)(q-1)]$. Informally, *RSA problem* is to find $e^{th}$ roots modulo N, while *Strong RSA problem* is to find $r^{th}$ roots modulo N, where $r > 1$.

**Assumption 4 (RSA Assumption)** For every probabilistic polynomial time algorithm

I, probability $Pr[c \leftarrow \mathbb{Z}_N^*; I(N, e, c) = c^{1/e} \bmod N]$ is negligible, where $(N, e)$ denotes the RSA public key.

**Assumption 4 (Strong RSA Assumption)** For every probabilistic polynomial time algorithm I, probability $Pr[c \leftarrow \mathbb{Z}_N^*, r > 1; I(N, c) = (r, c^{1/r} \bmod N)]$ is negligible, where $N$ denotes the RSA modulus.

## 2.3 Random Oracle Model (ROM)

Some of our proofs of security are in the so-called Random Oracle Model [BR93], i.e. we model hash functions like MD5 or SHA1 as ideal random oracles. Doing security analysis in the ROM model effectively means that our proofs will consider only such attacks on the cryptographic schemes we propose whose success does not change if the fixed hash function like MD5 or SHA in these schemes are replaced with truly random functions. Of course, since functions like MD5 or SHA are not truly random functions, the security analysis in the ROM model provides only a heuristic argument for the security of the actual scheme. However, such heuristic seems the best we can currently hope for. Indeed, the ROM heuristic arguments are currently the only security arguments for most practical cryptographic schemes including OAEP RSA encryption [BR93] and full-domain hash RSA signatures [BR96], as well as the two fundamental discrete-log-based cryptosystems, the hashed ElGamal encryption [ElG99] and Schnorr signature scheme [Sch91, PS96].

## 2.4 ElGamal Encryption Scheme

In one of our constructions in this thesis, we will use a variant of ElGamal Encryption scheme, called *Hashed* ElGamal [ElG99], which is IND-CPA-secure under the CDH assumption in ROM. For a private key, public key pair $(x, y = g^x)$, the encryptor chooses a random $r \in \mathbb{Z}_q$ and computes the ciphertext $(c_1, c_2)$ where $c_1 = g^r \pmod p$ a $c_2 = m \oplus H({y_i}^r)$ ($\oplus$ denotes the bit-wise XOR operator). The plaintext can be obtained by computing $c_2 \oplus H(c_1^{x_i})$ from the ciphertext $(c_1, c_2)$.

## 2.5 Schnorr Signature Scheme

The Schnorr signature scheme [Sch91] works as follows. The private key is $x$, chosen at random in $\mathbb{Z}_q$. The public key is $y = g^x \pmod p$. A signature [Sch91] on message $m$ is computed as follows. The signer picks a one-time secret $k$ at random in $\mathbb{Z}q$, and computes the signature on $m$ as a pair $(c, s)$ where $s = k + cx \pmod q, c = H(m, r)$, and $r = g^k \pmod p$. Signature $(c, s)$ can be publicly verified by computing $r = g^s y^{-c} \pmod p$ and then checking if $c = H(m, r)$. The Schnorr signature scheme is proven secure under the DL assumption in ROM against existential forgery under chosen message attack.

## 2.6 Commitment Schemes

In this thesis, we consider a tagged commitment scheme, where a portion of the committed value acts as a tag and can be published in the clear. The commitment scheme consists of following three functions.

- $\mathsf{gen}$ generates a public parameter $K_p$ on input of a security parameter.
- $\mathsf{commit}_{K_p}(m, R)$, on input $(m, R)$, outputs a pair of a "commitment" $c$ and "decommitment" $d$.
- $\mathsf{open}_{K_p}(m, c, d)$, on input a tag $m$ and the $(c, d)$ pair, either outputs some value $R$ or rejects.

This triple of algorithms must meet a completeness property, namely that for any $K_p$ generated by $\mathsf{gen}$ and for any $m$ and $R$, if $(c, d)$ is output by $\mathsf{commit}_{K_p}(m, R)$ then $\mathsf{open}_{K_p}(m, c, d)$ outputs $R$. For simplicity of notation, for our application we will assume a *common reference string* (CRS) model, where a trusted third party generates the commitment key $K_p$ and this key is then embedded in every instance of the protocol. Therefore we'll use a simplified notation, and write $\mathsf{commit}(m, R)$ and $\mathsf{open}(m, c, d)$ without mentioning the public parameter $K_p$ explicitly.

A commitment scheme is said to be $(T, \epsilon)$-*binding*, if no adversary $\mathcal{A}$ running in time $T$, can win the game shown in Figure 2.1 with a probability greater than $2^{-k} + \epsilon$. We call a commitment scheme *perfectly binding* if it is $(T, 0)$-*binding* for any $T$.

A commitment scheme is said to be $(T, \epsilon)$-*hiding* (in the sense of one-wayness), if no adversary $\mathcal{A}$ running in time $T$, can win the game shown in Figure 2.2 with a challenger

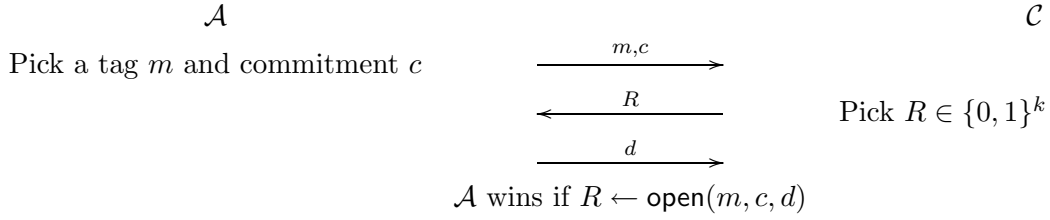



Figure 2.1: Binding Game for a Commitment Scheme

$\mathcal{C}$, with a probability greater than $2^{-k} + \epsilon$. A commitment scheme is said to be *perfectly hiding* if it is $(T, 0)$-*hiding* for any $T$.

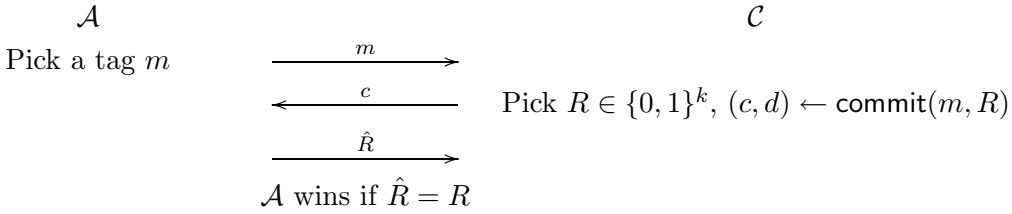



Figure 2.2: Hiding Game for a Commitment Scheme

## 2.7 Pedersen Commitment Scheme

A tag-less commitment scheme was proposed by Pedersen in [Ped91a]. This scheme is used in various protocols in threshold cryptography.

$\mathsf{gen}$ picks two prime numbers $p$ and $q$ such that $q$ divides $p-1$. Let $\mathbb{G}_q$ the unique subgroup of order $q$ of $\mathbb{Z}_p^*$ and let $g$ be a generator of $\mathbb{G}_q$. In addition, $\mathsf{gen}$ picks a random $h \in \mathbb{G}_q$. The public parameters are $(p, q, g, y)$; $log_g(h)$ is not known to anyone.

$\mathsf{commit}_{K_p}(x)$ algorithm on inputing $x \in \mathbb{Z}_q$ starts by picking a random $x' \in \mathbb{G}_q$ and then computes $c = g^x h^{x'}$. The decommit value $d = (x, x')$.

$\mathsf{open}_{K_p}(c, d)$ simply parses $d$ to obtain $x, x'$ and verifies if $c = g^x h^{x'}$.

This scheme is perfectly hiding since $h^{x'}$ is a random value and thus c reveals no information about $x$. On the other hand, the scheme is computationally binding. Note that if $g^x h^{x'} = g^y h^{y'}$, then $log_g(h) = (x-y)/(x'-y')$. In other words, an adversary winning the binding game can be used to compute the discrete logarithm.

## 2.8 Elliptic Curve Cryptography

### 2.8.1 Elliptic Curves

For a prime $p > 3$, an elliptic curve $E(\mathbb{F}_p)$ over the field $\mathbb{F}_p$[1] consists of a set of points $(x, y)$ with $x,\ y \in \mathbb{F}_p$ which satisfy the equation $y^2 = x^3 + ax + b$ where the discriminant $4a^3 + 27b^2 \neq 0$. $E(\mathbb{F}_p)$ constitutes an Abelian group under the point-addition [Kob95] operation with the point infinity as the identity of the group. The order of this group is denoted by $\#E(\mathbb{F}_p)$. The ECC domain parameters are represented by $(p, \mathbb{F}_p, a, b, P, q)$ where $P \in E(\mathbb{F}_p)$ has prime order $q$ such that $q$ divides $\#E(\mathbb{F}_p)$.

Now we briefly describe DL-related problems over EC. Let $\mathbb{G}$ be a cyclic group $\mathbb{G}$ which is a subgroup of the points generated by $P \in E(\mathbb{F}_p)$ of order $q$.

**Definition 2.6 (EC Discrete Logarithm (EC-DL) Problem).** *Given a pair of $\mathbb{G}$ elements $(P, aP)$ for $a \in \mathbb{Z}_q^*$, find $a$. If this problem is hard, we say the EC-DL assumption holds in $\mathbb{G}$.*

**Definition 2.7 (EC Computational Diffie-Hellman (EC-CDH) Problem).** *Given a triple $(P, aP, bP)$ for $a, b \in \mathbb{Z}_q^*$, compute $abP$. If this problem is hard, we say the EC-CDH assumption holds in $\mathbb{G}$.*

**Definition 2.8 (EC Decisional Diffie-Hellman (EC-DDH) Problem).** *Given a quadruple $(P, aP, bP, cP)$ for $a, b, c \in \mathbb{Z}_q^*$, decide whether $c = ab$. If this problem is hard, we say the EC-DDH assumption holds in $\mathbb{G}$.*

**Definition 2.9 (EC Gap Diffie-Hellman (EC-GDH) Problem).** *Given a triple $(P, aP, bP)$ for $a, b \in \mathbb{Z}_q^*$, find $abP$ with the help of a EC-DDH oracle (which answers whether a given quadruple is a EC-DDH quadruple or not). If this problem is hard, we say the EC-GDH assumption holds in $\mathbb{G}$.*

### 2.8.2 Bilinear Maps

Bilinear maps play an important role in the pairing-based protocols. Let $\mathbb{G}_1$ be a cyclic additive group of order $q$ with a generator $P$ and $\mathbb{G}_2$ a cyclic group of the same order $q$. A *bilinear map* is a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfying $\hat{e}(aP, bQ)) = \hat{e}(P, Q)^{ab}$

[1] Some elliptic curves are defined over extension fields $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$, where $m$ is a positive exponent.

and non-degeneracy, $\hat{e}(P, P) \neq 1$ for all $a, b \in \mathbb{Z}_q^*$ and $P, Q \in \mathbb{G}_1$. Modified Weil [BF01] and Tate [FMR99] pairings on supersingular elliptic curves are examples of such bilinear maps.

**Definition 2.10 (Bilinear Diffie-Hellman (BDH) Problem).** *Given $(P, aP, bP, cP)$ for some $a, b, c \in \mathbb{Z}_p^*$, compute $v \in \mathbb{G}_2$ such that $v = \hat{e}(P, P)^{abc}$. If this problem is hard, we say the BDH assumption holds.*

### 2.8.3 BLS Signature Scheme

Boneh, et al. [BLS01] proposed a short signature scheme that works in a GDH group $\mathbb{G}$ of order $q$ and a generator $A$. In brief, the scheme operates as follows:

- KeyGen. Pick random $x \in \mathbb{Z}_q^*$ and compute $B = xA$. $x$ is the private key and $B$ is the corresponding public key.
- Sig. To sign a message $m \in \{0,1\}^*$, compute $\sigma = xH_1(m)$, where $H_1$ is a hash function that maps binary strings onto points in $\mathbb{G}^*$. Output $\sigma$ as the signature on $m$.
- Ver. Given $B, m, \sigma$, verify whether $(A, B, H_1(m), \sigma)$ is a valid Diffie-Hellman tuple using the bilinear map operations.

## 2.9 Identity-based Cryptography

The idea of identity-based (or ID-based) cryptography was first introduced by Shamir [Sha84]. ID-based cryptography allows two parties to securely communicate with each other just by the knowledge of each other' identities. This is in contrast to a standard PKI-based solutions, where the communicating parties need to obtain each other's certificate prior to communication. The first concrete ID-based cryptosystem was proposed by Boneh and Franklin [BF01].

An ID-based cryptosystem is managed by a trusted authority called a private key generator (PKG). Let $x$ be PKG's private key and $B$ be the corresponding public key, such that $B = xA$. Also denote $H$ as a cryptographic hash function s.t. $H : \mathbb{G}_2 \rightarrow \{0,1\}^n$ for some $n$.

**Setup.** A user with identity $id$ obtains from PKG a signed secret value $T$, which is the BLS signature on $id$ with PKG's private key $x$, i.e., $T = xH_1(id)$.

**ID-based Encryption (IBE).** This was proposed in [BF01]. We present a basic version of the scheme. The extended version is proven secure against chosen-ciphertext and ID attack in ROM.

- Enc. To encrypt a message $m$ under the public key $id$, compute $Q_{id} = H_1(id)$, choose a random $r \in \mathbb{Z}_q^*$ and output the ciphertext $c = (U, V) = (rA, m \oplus H(e(Q_{id}, B)^r))$.
- Dec. To decrypt the ciphertext $c$ using the secret $T$, compute $m = H(e(T, U)) \oplus V$ and output $m$ as the plaintext.

**ID-based Signatures.** This signature scheme was proposed in [CC03] and is proven secure against existential forgery on adaptively chosen message and ID attack in ROM.

- Sig. To sign a message $m$ under secret key $T$, generate a random $k \in \mathbb{Z}_q$, calculate $U = kH_1(id)$ and $Y = (k + H(m, U))T$, and output $(id, m, U, Y)$.
- Ver. To verify the signature $(id, m, U, Y)$ using the public keys $id, B$, check if $\hat{e}(Y, A) = \hat{e}(U + H(m, U)H_1(id), B)$.

# Part I

# Two-Party Setting

# Chapter 3

# Secure Device Pairing Using a Visual Channel

*In this chapter, we show how to implement secure device pairing between two devices based on a visual channel. McCune, et al. [MPR05] proposed that one device displays the hash of its public key in the form of a barcode, and the other device reads it using a camera. Mutual authentication requires switching the roles of the devices and repeating the above process in the reverse direction. We show how strong mutual authentication can be achieved even with a unidirectional visual channel, without having to switch device roles. By adopting recently proposed improved pairing protocols, we propose how visual channel authentication can be used even on devices that have very limited displaying capabilities, such as a single blinking LED.*

## 3.1 Introduction

The popularity of short-range wireless technologies like Bluetooth and Wireless Local Area Networking (WLAN) based on the IEEE 802.11 family of protocols is experiencing enormous growth. Newer technologies like Wireless Universal Serial Bus[1] are around the corner and promise to be as popular. This rise in popularity implies that an ever increasing proportion of the users of devices supporting short-range wireless communication are not

[1] `http://www.usb.org/developer/wusb`

technically savvy. Such users need very simple and intuitive methods for setting up their devices. Since wireless communication is easier to eavesdrop on and easier to manipulate, a common set up task is to initialize secure communication. In this chapter, we will use the term *pairing* to refer to this operation.[2]

Consequently, both security researchers and practitioners have been looking for intuitive techniques for ordinary users to be able to securely pair their devices. Although the primary impetus comes from the need to secure short-range wireless communication, the issue of intuitive security initialization is more generally applicable whenever ordinary users need to set up secure communication without the help of expert administrators or trusted third parties.

The pairing problem is to enable two devices, which share no prior context with each other, to agree upon a security association that they can use to protect their subsequent communication. Secure pairing must be resistant to a man-in-the-middle adversary who tries to impersonate one or both of these devices in the process. The adversary is assumed to be capable of listening to or modifying messages on the communication channel between the devices. One approach to secure pairing is to use an additional physically authenticatable channel, called an out-of-band (OOB) channel which is governed by humans, i.e., by the users operating these devices. The adversary is assumed to be incapable of modifying messages on the OOB channel, although it can listen to them.

There has been a significant amount of prior work on building secure pairing protocols using OOB channels [SA99, BSSW02, G$^+$02, Han02]. They consider different types of OOB channels including physical connections, infrared, etc. Recently, McCune, et al. proposed a scheme called "Seeing-is-Believing" (SiB), where the OOB channel is implemented as a visual channel. The SiB visual channel consists of a two-dimensional barcode of [RG04], displayed by (or affixed to) a device $A$, that represents security-relevant information unique to $A$. A user can point another camera-equipped device $B$ at the barcode so that $B$ can read the barcode visually, and use this information to set up an authenticated channel to $A$. If both devices are camera-equipped, they can mutually authenticate each other. "Authentication" in this case is based on demonstrative identification [BSSW02] rather than with respect to a claimed name.

[2]The term *pairing* was introduced in the context of Bluetooth devices. Other roughly synonymous terms include "bonding," and "imprinting".

**Our Contributions:** In this chapter of the thesis, we propose several improvements and extensions to the SiB system. Our contributions are as follows:

1. We show how strong mutual authentication can be achieved using just a unidirectional visual channel. This results in two improvements:
   (a) strong authentication becomes possible in situations where SiB could only achieve a weaker property termed as "presence".
   (b) execution time for mutual authentication decreases significantly and usability improves.
2. By adopting a recently proposed improved pairing protocol [LAN05], we show how visual channel authentication can be used even on devices that have very limited displaying capabilities, all the way down to a device whose display consists of a cheap single flashing light-source, such as a single light-emitting diode (LED).
3. We also propose a *video-based* codec which may help improve the speed of secure pairing in devices with constrained displays, as well as may lead to applications other than secure device pairing.

**Organization.** The rest of the chapter is organized as follows. First, we start with a brief description of SiB in Section 3.2. In Section 3.3 we describe an alternative protocol that improves the presence guarantee provided by SiB to full-fledged mutual authentication. Then, in Section 3.4, we show how visual channel authentication can be done even in highly constrained environments. We discuss the applicability and relevance of our improvements and extensions in Section 3.5.

## 3.2 Seeing-is-Believing

Several researchers have proposed the idea of encoding service or device discovery information in the form of barcodes so that they can be read using camera phones [RG04, CiM05, Woo05, MSSU04]. The idea of encoding cryptographic secret material into barcodes was first proposed by Hanna [Han02] as well as Gehrmann, et al. [G$^+$02], both of which also mention the use of asymmetric key cryptography in this context. The SiB paper [MPR05]

by McCune et al. was the first research paper to propose that the information encoded in the barcode could be a commitment to a public key.

In SiB, a device $A$ can authenticate to a device $B$, if $B$ is equipped with a camera. $A$'s commitment to its public key (such as a hash) is encoded in the form of a two-dimensional barcode of [RG04]. A typical barcode has dimensions approximately 2.5x2.5 $cm^2$ to allow recognition from a reasonable distance, and consists of a total of 83-bits of information (68-bits of data and 15-bits for forward error correction). If $A$ has a display, the public key can be ephemeral, and the barcode is shown on the display. Otherwise, $A$'s public key needs to be permanent and the barcode is put on a printed label affixed to the housing of $A$. Authentication is done by the user pointing $B$'s camera at $A$'s barcode. The basic unidirectional authentication process is depicted in Figure 3.1.

1. $A$ calculates $h_A$ as $h(K_A)$

   $A \longrightarrow B$ (visual channel): $h_A$

2. $A \longrightarrow B$ (insecure channel): $K_A$

   $B$ calculates $h'$ as $h(K_A)$ using the $K_A$ received. If $h'$ does not match the $h_A$ received in Step 1, $B$ aborts.

Figure 3.1: SiB unidirectional authentication protocol ($B$ authenticates $A$)

$K_A$ is $A$'s public key. $h()$ is a cryptographic hash function, which is resistant to second pre-image finding. $K_A$ can be long-lived, in which case the output of $h()$ must be sufficiently large, e.g., at least 80-bits. If $K_A$ is ephemeral, the output of $h()$ can be smaller, e.g. 48 bits [GMN04]. SiB could accommodate 68 bits of hash into a single two-dimensional barcode, but requires a good quality display due to the typical size of the barcode[3]. Mutual authentication requires the protocol of Figure 3.1 being run in each direction. This has two implications for SiB.

- First, mutual authentication is possible only if **both** devices are equipped with cameras. McCune, et al. state (Section 7 of [MPR05])

  > A display-only device ... is unable to strongly authenticate other devices using SiB ... [because it] cannot "see" them.

[3] SiB can encode the data into several barcodes displayed in sequence.

If a device $A$ is not equipped with a camera, it can only achieve a weaker property known as "presence," by including a secret key $K$ in the barcode. The camera-equipped device $B$ that reads the barcode can use $K$ to compute message authentication code (MAC) over the message it sends to $A$. If the MAC is correct, $A$ can conclude that it was sent by some device that was able to "see" its barcode, and thus was "present". Presence is a weaker security notion than authentication because $A$ has no means of knowing if $B$ is really the device that the user of $A$ intended to communicate with.

We summarize the types of authentication achievable using SiB for given combinations of device types in Table 3.1.

- Second, in order to run the protocol in each direction, the roles of the devices have to be switched so that first $A$'s camera can scan $B$'s display and then $B$'s camera can scan $A$'s display. Such switching of devices by users not only increases the execution time of the SiB process but also decreases usability. McCune, et al. report that the average SiB execution time in their user trials was 8 seconds, even though time required to recognize a barcode is just about one second [RG04].

| $Y$ has $\rightarrow$<br><br>$X$ has $\downarrow$ | Camera and display | Camera only | Display only | None |
|---|---|---|---|---|
| Camera and Display | $X \leftrightarrow Y$ | $X \leftrightarrow Y_s$ [a] | $X \leftarrow Y$<br>$X \xrightarrow{p} Y$ | $X \leftarrow Y_s$ |
| Camera only | $X_s \leftrightarrow Y$ | $X_s \leftrightarrow Y_s$ | $X \leftarrow Y$<br>$X \xrightarrow{p} Y$ | $X \leftarrow Y_s$ |
| Display only | $X \rightarrow Y$<br>$X \xleftarrow{p} Y$ | $X \rightarrow Y$<br>$X \xleftarrow{p} Y$ | none | none |
| None | $X_s \rightarrow Y$ | $X_s \rightarrow Y$ | none | none |

**Notation:**
[a] $P_s$: "Device $P$ needs a static barcode label affixed to it."
[a] $P \rightarrow Q$: "Device $P$ can strongly authenticate to device $Q$."
[a] $P \xrightarrow{p} Q$: "Device $P$ can demonstrate its presence to device $Q$."

Table 3.1: Types of authentication achievable using SiB for given device type combinations

These implications limit the applicability of SiB in various practical settings. Many

devices cannot have either cameras or high quality displays for different reasons. Commoditized devices like WLAN access points are extremely cost-sensitive and the likelihood of adding new hardware for the purpose of authentication is very small. Devices like Bluetooth headsets are typically too small to have displays or even to affix static barcode stickers.

To summarize, we identify the following drawbacks with the basic SiB scheme:

1. Mutual authentication is not possible unless both devices are equipped with cameras.
2. The overall execution time for mutual authentication is high, which impacts usability.
3. Applicability of SiB is limited in situations where one device has limited capabilities (e.g., small size, no camera, limited or no display at all).

In the rest of this chapter, we describe how we can address each of these drawbacks.

## 3.3 Seeing Better: Upgrading Presence to Authentication

In this section, we address the issue of mutual authentication. Recall that we identified two shortcomings of SiB in this respect. First, SiB can provide mutual authentication only if *both* devices are camera-equipped. Second, the processing time for mutual authentication is high.

We observe that both of these drawbacks stem from the fact that mutual authentication is done as two separate unidirectional authentication steps. Therefore, we propose to solve both problems by performing mutual authentication in a single step by having each of $A$ and $B$ compute a *common* checksum on public data, and compare their results via a unidirectional transfer using the visual channel. Let us call this protocol VIC, for "Visual authentication based on Integrity Checking." (See Figure 3.2.)

The security of the authentication of $A$ to $B$ in VIC depends on the attacker not being able to find two numbers $X1$ and $X2$ such that $h(K_A, X1) = h(X2, K_B)$. This implies that if the attacker can learn $K_B$ ahead of time, $h()$ needs to be collision-resistant. If $K_B$ is transient (or a nonce picked by $B$ is appended to $K_B$ in message 2 and in the calculation of $h_A$ and $h_B$), it is sufficient for $h()$ to be resistant against second pre-image finding, since the attacker can no longer use any pre-computed collisions. The security of the authentication of $B$ to $A$ depends, in addition, on the user correctly reporting the comparison result reported

1. $A \longrightarrow B$ (insecure channel): $K_A$

2. $A \longleftarrow B$ (insecure channel): $K_B$

   $A$ calculates $h_A$ as $h(K_A|K_B)$ and $B$ calculates $h_B$ as $h(K_A|K_B)$

3. $A \longrightarrow B$ (visual channel): $h_A$

   $B$ compares $h_A$ and $h_B$. If they match, $B$ accepts and continues. Otherwise $B$ rejects and aborts. In either case, $B$ indicates accept/reject to the user.

4. $A$ prompts user as to whether $B$ accepted or rejected. $A$ continues if the user answers affirmatively. Otherwise $A$ rejects.

Figure 3.2: VIC mutual authentication protocol

by $B$ back to $A$. (Note that the entities taking part in the protocol are always assumed to be honest.)

Because VIC needs only a unidirectional visual channel, it is now possible to achieve mutual authentication in the cases where SiB could only achieve presence. In addition, the execution time for mutual authentication and the user effort will be less since no device role switching is required anymore. Thus, VIC addresses the first two drawbacks of SiB identified in Section 3.2.

In Table 3.2, we summarize the types of authentication achievable using VIC for given combinations of device types. Notice that since the checksum is different for each instance of VIC, at least one device must have a display and that the static barcode labels cannot be used with VIC.

## 3.4 Seeing With Less: Visual Channel in Constrained Devices

Now we turn our attention to the third drawback of SiB. In this section, we show how to enable visual channel authentication on devices with very limited (or tiny) displays and in the minimal case, with extremely constrained displays consisting of only single light source (or LED). These extensions are made possible by using key agreement protocols that

| $Y$ has → <br> <br> $X$ has ↓ | Camera and display | Camera only | Display only | None |
|---|---|---|---|---|
| Camera and Display | X ↔ Y | X ↔ Y | X ↔ Y | none |
| Camera only | X ↔ Y | none | X ↔ Y | none |
| Display only | X ↔ Y | X ↔ Y | none | none |
| None | none | none | none | none |

**Notation:**
P ↔ Q: "Devices P and Q can strongly authenticate each other."

Table 3.2: Types of authentication achievable using VIC for given device type combinations.

use short authenticated strings or short authenticated integrity checksums. (We will build one such protocol in the next chapter)

We begin by recalling such protocols.

### 3.4.1 Authentication Using Short Integrity Checksums

The reason why SiB needs good displays is the high visual channel bandwidth required for the SiB protocol. Assuming that the attackers have access to today's state-of-the-art computing resources, the bandwidth needed is at least 48 bits in the case of ephemeral keys [GMN04], rising to 80 bits in the case of long-lived keys. These numbers can only increase over time.

Fortunately, there is a family of authentication protocols that has very low bandwidth requirements. The first protocols in this family, proposed by Gehrmann et al. in [G$^+$02, GMN04], were aimed at using the human user as the authentication channel; hence the name "Manual authentication (MANA)". Several subsequent variations on the same theme have been reported [Hoe04, Vau05, LAN05]. We apply the variation[4] called "MA-3" [LAN05] to get VICsh (VIC with short checksum) as shown in Figure 3.3.

$K_A, K_B$ are as in the case of SiB. $h()$ represents a commitment scheme and $hs()$ is a mixing function with a short $n$-bit output (e.g., $n = 15 \ldots 20$) such that a change in any input bit will, with high probability, result in a change in the output. In practice, $hs()$ can

[4] Although in the current prototype, we implemented the protocol by [LAN05], it could be replaced with our encryption-based authenticated key agreement protocol that we will describe in the chapter.

1. $A$ chooses a long random bit string $R_A$ and calculates $h_A$ as $h(R_A)$.

   $A \longrightarrow B$ (insecure channel): $h_A, K_A$

2. $B$ chooses its own long random bit string $R_B$

   $A \longleftarrow B$ (insecure channel): $R_B, K_B$

3. $A \longrightarrow B$ (insecure channel): $R_A$

   $B$ now computes $h'_A$ as $h(R_A)$ and compares it with the $h_A$ received in message 1. If they do not match, $B$ aborts. Otherwise $B$ continues.

4. $A$ calculates $hs_A$ as $hs(R_A, R_B, K_A, K_B)$ and $B$ calculates $hs_B$ as $hs(R_A, R_B, K_A, K_B)$

   $A \longrightarrow B$ (visual channel): $hs_A$

   $B$ compares $hs_A$ and $hs_B$. If they match, $B$ accepts and continues. Otherwise $B$ rejects and aborts. In either case, $B$ indicates accept/reject to the user.

5. $A$ prompts user as to whether $B$ accepted or rejected. $A$ continues if the user answers affirmatively. Otherwise $A$ rejects.

Figure 3.3: VICsh mutual authentication protocol based on short integrity checksum

be the output of a cryptographic hash function truncated to $n$ bits. Refer to [LAN05] for formal description of the requirements on $h()$ and $hs()$, and their instantiations, as well as for the proofs of security of the protocol. Informally, the security of the protocol depends on the following:

- neither party reveals the value of its random bit string ($R_A$ or $R_B$ respectively) until the other party commits to its own random bit string, and
- each party knows that the public data ($K_A$ and $K_B$) used in the computation of the check-value ($hs_A$ or $hs_B$) is known to it before it reveals its random bit string.

Suppose the man-in-the-middle attacker has a public key $K_M$. To fool device $A$ into accepting $K_M$ as $B$'s public key, the attacker needs to ensure that $hs_A = hs(R_A, X, K_A, K_M)$ and $hs_B = hs(Y, R_B, Z, K_B)$ are equal. The attacker can choose $K_M$, $X$, $Y$ and $Z$, but he must make his choices before knowing $R_A$ or $R_B$. Therefore, whatever his strategy for choosing the values, the chance of success is $x = 2^{-n}$. Similarly, the probability of the attacker fooling device $B$ into accepting $K_M$ as $A$'s public key is also $x$. More importantly, this probability does not depend on the computational capabilities of the attacker, as long as $h()$ is secure.

### 3.4.2 Trimming Down the Display

Armed with the variation of VIC described above, we are now ready to investigate visual channel authentication on devices with very limited displays. Recall that our motivation is to support visual channel authentication on various commercial devices, such as wireless access points, Bluetooth headsets, etc. These devices typically have only the most limited form of a display consisting of a single bi-state light source, such as a single light-emitting diode (LED). In this section, we describe each aspect of the realization of single LED based visual channel authentication.

**Transmission.** We use frequency modulation to encode the data being transmitted (see Figure 3.4). The sender turns the light-source on and off repeatedly. The data is encoded in the time interval between each successive "on" or "off" event: a long gap represents a '1' and a short gap represents a '0'. Since the channel is unidirectional, the transmitter cannot know when the receiver starts reception. Therefore, the transmitter keeps repeating the sequence until either the user approves the key agreement, or a timeout occurs.

The camera phones of today are limited to a frame rate of about 10 video frames/second, and as we are receiving the bits with frequency modulation without synchronization, we are bound by the Nyquist-Shannon sampling theorem (sampling rate = 2 × bandwidth for no loss of information) [Nyq28]. This limits the transfer speed with this algorithm to around 5 bits/second.

**Reception.** The receiver processing is analogous: simplified, each received video frame is compressed into one value per frame (the sum of all the pixel values)[5], and the first-order difference between consecutive values (i.e., the derivative) is compared against a relative threshold based on maximum observed variation in the pixel sum. If the derivative is steep enough and in the right direction (alternating between positive and negative) a transition in lighting is registered. The time between two consecutive changes indicates the transfer of either a '1' or a '0' bit as depicted in Figure 3.4.

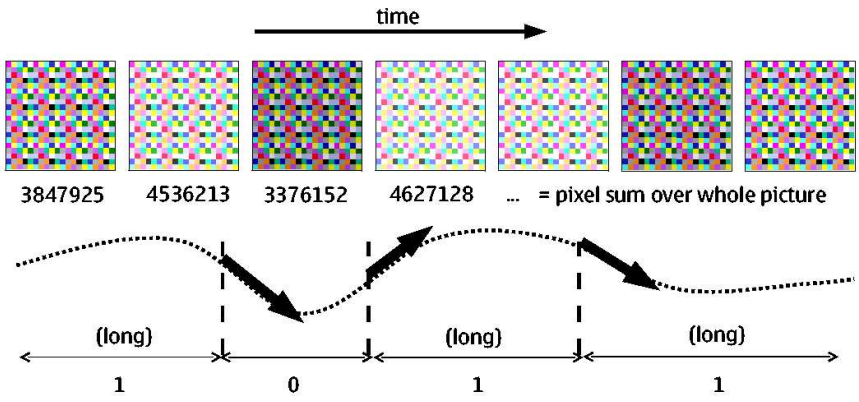



Figure 3.4: Data transmission via a single light-source visual channel

**Trading Efficiency with Security.** We designed two mechanisms that allow the possibility of a parameterizable trade-off between execution time and the level of security.

[5] The fact that the video frame is collapsed into one value per frame also shows the feasibility of using a sensitive light sensor combined with an analog-to-digital converter as a cheaper form of receiving device – with no change to the algorithms described here. We have left the implementation of such a receiver as future work.

First, the data being transmitted via the visual channel, i.e., the integrity checksum, is known to the receiver in advance. We use this simple observation to reduce execution time. Recall that the sender repeats the $n$-bit string a number of times. The receiver proceeds in the following way: reception may start at any bit position, and the receiver records until the $n$-bit tail of the received bit-string matches against any of the rotated versions of the expected $n$-bit string. Therefore, the receiver accepts at most $n$ possible matches for the transmitted value. For example, if the transmitted string is '1011', the receiver accepts if it receives any of the strings '1011', '0111', '1110', '1101'.

Second, rather than doing error correction, we tolerate (or simply accept) a certain number of errors in the $n$-bit transmission. With $k$ accepted errors, the number of possible matches, based on a binomial distribution of errors, is $\sum_{i=0\ldots k} \binom{n}{i}$.

Using these mechanisms the probability that the receiver will accept a random string as valid will increase from the original value of $p = \frac{1}{2^n}$. Accounting for both modifications we can estimate an upper bound to

$$p = n \, \frac{\sum_{i=0}^{k} \binom{n}{i}}{2^n}$$

The given bound allows us to get an idea of the degree of loss of security. If e.g. $k = 3$ bits are allowed to be wrong in an $n = 24$ bit sequence, $p$ is 0.0064, whereas if only 1 bit error is allowed, $p$ is 0.00004.[6]

For personal use, e.g., when a user wants to pair his workstation with his own wireless access point, an attack success probability of 0.00004 is acceptable. In other situations where, say, every day thousands of pairings are done with a device located in a public space, the attack success probability needs to be lower.

There are several ways to trade off security and execution time. The attack success probability $p$ can be decreased by:

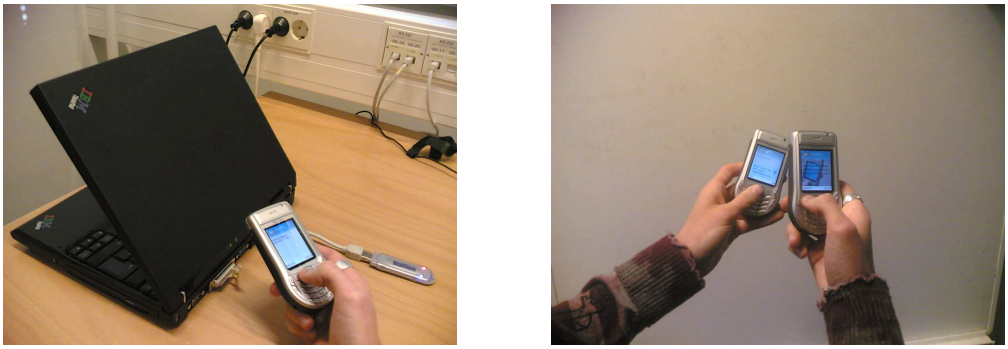
- increasing the length of the checksum $n$,
- reducing the number of acceptable errors $k$,

[6] In the pairing protocol case, the attack scenario is limited by the fact the the visual channel is authenticated, and the attacker is assumed to only operate in-band. In a more general case, where somebody might be feeding random visual data, the receiver also needs to check the signal history if the match is done later than after the first $n$ received bits (with $k$ errors). The history should give an indication that there is a repetition of the intended sequence (with possible errors) – if this is not the case the receiver is subject to a "visual attack" and accidentally found a match in a big sample of random input.

- reducing the number of possible rotations that are acceptable as matches (say only every fourth)
- adding an external end marker to the protocol (e.g., the light-source staying "on" for 0.5 seconds) to indicate when it starts to repeat the checksum string, bringing the attack success probability down to $\frac{\sum_{i=0}^{k}\binom{n}{i}}{2^n}$.

Applying one or several of these measures will result in changed lower and upper "bounds" for the execution time.

**Implementation and Timings.** We have developed a proof-of-concept implementation where a single blinking LED (connected to the parallel port of a PC) sends a signal that is received by a camera phone. Figures 3.5(a) and 3.5(b) illustrate our two demonstrator implementations. In 3.5(a), a Bluetooth pairing is established between a Symbian 8.0 camera phone and a Linux laptop with an LED (illustrating, e.g., a wireless access point). In 3.5(b), two phones are paired using the display of one phone as the bi-state light.

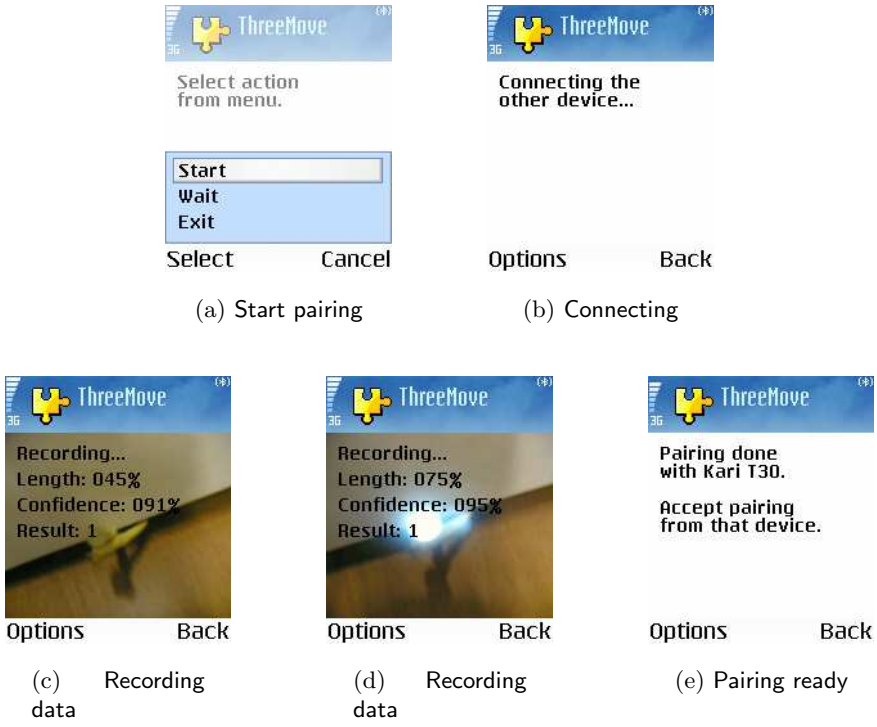


(a) Pairing phone and laptop (b) Pairing two phones

Figure 3.5: Scenarios for the proof-of-concept implementation

Our algorithm makes bit reception quite tolerant. The data can be received at a distance of several tens of centimeters, the implementation is agnostic to camera focus problems and tolerates a fair bit of camera shaking, turning, etc. The real-time progress of the matching is indicated at runtime on the handset screen by displaying two parameters: percentage of the string successfully received so far and a related confidence level.

Figure 3.6 gives a more detailed description of the user interface of our Symbian implementation during pairing with the laptop. In Figure 3.6(a), the user starts the pairing from a menu.[7] In Figure 3.6(b), the phone scans the Bluetooth neighborhood and finds the laptop. In Figures 3.6(c) and 3.6(d), the phone starts recording with its camera and the user positions the phone so that the blinking of the LED is shown in the viewfinder. The recording status is updated in the viewfinder in real-time. In 3.6(e), the pairing is complete for the phone once the correct checksum has been received and accepted. The success is reported to the user, who is instructed to accept the pairing at the access point to achieve mutual authentication.




(a) Start pairing (b) Connecting

(c) Recording data (d) Recording data (e) Pairing ready

Figure 3.6: Screen-shots from the Symbian implementation

With our setup, a 24-bit checksum signaled (1 error accepted) with the laptop is received and matched by the camera phone. The execution times for a positive indication

[7]The pairing must be initiated also from the laptop side. The rationale for this is explained in Section 3.5.2.

(match) is typically in the range of 5 to 8 seconds. The increased execution time is the price we pay for achieving visual channel authentication with devices that can not afford a full display. As mentioned, we consider these parameters acceptable for ordinary home use. A more secure version (32-bit checksum with 1 error) ranges from 8 to about 15 seconds.

### 3.4.3 Extending the Bandwidth on Better Displays

As we saw in Section 3.4.2, using VICsh with a single light source, and limiting the attack success probability to $2^{-20}$, the execution time cannot be smaller than about 5 seconds.

A natural question is whether any speedup in the execution time is possible if there were multiple light sources or in other words, a better display. In this section, we describe the design and analysis of a new *video codec* that can be used to set up a visual channel between a device with a small display and a device with a video camera. Our motivation was to investigate two different questions: whether the video codec can significantly improve the transfer time of a short checksum (15-20 bits), so that it can be used to reduce the execution time of secure pairing, and whether the video codec can enable applications other than secure pairing. In the remaining section, we discuss that even with straight-forward and naive techniques, such a video codec can be designed and that it performs reasonably efficiently.

**Encoding Process.** The idea of the encoding process is to represent the bits of data to be transmitted after encoding it for error correction into slots (rectangles) of black and white colors (say, 'black' to encode bit '0' and 'white' to encode bit '1') displayed in the form of animated frames at a certain rate. The number of slots that can be displayed in one frame depends upon the size of the display on the device and the video capturing ability of the decoding device, and the number of such frames depends upon the amount of data to be transmitted (e.g., 20 bits plus some error correcting bits in case of the pairing application) and the display rate. Following are the three main constituents of the encoding procedure:

*1. Beacon Slot:* To allow the decoding device to be able to capture all the frames, the display rate $R_d$ at the encoder should always be smaller than the capture rate $R_c$ at the decoding device. However, since $R_c > R_d$, the decoding device captures more frames than displayed by the encoding device, some of which get repeated a number of times. In order for the decoder to be able to identify and discard these repeated frames, we devote one

slot in each frame (say, at the top left corner) for a beacon frame, which always blinks, i.e, its color always changes from black to white, white to black, and so on. If the decoding device detects that the color of the beacon slot in the current frame is the same as the color of the beacon slot in the previous frame, it can safely discard the current frame. We observe through experiments that $R_d = 10$ frames/second is a reasonable display rate for most camera phones for which $R_c = 15$ frames/second.

*2. Blinking Corners:* Moreover, to facilitate the decoding device in detecting the screen of the encoding device in the captured frames, we use a small number of always-blinking pixels at the corners of the displayed frames at the encoding device. This simple technique allows the decoder to efficiently identify the exact location of the screen, as we illustrate in the decoding process part below.

*3. Marker Frames:* In Section 3.4.2, we exploited the fact that in the case of secure device pairing, the receiving device knows what string to expect via the visual channel. But since we want this video codec to be potentially usable in other applications, we cannot make this assumption. Therefore, since the decoding device may start recording at any bit position in the data string, we need to transmit the data frames at least twice, separated by a small number of marker frames.

**Decoding Process.** The decoding process involves capturing the video frames displayed by the transmitter, and using these frames to first detect the location of the screen, and then to read the data bits. We describe each of these procedures separately as follows:

*1. Locating the Screen:* We use the blinking corners (as described previously) in the display to locate the screen. The algorithm is very simple: on input of a certain number $t$ of consecutive frames, denoted by $F_1, \cdots, F_t$, compute a "Sum-of-Differences" frame $SD$, such that $SD = \sum_{i=2}^{n} |F_i - F_{i-1}|$, and scale $SD$ to pixel values 0 to 255, to obtain an image $F$. Note that the $|F_i - F_{i-1}|$ denotes the image corresponding to the absolute difference between the pixel values of $F_i$ and $F_{i-1}$, and adding two images means adding their corresponding pixel values. Notice that the image $F$ brightens the always changing or blinking pixels values (such as the ones corresponding to the corner pixels and the beacon slot) and at the same time darkens the ones which hardly change. See Figure 3.7(b) for an example $F$ image. Now, to further brighten the smaller regions corresponding to the corner pixels and to darken the other bigger bright regions (such as the one corresponding to the beacon slot), we use a standard tool in image processing called *convolution product*.

(a) A Few Captured Frames

(b) Sum-of-Differences Image

(c) Final Convoluted Image

Figure 3.7: An example of various images in the decoding process (with 4 slots per frame)

Figure 3.7(c) shows the convoluted image that has only the corner pixels bright. Once the convoluted image is obtained, it is easy to retrieve the corner pixels by using thresholding technique (e.g., in Figure 3.7(c), all pixels with values greater than 210 correspond to the corner pixels) and thus to locate the screen of the encoding device. Through experiments, we notice that the above algorithm performs quite robustly to detect the screen location if $t = 10$, i.e., if it is given 10 frames as input. The algorithm is also robust to rotation of the screen.

*2. Reading the Data:* Whenever two corresponding data slots in two consecutive frames have the same color, i.e., both black or both white, we obtain a black or white slots, respectively. However, when they have different colors, we obtain grey slots. The decoder first determines the color of each slot by looking at the distribution of pixel values in the slot and using simple thresholding technique. For example, if maximum number of pixels have values in range $(0, 85)$, the color is black, if they have values in range $(190, 255)$, the color is white, otherwise the color is grey.

Now, if the color of the beacon slot $B_i$ in current frame is the same as the color of the beacon slot $B_{i-1}$ in the previous frame, the current frame can be safely discarded. If the color of (non-beacon) slot $S_i$ is black, output data bit as 0; if it is white, output 1; otherwise if the color is grey, output the complement of the output of $S_{i-1}$.

**Error Correction.** Since we aim for applications besides pairing, we also need to use a robust error correction scheme for the video channel. Currently we use Reed-Solomon (RS) forward error correcting codes [RS60]. Reed-Solomon is one of the strongest error correcting codes known today, and applies very neatly in our scenario. Firstly, we have observed through experiments that we get errors in bursts (for example, errors in all the slots of one whole frame). Since, the RS codes operate on and corrects errors in symbols of a certain number of bits, they are well-suited to our codec. Secondly, RS codes are capable of correcting errors both in cases of erasures (such errors occur when it is known which symbols are corrupted) and non-erasures. In our codec, we get errors of both types. Erasures occur when it is very difficult to determine the exact color of a particular slot using the method of thresholding as described in the previously (for example, when the maximum number of pixel values are distributed around the boundaries of the thresholds for black and grey or grey and white). Non-erasures occur when we get errors, but we can't predict their locations.

With the $(k, n)$ RS error correction with $m$-bit symbols, where $n = 2^m - 1$, if there are $e$ erasures and $s$ non-erasure symbols in the received data, the RS code is capable of correcting them as long as $e + 2s \leq n - k$. For example, to send out 20-bits of data in the pairing application, we can use $(8, 4)$ RS codes (which is shortened from RS code $(31, 27)$), which corrects $e$ erasures and $s$ non-erasures in 5-bit symbols if $e + 2s \leq 4$.

**Implementation and Timings.** We have implemented our preliminary video codec prototype using Python Imaging Library[8] on Linux. In the current implementation, our decoding algorithm is given as input the video frames captured from a camera phone. Here, we report on some timing results based on the initial testing that we have done with this Python codec.

To send out 20-bits of original data (or 40-bits of encoded data) with $(8, 4)$ Reed-Solomon codes, as described in before, with 10 frames/second display rate, it takes around 3 seconds, when the display size is capable of displaying only 4 slots a frame, and almost 1 second when the display consists of 8 slots per frame. The screen location detection algorithm takes $2 - 3$ seconds with 10 frames on input, and the decoding and correcting of the data takes almost a second. Overall, it takes approximately $5 - 7$ seconds for the whole process.

[8] http://www.pythonware.com/products/pil/

These timing results are only preliminary. We anticipate the performance to improve when the python implementation is ported to a native C++ implementation on the Symbian platform. Yet, it is not clear if the execution time for transferring a short integrity checksum can be significantly reduced.

## 3.5 Discussion

In this section we discuss the applicability of our results, examine practical use cases, discuss related issues like performance, device discovery, and usability and briefly mention other related work.

### 3.5.1 Comparison of Different Protocols

Table 3.3 summarizes our recommendations on how mutual authentication can be achieved with different device type combinations. If both devices have camera and display, mutual authentication can be achieved either using SiB or VIC. SiB can be used with camera-only devices which can have static barcodes affixed to them. The case of two display-only devices is out of scope for this paper, and the basic MANA techniques which require the user to visually compare two short strings [G+02, GMN04] can be used. In all the other cases, VIC could be the best choice since it provides mutual authentication and potentially better usability.

| $Y$ has → <br> $X$ has ↓ | Camera and display | Camera only | Display only |
|---|---|---|---|
| Camera and Display | SiB/VIC | VIC | VIC |
| Camera only | VIC | SiB[a] | VIC |
| Display only | VIC | VIC | MANA |

[a] Both devices need static barcode labels affixed to them.

Table 3.3: Recommended protocol to achieve mutual authentication for given device type combinations

Table 3.4 summarizes when to use the two different flavours of VIC: If either one of the devices has a full display, then plain VIC as described in Section 3.3 can be used.

Otherwise VIC combined with MA-3 (which we called VICsh) can be used. Table 3.4 also summarizes the execution time measurements for the two cases. The execution times for the constrained display case or for the limited display is substantially longer than in full display case. Despite this, we stress that this case is extremely relevant, since not all devices have full displays to support the display of barcodes. Commodity devices like access points are very cost sensitive and it is highly unlikely that full displays are added to such devices. In addition, devices like headsets are so small that adding full displays is not possible. Also, note that the timings for constrained display case are geared for home usage scenarios.

| Display type | Recorder type | Protocol | Execution time |
|---|---|---|---|
| Full display | Still camera | VIC | 1 second[a] |
| Limited display | Video camera | VICsh | 5-7 seconds[b] |
| Constrained display | Video camera[c] | VICsh | 5-8 seconds[d] |

[a]Symbian OS implementation on Nokia 6600 [MPR05]
[b]Python implementation on PC
[c]Can also be a light sensor
[d]Symbian OS implementation on Nokia 6630

Table 3.4: Applicability of different flavors of VIC

Since the bandwidth requirement for VICsh protocol is low, this protocol could be used in scenarios where it is not possible to reach the bandwidth required by the VIC protocol. One example of such a scenario is a WLAN access point that is mounted high up on the wall or ceiling. It is not possible to read the barcode affixed to such an access point with the current camera phones, but it might be possible to read the "blinking" of the access point if the light source is powerful enough.

The preliminary timing results for transferring short strings using the video codec described in Section 3.4.3 are more or less comparable to the timing results for the single light-source approach described in Section 3.4.2. The former approach is more robust because of the forward error correction. The latter approach is somewhat cheaper. For ordinary uses of secure pairing, the single light-source approach may be more suitable even when the available display is slightly better than a single light-source. However, the video codec is a useful service for device discovery applications (as we discuss next) where several hundred bits of information need to be transferred via the visual channel, and there is no other communication channel between the two devices. For example, a small section

of the television display may be used to transmit the address of a web page relating to the television program in progress. Note that using a sequence of barcodes to encode this information is not a viable option since it would require the user to capture the barcode, notice when the barcode changes and ensure that each barcode is recorded.

### 3.5.2 Device Discovery Strategies

Previous proposals on security initialization using out-of-band methods [SA99, BSSW02] have argued that one of the main benefits of using an out-of-band channel for security initialization is the fact that device discovery is part of the OOB message exchange. For example in the approach proposed by Balfanz et al. [BSSW02] the devices exchange complete addresses over infrared, and thus no in-band device discovery is needed. In SiB approach, the device discovery is done manually (because current phones can not display big enough bar codes to contain both the address and the hash of a public key), but the authors state that the optimal solution would be to encode both the address and the public key hash to the bar code.

We argue that in many scenarios an in-band device discovery is actually needed before the OOB message exchange. The increasing number of different OOB channels (such as infrared, camera and full display, camera and single LED etc.) results in situations where the user might not always know which OOB to use with the two particular devices at hand. For example a user wanting to pair a camera phone (camera, display, no infrared) with a laptop (infared, display, no camera) might be confused about the different OOB possibilities. It should not be the user's burden to figure out which OOB to use (and how), but instead an in-band device discovery should take place and the best mutually supported OOB channel should be negotiated in-band and the user should be guided to use this OOB. Negotiations must be protected against bidding-down attacks in the usual manner, by having the parties exchange authenticated confirmations of the negotiation messages once key establishment is completed (as is done with the "Finished" message in TLS[DA99]). As long as the chosen authentication mechanism can not be broken in real-time, attempts to bid-down will be detected by this check.

In order to conveniently discover the desired device in-band, the user must put one of the devices into a temporary special discoverable mode so that the user does not have to select the correct device from a long list of (probably meaningless) device names. We call

this action *user conditioning*. From the user's point of view this action can be performed, e.g., by pressing a button on the device or by selecting a menu option.

Not all bearers support in-band discovery without manual device selection. Likewise, pure out-of-band discovery is not always feasible with constrained OOB channels. In these cases, the constrained OOB can be used to improve the usability of the in-band discovery process. A device can, e.g., send the last 10 bits of its address over OOB. At the same time the other device can scan and automatically discard devices whose address does not match these 10 bits. With high probability the correct device can be selected automatically and the user does not have to be presented a list of device names.

### 3.5.3 Usability Considerations

The security of VIC and VICsh relies on the user answering affirmatively in the last step (in Figures 3.2 and 3.3). If device $B$ rejects the key agreement and indicates failure to the user, but the user inadvertently answers affirmatively in the last step, device $A$ would conclude that the key agreement was authenticated even though $B$ does not. One way to mitigate the impact of this failure is as follows. $A$ picks a secret $k$ and sends it via the visual channel, along with the checksum. If $B$ accepts the key agreement, it can use the resulting secure channel to prove knowledge of $k$. $A$ will accept the key agreement only if the user accepts in the last step, *as well as* a proof of knowledge $k$ is received via the secure channel. On the downside, the additional check increases the amount of data transferred over the visual channel, thereby increasing the execution time. Moreover, the additional check is effective only if the attacker can not snoop on the visual channel.

Another way to reduce the likelihood of accidental (or out of habit) confirmation is to use a specific confirmation button only for the purpose of secure device pairing. The downside is the cost of adding such a button.

Whether this accidental confirmation is a real concern can only be determined by extensive usability testing. To date, none of the research papers dealing with the problem of secure device pairing have reported substantial *comparative* usability testing. The only exception is [BDG+04], which presents some usability analysis of their approach of using infrared as an OOB channel. Given the level of recent interest in this area which has resulted in several pairing approaches, a comprehensive comparative usability testing will be a very valuable research contribution. We are addressing this in our current work.

### 3.5.4 Denial-of-Service

Another concern is the possibility of a denial-of-service attack. An attacker can disrupt a pairing attempt between two devices by simultaneously initiating pairing with one or both of the same devices. Accidental simultaneous pairing is likely to be very rare because of the user conditioning described in Section 3.5.2. Thus, if a device detects multiple pairing attempts, the best strategy may be to ask the user to try again later, rather than ask the user to choose the correct device. Moreover, sending part of the device identifier via the visual channel, as described in Section 3.5.2, will help in picking the correct device in case of multiple parallel device pairing attempts. Note that in wireless networks, elaborate attempts to protect the pairing protocol against malicious attempts of denial-of-service are not cost effective because an attacker can always mount denial-of-service by simply disrupting the radio channel.

### 3.5.5 Other Related Work

Recently Goodrich, et al. [GSS+06], proposed a pairing mechanism making use of audio as the OOB channel. Their idea is to encode the public key hash value into an auditorially-robust, grammatically-correct sentence, which is displayed on one device and read out on the other using a voice synthesizer. The user then manually compares the two versions of the sentence in order to authenticate the public key. However, this scheme also suffers from the same problems as does SiB: namely, mutual authentication is either not possible (i.e, when one of the devices does not have an audio output and a display), or could be quite inefficient and taxing on the users. Fortunately, they can also use the MANA family of authentication protocols similar to our proposal in Section 3.4.1.

# Chapter 4

# Authenticated Key Agreement Using Short Authenticated Strings

*In this chapter, we present a new efficient encryption-based protocol for authenticated key agreement between two devices. The protocol is based on short (e.g., 20-bit long) manually authenticated channels between the devices. Our protocol is a useful alternative to the prior Diffie-Hellman-based protocols, especially in settings where the two devices have different computational powers, e.g. a PC and a cell-phone, a cell-phone and a headset, etc.*

## 4.1 Introduction

Last year Vaudenay introduced a notion of a message authentication protocol based on short authenticated strings (SAS) [Vau05]. Such protocol allows authenticating messages of arbitrary sizes (over an insecure channel) making use of an auxiliary or an OOB channel (as we called it in the previous chapter) which can authenticate short, e.g. 20-bit long, messages. It is assumed that an adversary has complete control over the insecure channel, i.e. it can eavesdrop, delay, drop, replay, inject and/or modifying messages. The only restriction on the special SAS channel is that the adversary cannot modify or inject messages on it, but it can eavesdrop on them, or drop, delay, or replay them.

A SAS-MCA protocol allows cross authentication of messages of arbitrary length

between two devices with an access to this auxiliary SAS channel. The primary application of SAS-MCA protocols is to enable SAS-based authenticated key agreement (SAS-AKA) between devices with no reliance on key pre-distribution or a public-key infrastructure. Such SAS-AKA protocols can be used in practice, for example in secure device pairing, i.e., to create a secure connection between two devices connected with a short-range communication medium, such as Bluetooth or WiFi, for which one can implement a SAS channel assuming some manual supervision from the device's user.

**Prior Work on SAS-MCA and SAS-AKA Protocols.** The shorter the message that needs to be authenticated on this auxiliary or OOB channel, the more user friendly and error-resilient the application becomes. This motivates cryptographic research whose goal is to provide provable security guarantees for SAS-MCA and SAS-AKA protocols assuming as little bandwidth on the SAS channel as possible. A straight-forward solution for a SAS-MCA protocol (as we already saw in chapter 3) was suggested by Balfanz, et al. [BSSW02]: Devices A and B exchange the messages $m_A$, $m_B$ over the insecure channel, and the corresponding hashes $H(m_A)$ and $H(m_B)$ over the auxiliary authenticated channel. Although non-interactive, the protocol requires the hash function to be strongly collision-resistant and therefore the bandwidth needed on the authenticated channel is quite high, at least 160-bits.[1] The protocols in Gehrmann et al. [GMN04] were aimed at using the human user as the authentication channel (and hence the name "manual authentication (MANA)"), and reduce the bandwidth requirement on the authentication channel to $16-20$-bits. However, these protocols required a stronger assumption on the SAS channel, i.e., the adversary is assumed to be incapable of delaying or replaying the SAS messages.

In [Vau05], Vaudenay presented the first message authentication scheme secure assuming the SAS channel, with an upper bound on the attack probability of $2^{-k}$ for a $k$-bit SAS channel. We include the unidirectional ($P_i \rightarrow P_j$) message authentication protocol, V-MA, presented in [Vau05], in Figure 4.1. By running two copies of this protocol, in $P_i \rightarrow P_j$ and $P_j \rightarrow P_i$ directions, and piggybacking the common flows, a SAS-MCA protocol is obtained. This protocol is based on any commitment scheme with certain non-malleability properties (see Section 4.2), and requires 4 communication rounds over the insecure channel.

[1] Pasini and Vaudenay [PV06] proposed a different non-interactive protocol that requires only a weakly collision resistant hash function, and reduces the bandwidth requirement to 80-bits.

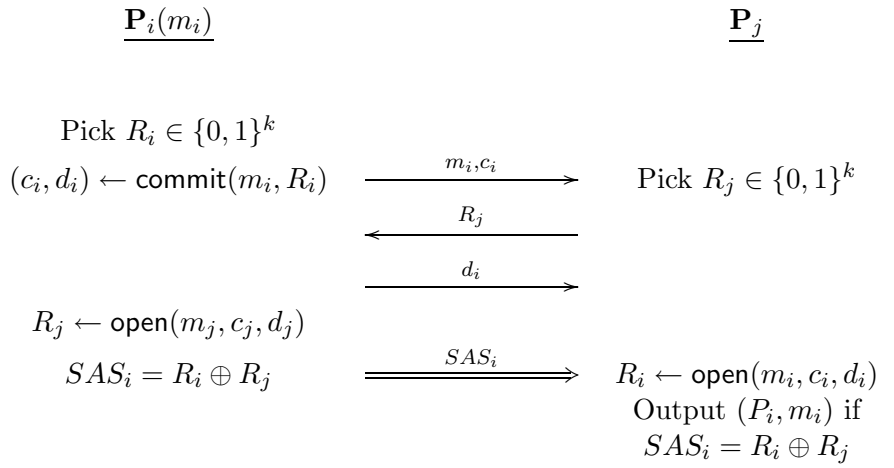



Figure 4.1: V-MA : unidirectional authentication ($P_i$ to $P_j$) based on generic commitments [Vau05]

In the follow-up work, Laur, Asokan, and Nyberg [LAN05] and Pasini and Vaudenay [PVar] independently gave three-round SAS-MCA protocols, using commitments with non-malleability-like properties and universal hash functions.[2] The two protocols have subtle differences, but both make just a few hash operations if the commitment scheme is implemented assuming the Random Oracle Model for the hash function.

Pasini and Vaudenay [PVar] argue that one can construct a 3-round SAS-based key agreement protocol SAS-AKA, from any 3-round SAS-based message cross-authentication protocol, SAS-MCA, and any 2-round key agreement scheme (KA) secure over authenticated links, e.g. a Diffie-Hellman or encryption-based KA scheme. The idea is to run the KA protocol over an insecure channel and authenticate all protocol messages using SAS-MCA as a sub-protocol. This is a standard "protocol compilation" methodology, similar for example to the Canetti-Krawczyk method for transforming KA protocols secure assuming authenticated links to AKA protocols secure over standard links using any message authentication scheme as a sub-protocol [CK01]. Such "protocol compilation" results in secure SAS-AKA protocols when applied to both standard Diffie-Hellman or Encryption-based KA protocols. However, the security proof of this compilation given in [PVar] works only when applied to KA protocols which do not share state between sessions. This excludes common key-agreement protocols which sacrifice perfect forward-secrecy by re-using the same secret key

[2] Recall that we employed the former protocol to efficiently implement a visual OOB channel in Chapter 3.

material across sessions, e.g. the same secret exponent in the Diffie-Hellman case, or the same private/public key pair in the case of the encryption-based key agreement. This is especially important for an encryption-based key agreement using RSA, where picking a fresh public/private key pair per session would be costly. We remark that extending the general composition theorem of [PVar] to key-agreement schemes that share state between sessions appears difficult, esp. while avoiding completing a 3-round MCA sub-protocol per each communication round in the KA protocol. (See Appendix B for further explanations.)

**Our Contributions.** We present a direct construction of a 3-round encryption-based SAS-AKA protocol. The protocol uses a CCA-secure encryption scheme, and a generic commitment scheme with the same non-malleability properties required also in the previous SAS-MCA protocols [Vau05, LAN05, PVar]. Our encryption-based SAS-AKA protocol is also a 3-round message SAS-MCA protocol, but since it uses public-key encryption, it is less efficient than the 3-round SAS-MCA protocols mentioned above. The protocol is very similar to the original 3-round uni-directional SAS-based message authentication protocol V-MA, see Figure 4.1, but it imposes on it a protocol flow of an encryption-based Key Agreement protocol: The initiator's first message $m_i$ includes her public key, and the responder sends his random challenge $R_j$ encrypted, together with his message $m_j$, under the initiator's public key.

In the Random Oracle Model, where the commitment scheme can be implemented with a single hash, the resulting SAS-AKA protocol without perfect forward secrecy involves a single public key encryption for the responder and a decryption for the initiator. If the user cares about perfect forward secrecy, the initiator has to also generate, although off-line, a public/private key pair. In other words, the costs of our protocol are essentially the same as the costs of the (unauthenticated) encryption-based key agreement protocol. As we mentioned above, the results of [PVar] also imply 3-round SAS-AKA protocols (both encryption-based and Diffie-Hellman based), but only perfect forward-secure ones. Their costs are also determined by the costs of the (unauthenticated) key agreement protocols they start with. We stress that a fast encryption-based key agreement protocol secure under public key re-use (although without perfect forward secrecy) is an attractive alternative to Diffie-Hellman SAS-AKA protocols, especially in the case of devices with asymmetric computational powers, e.g. a PC and a cell phone, a cell phone and an earset speaker, etc.

Interestingly, our construction of a SAS-AKA protocol does not build in a modular

way on a SAS-MCA protocol, because if we strip-away the encryption layer the remaining protocol is not a SAS-MCA. This forced us to make a direct security argument for it, instead of following a general "protocol compilation" approach as in [PVar]. This choice, and some other subtle differences in the modeling of the SAS-AKA security (e.g. we treat sessions id's explicitly), resulted in a tighter security bound we were able to show for this SAS-AKA protocol, compared to what results of [PVar] imply for the (perfect forward secure) 3-round SAS-AKA protocols. Our better exact security bound means that we can show the same level of security for our SAS-AKA protocol for SAS channels which have $\log_2(n)$ fewer bandwidth, where $n$ is the total number of entities executing the protocol.

**Organization.** We describe our communication and adversarial model in Section 4.2. We present our SAS-MCA and SAS-AKA protocols in Sections 4.4 and 4.5, and we discuss the implications of our result for SAS channel bandwidth requirements in Section 4.6.

## 4.2 Communication and Adversarial Model

### 4.2.1 Network/Communication Setting

We consider a network consisting of $n$ players $P_1, \cdots, P_n$. Each ordered pair of players $(P_i, P_j)$ is connected by two unidirectional point-to-point communication channels: (1) an insecure channel, e.g. a Bluetooth or a WiFi channel, over which an adversary has complete control by eavesdropping, delaying, dropping, replaying, and/or modifying messages, and (2) a low-bandwidth out-of-band authenticated (but not secret) channel, referred to as a *SAS channel* from here on, which preserves the integrity of messages and also provides source and target authentication. In other words, on the insecure channel, an adversary can behave arbitrarily, but it is *not* allowed to modify (or inject) messages sent on the SAS channel (which we'll call *SAS messages* for short), although it can still read them, as well as delay, drop, or re-order them. Refer to Section 4.1 for examples of implementations which provide such SAS channels

To differentiate among messages corresponding to different sessions, as in any standard security protocols, we include session identifiers in each message sent on the insecure channel.[3] The session identifiers must satisfy a constraint that a single player never re-uses

[3] These session identifiers can be exchanged by the communicating parties in a "prologue", prior to starting the protocol. Such a prologue is needed (as in various protocols such as SSL, ssh, IKE, etc.) for the

the same session identifier twice. This is easy to enforce in practice: In an instance of a protocol between users $P_i$ and $P_j$, the players can first exchange random long-enough values $s_i$ and $s_j$ and set $s = (s_i, s_j)$. With high probability, the $s_i$ and $s_j$ values are locally unique and hence so is $s$.

### 4.2.2 SAS-MCA and its Security

A SAS-MCA protocol is a "cross-party" message authentication protocol, executed between two players $P_i$ and $P_j$ on a joint session id $s$, whose goal is for $P_i$ and $P_j$ to send authenticated messages to one another. The inputs of $P_i$ are a tuple $(ROLE_i, P_j, s, m_i)$ where $ROLE_i$ designates $P_i$ as either the initiator ("*init*") or a responder ("*resp*") in this protocol, $P_j$ identifies the communication partner for this protocol instances, which essentially identifies for $P_i$ a pair of SAS channels $(P_i \rightarrow P_j)$ and $(P_i \leftarrow P_j)$ with an entity with whom the application wants to communicate, $s$ is a session id, and $m_i$ is the message to be sent to $P_j$. The other player, $P_j$, is presumably concurrently started on inputs $(ROLE_j, P_i, s, m_j)$ on a matching $s$ and on $ROLE_j \neq ROLE_i$. The outputs of $P_i$ can be either a tuple $(P_j, \hat{m}_j, s)$ or a rejection. Similarly $P_j$ can either output $(P_i, \hat{m}_i, s)$ or reject. The protocol should satisfy the trivial completeness condition which says that if the two players are started on matching sessions (i.e. the identities $P_i$ and $P_j$, the session ids, and the roles in their inputs all match), and if the adversary delivers all their messages without modifying them, then both players output the messages actually sent by the other player, i.e. $\hat{m}_j = m_j$ and $\hat{m}_i = m_i$.

We model the security of a SAS-MCA protocol via a following game between the network and the adversary $\mathcal{A}$. With queries of the form $\mathsf{launch}(P_i, {ROLE_i}^{(s)}, P_j, s, {m_i}^{(s)})$, $\mathcal{A}$ can trigger any of the $n$ players $P_i$ to start the SAS-MCA protocol on the given inputs $({ROLE_i}^{(s)}, P_j, s, {m_i}^{(s)})$, subject to the only constraint that the same player $P_i$ is never started on same $s$ twice. We say that $\mathcal{A}$ succeeds in attacking the SAS-MCA protocol if any $P_i$ outputs $(P_j, \hat{m}_j, s)$ but the player $P_j$ was never started on inputs $(ROLE_j, P_i, \hat{m}_j, s)$, for some $ROLE_j$. In other words, the adversary wins if it never issues the command $\mathsf{launch}(P_j, {ROLE_j}^{(s)}, P_i, s, \hat{m}_j^{(s)})$, but $P_i$ nevertheless outputs $(P_j, \hat{m}_j, s)$. (Note that in particular the adversary must also issue some command of the form $\mathsf{launch}(P_i, \star, P_j, s, \star)$.)

parties to mutually agree upon a set of algorithms to use during the execution of the protocol. Note that in a SAS-based protocol, parties also need to agree upon the SAS channel (audio, visual, infra-red, etc.).

In the above SAS-MCA game, the adversary can launch multiple concurrent sessions among every pair of players. To make our security results concrete, we will consider an $(n, R, \bar{R})$-attacker $\mathcal{A}$ against the SAS-MCA protocol, which plays the above game with $n$ players in the network subject to the constraint that $\mathcal{A}$ is allowed to launch only $R$ sessions per player, and only $\bar{R}$ sessions between any *pair* of players.

### 4.2.3 SAS-AKA and its Security

A SAS-AKA protocol is an authenticated key exchange protocol executed by a pair of players $P_i, P_j$ on a common session id $s$. The inputs to the protocol for both parties are as in the SAS-MCA above, except that now there are no messages $m_i, m_j$. Player $P_i$ outputs either a rejection or a tuple $(P_j, s, K)$, where $K$ is a fresh, authenticated, and secret key which $P_i$ presumably shares with $P_j$, and which both players can use now for securing any subsequent communication between them. Similarly player $P_j$ either outputs $(P_i, s, K)$ for some $K$ or rejects. We call each instance of the SAS-AKA protocol running by a player a separate *session*, and we call two sessions, $(P_i, ROLE_i, P_j, s)$ and $(P_i', ROLE_i', P_j', s')$ *matching* if $P_i' = P_j$, $ROLE_i' \neq ROLE_i$, $P_j' = P_i$, and $s' = s$. In other words, only two sessions of the type $(P_i, init, P_j, s)$ and $(P_j, resp, P_i, s)$ are matching. The completeness property for a SAS-AKA protocol is that if two matching sessions execute and all their messages are delivered properly then both players accept and output the same key $K$.

We model security of the SAS-AKA protocol in a similar way as Canneti and Krawczyk model security of an authenticated key exchange (AKA) protocol running over a standard network [CK01], except that we will limit ourselves only to *security*, and leave out the explicit authentication requirement.[4] As in the case of the SAS-MCA security model above, we consider an $(n, R, \bar{R})$-attacker $\mathcal{A}$, which plays the following game with a network of $n$ players $P_1, ..., P_n$. The adversary issues commands of the form $\mathsf{launch}(P_i, {ROLE_i}^{(s)}, P_j, s)$, which triggers player $P_i$ to start the SAS-AKA protocol instance on these inputs. On any of the launched sessions, $\mathcal{A}$ can also issue a query of the form $\mathsf{reveal}(P_i, P_j, s)$, which gives him $P_i$'s output in the previously launched session matching this tuple (if such exists). The response to this query is either a rejection symbol, if $P_i$ either has not completed this session or rejected on it, or, if $P_i$ completed this session successfully and outputted some $(P_j, s, K)$

[4] However, since our SAS-AKA protocol is also a message authentication protocol, i.e. SAS-MCA protocol, the output key actually *is* explicitly authenticated in our protocol.

tuple, the corresponding session key $K$. Finally, on one of the tuples $(P_i, P_j, s)$ for which no reveal() query was issued, $\mathcal{A}$ issues a $\mathsf{Test}(P_i, P_j, s)$ query, and, if the session is successful, bit $b$ is chosen at random without adversary's knowledge, and if $b = 0$ then $\mathcal{A}$ gets the key $K$ output by $P_i$ on that session, or, if $b = 1$, $\mathcal{A}$ gets a random value of the same length. The adversary can then keep issuing the launch and reveal commands, except it cannot reveal the tested session or a session that matches it. Eventually $\mathcal{A}$ outputs a bit $\hat{b}$, and we say that $\mathcal{A}$ wins if $\hat{b} = b$. We say that an adversary has *advantage* $\epsilon$ in the SAS-AKA attack if the probability that $\hat{b} = b$ is at most $1/2 + \epsilon$.

### 4.2.4 Comparison with Other Security Notions for SAS-MCA and SAS-AKA

The security notion that we have described above is slightly different than the one considered in the previous proposals by Vaudenay et al [Vau05, PVar]. In both of these works the (multi-session) attack against a SAS-MCA protocol is defined as follows. An adversary launches multiple instances of the protocols for various players, w.l.o.g. including many instances involving the same pair of players $(P_i, P_j)$. The adversary is then said to succeed if it can make one of these players, say $P_i$, output a message $\hat{m}_j$ which was not an input on **any** of the launched instances of $P_j$ with $P_i$. This is a natural definition in their context, because they do not include session id's as inputs to the SAS-MCA protocol, and therefore if there are many instances of $P_i$, each of which is trying to communicate with $P_j$, and similarly many instances of $P_j$ communicating with $P_i$, there's no notion of an a-priori matching between these instances. In contrast, we use session id's as explicit inputs to each SAS-MCA instance, and under our security notion, an attacker succeeds if an instance of $P_i$ running on session $s$ outputs message $\hat{m}_j$ which was an input to the instance of $P_j$ which was launched on the **same** session id $s$.

Of course, session id's can be handled on an intermediary level between the SAS-MCA protocol and the applications which instantiate it and use its outputs. This level can attach the session id's to the messages sent on the underlying "session-less" SAS-MCA protocol like the protocol of Vaudenay et al, and then retrieve the session id's from the received messages and direct them to the application which a matching session id. In this way the security guarantees given by Vaudenay et al can be used to achieve the same security notion we define here. However, the fact that we handle session id's explicitly simplifies the

security arguments in the multi-session setting (for both SAS-MCA and SAS-AKA), which contributes to a better exact security analysis, and therefore to stronger exact security claims for the same parameters, and consequently to lower bandwidth requirements on SAS channels required for the same security level.

## 4.3 Preliminaries

Our SAS-MCA and SAS-AKA protocols utilize both a public-key encryption scheme and a commitment scheme (we defined these primitives in Chapters 2). The security notions for these schemes that our protocols rely upon, are described below.

### 4.3.1 OW-R-CCA-secure Public Key Encryption

In our protocols, we rely on a slightly weaker security property of *partial one-wayness* of encryption (under the CCA attack), denoted OW-R-CCA, which we define via the following game between the adversary and the challenger: The keys $(pk, sk)$ are chosen by KeyGen as above, the adversary $\mathcal{A}$ on $pk$ selects message $m$ and sends it to $\mathcal{C}$. $\mathcal{C}$ picks $R \in \{0,1\}^k$, and sends back the ciphertext $c = \mathsf{Enc}_{pk}(m, R)$. $\mathcal{A}$ responds with $\hat{R}$ and we say that $\mathcal{A}$ "wins" if $\hat{R} = R$. As in the above game, $\mathcal{A}$ is also given an adaptive access to a decryption oracle. An encryption scheme is said to be $(T, \epsilon)$-OW-R-CCA if no adversary running in time $T$ can win this game with a probability greater than $2^{-k}+\epsilon$. It is easy to see that if an encryption scheme is $(T, \epsilon)$-IND-CCA, i.e., indistinguishable under CCA (refer to Chapter 2 for this definition) then it is also $(T, \epsilon)$-OW-R-CCA.

### 4.3.2 OW-ExA-secure Commitment Scheme

We call a perfectly binding scheme $(T, \epsilon)$-OW-ExA, which stands for *hiding (in the sense of one-wayness) under the adaptive extraction attack*, if a $T$-bounded adversary $\mathcal{A}$ cannot win the hiding game with probability better than $2^{-k} + \epsilon$, even if the game is *modified* so that $\mathcal{A}$ additionally has access to an extraction oracle extract, which $\mathcal{A}$ can query on adaptively chosen inputs $(m', c')$, as long as $m'$ is different from tag $m$ used in the commitment challenge. The extraction oracle outputs $R'$ if and only if there exists $d'$ such that $\mathsf{open}(m', c', d') = R'$. Otherwise it outputs a special symbol $\perp$. (Note that if the commitment scheme is perfectly binding, this extraction oracle is well-defined.)

**OW-ExA-Commitment from CCA-secure Encryption.** $(T, \epsilon)$-OW-ExA commitment scheme can be created from any $(T, \epsilon)$-OW-R-CCA encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ (and hence in particular also from any $CCA$-secure encryption scheme). The $K_P$ is a public key $PK$ of the encryption scheme. $\mathsf{commit}_{PK}(m, R)$ picks a random string $s$ and outputs $c = \mathsf{Enc}^{(s)}(PK, (m, R))$ and $d = (R, s)$, where $\mathsf{Enc}^{(s)}(\cdot)$ denotes running the (randomized) encryption procedure using randomness specified in string $s$. Procedure $\mathsf{open}_{PK}(m, c, (R, s))$ outputs $r$ if $c = \mathsf{Enc}^{(s)}(PK, (m, R))$. Note that this commitment scheme is perfectly binding. This reduction of the OW-ExA attack to the OW-R-CCA attack is easy, because the decryption oracle in the OW-R-CCA game outputs $(m', R')$ on any $c' = \mathsf{Enc}(PK, (m', R'))$ s.t. $c' \neq c$, and so it can be used to simulate the extraction oracle for the OW-ExA attacker, which needs both $c'$ and $m'$ to output just the $R'$ part of the cleartext. Note that if $m' \neq m$ but $c' = c$ in the commitment extraction query $(m', c')$ then the reduction can output $\perp$ without consulting the OW-R-CCA decryption oracle, because ciphertext $c$ encrypts a unique message $(m, R)$, so $m' \neq m$ is an incorrect tag for commitment $c$. The final challenge the OW-ExA adversary must solve is exactly the same as the challenge of the OW-R-CCA adversary.

**OW-ExA-Commitment in ROM.** As observed in [Vau05], one can create a very fast and simple commitment scheme with a hash function $H : \{0,1\}^* \to \{0,1\}^{l_c}$ modeled as a random oracle, provided a long enough $l_c$. $\mathsf{commit}(m, R)$ picks $e \in \{0,1\}^{l_e}$ and returns $(c, d)$ such that $d = (R, e)$ and $c = H(m, d)$. $\mathsf{open}(m, c, (R, e))$ returns $R$ if $c = H(m, (R, e))$, and rejects otherwise. To handle this construction in a perfectly formal way we would need to extend the notion of OW-ExA-security stated above, because this simple ROM-based commitment scheme is actually not perfectly binding. However, for any adversary $\mathcal{A}$ running in time $T$ one can implement an extraction oracle which returns $R$ for any $(m, c)$ s.t. the probability that $\mathcal{A}$ ever opens any of these $c$'s to some other string $R' \neq R$, is bounded by $T^2 2^{-l_c}$, which is the probability of a collision in $T$ queries to the hash function. Given such extraction oracle, a $T$-limited adversary has only $2^{-k} + T2^{-k+l_e}$ probability of winning the hiding game. The proofs we present in this write-up all assume a perfectly binding (and OW-ExA-secure) commitment scheme, but they can be extended to imply security also for this ROM-based scheme which has the $T^2 2^{-l_c}$ "binding error": This binding error needs to be added to the error term in each of the theorems, but it's not a problem because $l_c$ can be adjusted to make this error quantity arbitrarily small for any $T$.

### 4.3.3 B-EqA-Secure Commitment Scheme

A commitment scheme is *perfectly hiding* if it is $(T, 0)$-*hiding* for any $T$. We call a perfectly hiding commitment scheme $(T, \epsilon)$-B-EqA, which stands for *binding under the equivocation attack*, if a $T$-bounded adversary $\mathcal{A}$ cannot win the binding game with probability better than $2^{-k} + \epsilon$ even if the game is modified so that $\mathcal{A}$ has access to oracles simcommit and equivocate: Oracle simcommit, on any query input $m'$, outputs a fake commitment $c'$, whose distribution is identical, for any $R$, to the distribution of real commitment $c$ output by commit$(m, R)$. The adversary is restricted from queries $m' = m$ to the simcommit oracle where $m$ is a part of the binding challenge $(m, c)$ which $\mathcal{A}$ sends to $\mathcal{C}$ in this game. Oracle equivocate, on input $(c, m', R')$, where $c$ was previously output by simcommit on $m'$, returns a decommitment value $d'$ such that open$(m', c', d') = R'$.

B-EqA-secure commitment schemes can be constructed from simulation-sound trapdoor commitments (see [Vau05]), and therefore, combined with the results of [MY04], they follow from any signature scheme, e.g. the Cramer-Shoup signature scheme [CS99] based on the strong RSA assumption.

*Remark:* In this thesis, we show security of the resulting SAS-MCA and SAS-AKA protocols under the assumptions that the commitment scheme is OW-ExA-secure and the encryption scheme is OW-R-CCA-secure. Because our protocol is based on the unidirectional MCA protocol V-MA proposed by Vaudenay, the security of our protocols can be alternatively based on OW-R-CCA-secure encryption and a commitment scheme which is *binding even in the presence of an equivocation oracle*, i.e. "B-EqA-secure". However, we omit any proofs and formal claims about security for our protocols based on this assumption from the current write-up.

## 4.4 Encryption-based SAS Message Authentication Protocol SAS-MCA

We present a 3-round encryption-based SAS-MCA protocol. To simplify its presentation and its security analysis, we will first present the SAS-MCA protocol called Enc-MCA for the case of only two parties running a single session, and prove its security against the $(2, 1, 1)$-attacker. (This simplified form of an attacker is called *one-shot adversary* by Vaudenay [Vau05].) We then proceed to the full multi-player and multi-session version of the

same protocol, denoted Enc-mMCA.

### 4.4.1 Two-Party/ Single-Session Setting

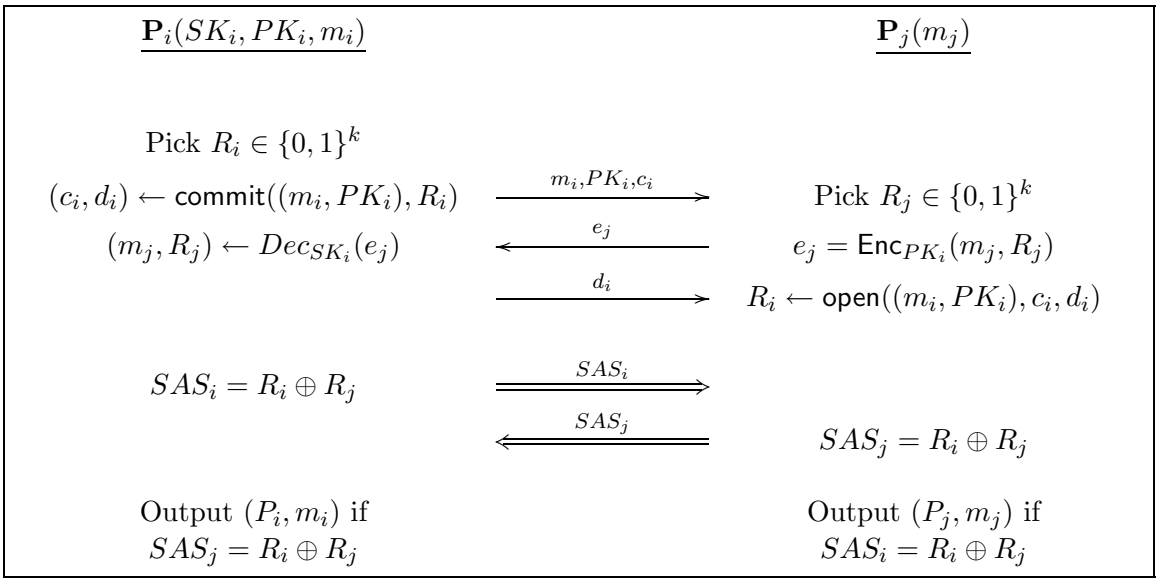



Figure 4.2: Enc-MCA: Encryption-based SAS-MCA (two-party, single-session setting)

The Enc-MCA protocol is depicted in Figure 4.2. It runs between the "initiator" $P_i$, who intends to authenticate a message $m_i$, and the "responder" $P_j$, who intends to authenticate a message $m_j$. $(SK_i, PK_i)$ denotes the (possibly permanent) private and public key pair for $P_i$. The protocol is based on the unidirectional message-authentication V-MA protocol of Vaudenay [Vau05], Figure 4.1. The only difference is that $P_i$ includes its public key $PK_i$ with its message $m_i$, and the responder $P_j$ sends its randomness $R_j$ *encrypted* under this key, together with its message $m_j$. In other words, the protocol begins by $P_i$ sending $m_i$ along with $PK_i$ and commitment $c_i$ to $(m_i, PK_i)$ using a random value $R_i \in \{0,1\}^k$. $P_j$ encrypts its message $m_j$ and a random value $R_j \in \{0,1\}^k$, and sends the encrypted value $e_j$ back to $P_i$. Next, $P_i$ decommits to the values committed in the first message by sending over the decommitment $d_i$. $P_i$ and $P_j$ then exchange over the SAS channel values $SAS_i = R_i \oplus R_j$, where $P_i$ obtains $R_j$ by decrypting $e_j$, and $SAS_j = R_i \oplus R_j$, where $P_j$ obtains $R_i$ by opening $m_i, PK_i, d_i$. The players accept if the $SAS$ values match, and reject otherwise.

**Theorem 1 (Security of *Encryption-based SAS-MCA (two-party, single-session setting)*).** *If a* $(2,1,1)$*-attacker against the* Enc-MCA *protocol bounded by a time* $T$ *wins the SAS-MCA game with a probability* $p$ *and if the commitment scheme is* $(T_C, \epsilon_C)$-OW-ExA *and the encryption scheme is* $(T_E, \epsilon_E)$- OW-R-CCA*, then* $p \leq 2^{-k+1} + max(2\epsilon_C, 2\epsilon_E)$ *or* $T \geq min(T_C, T_E) - \mu$*, for a small constant* $\mu$*.*[5]

*Proof.* We prove the above claim by contradiction, i.e, we show that if there exists an adversary $\mathcal{A}$ which can attack the proposed protocol in time $T < min(T_C, T_E) - \mu$ with probability $p$ better than $2^{-k+1} + max(2\epsilon_C, 2\epsilon_E)$, then there exists *either* a $T$-time adversary $\mathcal{B}_C$ which wins the hiding-on-extraction-attack game OW-ExA with a probability better than $2^{-k} + \epsilon_C$, *or* there exists a $T$-time adversary $\mathcal{B}_E$ which wins the OW-R-CCA game for the encryption scheme with probability better than $2^{-k} + \epsilon_E$.

Adversary $\mathcal{A}$ succeeds if he instantiates the two parties $P_i, P_j$ on some messages, $m_i, m_j$, respectively, and if either $P_i$ accepts and outputs some message $\hat{m}_j \neq m_j$ or $P_j$ accepts and outputs some message $\hat{m}_i \neq m_i$. Since $\mathcal{A}$ must get either party to complete the protocol to succeed, he must get both of them to trigger their SAS messages, and so any successful attack that $\mathcal{A}$ executes has the following form:

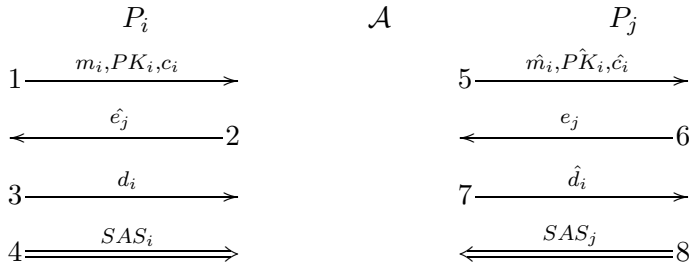



Figure 4.3: Adversarial Behavior in the Enc-MCA protocol

Moreover, the only way $\mathcal{A}$ can get either $P_i$ or $P_j$ to accept is if $SAS_i = R_i \oplus \hat{R}_j$ is equal to $SAS_j = \hat{R}_i \oplus R_j$. In other words, the values $R_i, R_j, \hat{R}_i, \hat{R}_j$ defined in the above interaction between $\mathcal{A}$ and $(P_i, P_j)$ must satisfy constraint $R_i \oplus R_j \oplus \hat{R}_i \oplus \hat{R}_j = 0$.

[5]The security bound in this Theorem is not optimal as it wastes one bit of the SAS bandwidth. However, by using a more involved security argument we *can* improve on theorem 1 and show that the probability of attack against the Enc-MCA protocol can be bounded by a value arbitrarily close to $2^{-k}$, under the same assumptions. We omit this improved analysis from this write-up, but we give an intuition for it in Appendix C.

Note, however, that $\mathcal{A}$ can control the *sequence* in which the above messages are interleaved, and he has a choice of the following three possible sequences:

$$\begin{aligned}
\text{Message interleaving pattern I}: &\quad (1 \prec 5 \prec 6 \prec 2 \prec 3 \prec 4 \prec 7 \prec 8) \\
\text{Message interleaving pattern II}: &\quad (1 \prec 5 \prec 6 \prec 7 \prec 8 \prec 2 \prec 3 \prec 4) \\
\text{Message interleaving pattern III}: &\quad (1 \prec 2 \prec 3 \prec 4 \prec 5 \prec 6 \prec 7 \prec 8)
\end{aligned}$$

For each of these three message interleaving patterns we'll have to consider two sub-cases, depending on whether the pair $(\hat{m}_i, \hat{PK}_i)$ that the adversary delivers to $P_j$ in message #5 (see Figure 4.3) is equal to $(m_i, PK_i)$ that player $P_i$ sends in message #1.

Let's denote the event that adversary succeeds in an attack as $\mathsf{AdvSc}$, the event that $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ *and* that the attack succeeds as $\mathsf{SM}$, the event that $(\hat{m}_i, \hat{PK}_i) \neq (m_i, PK_i)$ *and* that the attack succeeds as $\mathsf{NSM}$, and we'll use $\mathsf{Int}[1], \mathsf{Int}[2], \mathsf{Int}[3]$ to denote events when the adversary follows, respectively, the 1st, 2nd, or 3rd message interleaving pattern. We divide the six possible patterns which the successful attack must follow into the following two cases:

$$\mathsf{Case1} = (\mathsf{NSM} \vee (\mathsf{SM} \wedge \mathsf{Int}[2])) \quad \text{and} \quad \mathsf{Case2} = (\mathsf{SM} \wedge \mathsf{Int}[1]) \vee (\mathsf{SM} \wedge \mathsf{Int}[3]))$$

We show that if adversary succeeds in any of the above cases with probability $p$, then this implies an attack against one of the above security assumptions. We construct two reduction algorithms, $\mathcal{B}_C$ and $\mathcal{B}_E$, which attack respectively the commitment and the encryption scheme used in the Enc-MCA protocol. More specifically, $\mathcal{B}_C$ attacks the OW-ExA property of the commitment scheme, and $\mathcal{B}_E$ attacks the OW-R-CCA property of encryption. Both algorithms $\mathcal{B}_C$ and $\mathcal{B}_E$ will use the Enc-MCA attacker $\mathcal{A}$. The reductions $\mathcal{B}_C$ and $\mathcal{B}_E$ are successful depending on which of the above cases $\mathcal{A}$ takes. Namely, in event Case1 reduction $\mathcal{B}_C$ wins the OW-ExA commitment game, and in event Case2 reduction $\mathcal{B}_E$ wins the OW-R-CCA encryption game.[6]

Note that $\mathsf{AdvSc} = \mathsf{Case1} \vee \mathsf{Case2}$, and therefore, if $\Pr[\mathsf{AdvSc}] = p$ then either $\Pr[\mathsf{Case1}] \geq p/2$ or $\Pr[\mathsf{Case2}] \geq p/2$. Note that by assumption on $p$, we have that $p/2 > 2^{-k} + \epsilon_C$ *and* $p/2 > 2^{-k} + \epsilon_E$, and hence *either* $\mathcal{B}_C$ wins the OW-ExA commitment game with probability greater than $2^{-k} + \epsilon_C$ *or* $\mathcal{B}_E$ wins the OW-ExA commitment game with

[6] This is a sketch: In one case the correspondence between event Case1 for $\mathcal{A}$ and the winning of reduction $\mathcal{B}_C$ is more complex, but still the probability that $\mathcal{B}_C$ wins is the probability that Case1 happens.

probability greater than $2^{-k}+\epsilon_E$. Moreover, both $\mathcal{B}_C$ and $\mathcal{B}_E$ make only a few cryptographic operations in addition to running $\mathcal{A}$, so their running time will be $T+\mu$ for a small constant $\mu$, and hence, by assumption on $T$, their running times are below both $T_C$ and $T_E$. Since this contradicts either the $(T_C, \epsilon_C)$ OW-ExA-security of the commitment scheme or the $(T_E, \epsilon_E)$ OW-R-CCA-security of the encryption, the theorem will follow.

It remains for us to construct algorithms $\mathcal{B}_C$ and $\mathcal{B}_E$ with the properties claimed above. Algorithm $\mathcal{B}_C$, depending on the behavior of $\mathcal{A}$, executes one of the following sub-algorithms:

If $(\hat{m}_i, \hat{PK}_i) \neq (m_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern I, II, or III, then $\mathcal{B}_C$ executes sub-algorithms, respectively, $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$, and $\mathcal{B}_C[3]$.

If $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern II, $\mathcal{B}_C$ executes $\mathcal{B}_C[4]$.

Otherwise, i.e. if $\mathcal{A}$ sends $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ and $\mathcal{A}$ follows patterns I or III, $\mathcal{B}_C$ fails.

Similarly, based on the behavior of $\mathcal{A}$, algorithm $\mathcal{B}_E$ proceeds in one of the following ways:

If $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern I, $\mathcal{B}_E$ executes $\mathcal{B}_E[1]$.

If $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern III, $\mathcal{B}_E$ executes $\mathcal{B}_E[2]$.

Otherwise, i.e. if $\mathcal{A}$ sends $(\hat{m}_i, \hat{PK}_i) \neq (m_i, PK_i)$ or $\mathcal{A}$ follows interleaving pattern II, $\mathcal{B}_E$ fails.

We show algorithms $\mathcal{B}_C[1]$,$\mathcal{B}_C[2]$, $\mathcal{B}_C[3]$ and $\mathcal{B}_C[4]$ in Figures E.1, E.2, E.3, and E.4, respectively. Note that if $(\hat{m}_i, \hat{PK}_i) \neq (m_i, PK_i)$ then $\mathcal{A}$ essentially attacks the V-MA protocol of Vaudenay, because pair $(m_i, PK_i)$ in the Enc-MCA protocol plays a role of the message in the V-MA protocol, so this event in the Enc-MCA protocol is equivalent to $P_j$ accepting the wrong message in the V-MA protocol. Therefore, the three reduction (sub)algorithms $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$, and $\mathcal{B}_C[3]$, essentially perform the same attacks on the OW-ExA game of the commitment scheme as the corresponding three reductions given by Vaudenay for the V-MA protocol. The only difference is that our reductions put a layer of encryption on the messages sent by $P_j$, as is done in our protocol Enc-MCA. As in Vaudenay's reductions,

we extend the OW-ExA game so that the challenger, at the end of the game sends to the attacker the decommitment $d$ corresponding to the challenge commitment $c$. Since this happens after the attacker sends its $R$, the difficulty of the OW-ExA game remains the same. However, if the $\mathcal{B}_C$ reduction gets the decommitment $d$ from the OW-ExA challenger, the reduction can complete the view of the protocol to $\mathcal{A}$, which makes it easier (esp. in case of $\mathcal{B}_C[4]$) to compare the probability of $\mathcal{A}$'s success with the probability of success of $\mathcal{B}_C$.

For completeness, we show these three sub-cases of the reduction to an OW-ExA attack in Appendix E. The fourth case $\mathcal{B}_C[4]$ is also shown in Appendix E, Figure E.4. However, the procedure $\mathcal{B}_C[4]$ actually follows exactly the same protocol as procedure $\mathcal{B}_C[2]$, which means that in the case of the interleaving pattern II, reduction $\mathcal{B}_C$ proceeds in the same way regardless whether $\mathcal{A}$ modifies message $(m_i, PK_i)$ or not. By inspection of the figures, note that each of these sub-cases of the $\mathcal{B}_C$ reduction at first follows the same protocol with the OW-ExA challenger, and that $\mathcal{B}_C$ can decide which path to follow, namely whether to switch to sub-algorithm $\mathcal{B}_C$[1,2,4] or $\mathcal{B}_C[3]$, based on the first message it receives from $\mathcal{A}$. In this case, $\mathcal{B}_C$ switches to $\mathcal{B}_C[3]$ if $\mathcal{A}$ first sends message $\hat{e}_j$, and otherwise $\mathcal{B}_C$ follows $\mathcal{B}_C$[1,2,4]. Similarly, in the latter case, $\mathcal{B}_C$ switches to either $\mathcal{B}_C[1]$ or $\mathcal{B}_C$[2,4] based on $\mathcal{A}$'s next response. Therefore the four pictures represent not different algorithms $\mathcal{B}_C$[1-4] but just four sub-cases of a single algorithm $\mathcal{B}_C$.

By inspection of Figure E.1, note that $\mathcal{B}_C[1]$ wins in the OW-ExA game in the case of event $\mathsf{NSM} \wedge \mathsf{Int}[1]$. Note that an extraction of $\hat{c}_i$ is allowed because $(\hat{m}_i, \hat{PK}_i)$ is different from tag $(m_i, PK_i)$ used in commitment $c_i$. Similarly, by inspection of Figures E.2 and E.4, note that $\mathcal{B}_C[2]$ and $\mathcal{B}_C[4]$ win the OW-ExA game in the case of events $\mathsf{NSM} \wedge \mathsf{Int}[2]$ and $\mathsf{SM} \wedge \mathsf{Int}[2]$, respectively. Consequently, $\mathcal{B}_C$ wins in any of these cases as well. The case of $\mathcal{B}_C[3]$ is slightly different: Here the probability that $\mathcal{A}$ wins is actually at most $2^{-k}$ unconditionally, as long the commitment scheme is perfectly binding. Note that $\mathcal{B}_C[3]$ has the same $2^{-k}$ probability of winning in this case because it just returns a randomly chosen $R$ to the challenger. Since event Case1 implies one of these four cases, and we have that $\mathcal{B}_C$ wins in cases $(\mathsf{NSM} \wedge \mathsf{Int}[1]) \vee \mathsf{Int}[2]$, while in the remaining case $\mathsf{Int}[3] \wedge ((\hat{m}_i, \hat{PK}_i) \neq (m_i, PK_i))$ the probability that $\mathcal{B}_C$ is greater or equal to the probability that $\mathcal{A}$ wins (given that this case happens), it follows that the probability that $\mathcal{B}_C$ wins is at least the probability Pr[Case1], as required.

The construction of $\mathcal{B}_E[1]$ is depicted in Figure 4.4 ($\mathcal{B}_E[2]$ follows similarly). The construction works as follows. Receive the public key $PK$ of the challenger. Then, on

receiving $m_i, m_j$ from $\mathcal{A}$, pick $R_i \in \{0,1\}^k$, compute $(c_i, d_i) \leftarrow \mathsf{commit}((m_i, PK), R_i)$ and forward $(m_i, PK, c_i)$ to $\mathcal{A}$. Send $m_j$ to the challenger and forward the received ciphertext $e_j = \mathsf{Enc}_{PK}(m_j, R_j)$ (where $R_j$ is a random $k$-bit string picked by the challenger) to $\mathcal{A}$. When $\mathcal{A}$ sends $\hat{e_j} = \mathsf{Enc}_{PK}(\hat{m_j}, \hat{R_j})$, query it to the decryption oracle to obtain the plaintext $(\hat{m_j}, \hat{R_j})$. Note that since $\hat{m_j}$ differs from $m_j$, $\hat{e_j}$ must also differ from $e_j$, and therefore the query to the decryption oracle is allowed. If $\mathcal{A}$ wins, then $R_j$ must equal $\hat{R_j} \oplus \hat{R_i} \oplus R_i$, which $\mathcal{B}_E$ sends to the challenger to win the challenger game. The same holds in the case of the $\mathcal{B}_E[2]$ reduction.

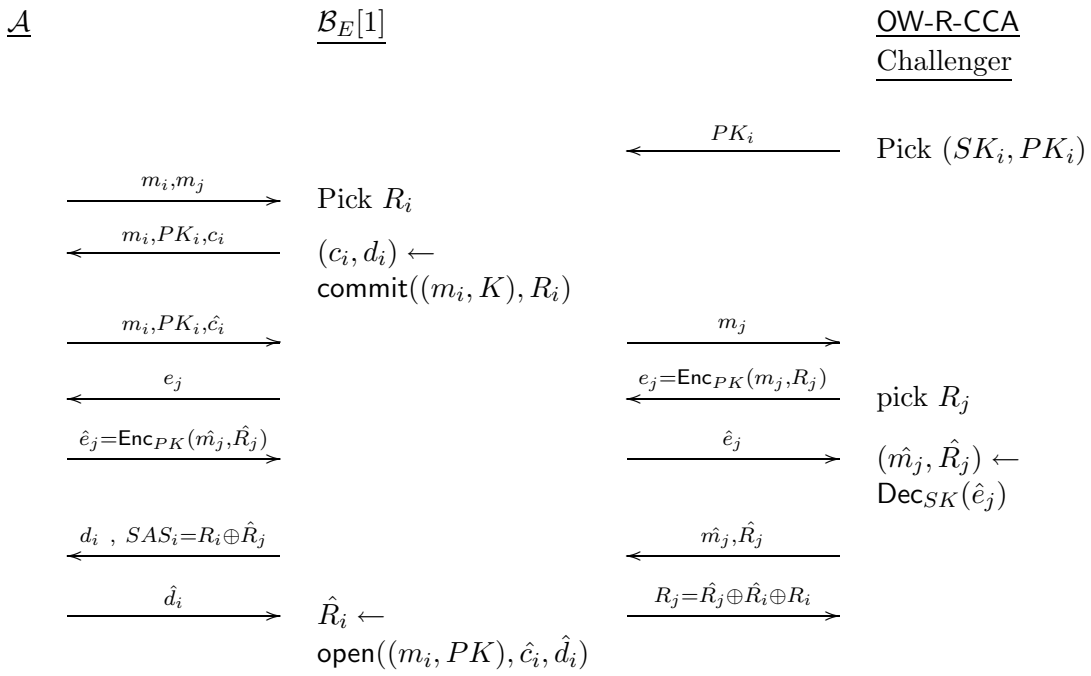



Figure 4.4: Construction of $\mathcal{B}_E[1]$ ($(m_i, PK_i) = (\hat{m_i}, \hat{PK_i})$, interleaving case I)

□

### 4.4.2 Multi-party/ Multi-Session Setting

The basic two-party message authentication protocol Enc-MCA can be extended to the multi-party/multi-session setting which is the real setting in which such protocols should operate. The Enc-mMCA protocol is depicted in Figure 4.5. The multi-party/multi-session protocol is the same as the two-party, single-session protocol of Figure 4.2, but the parties run the Enc-MCA protocol on *pairs* $(m, s)$, where $m$ is the original message to be

authenticated and $s$ is the session identifier. Moreover, all protocol messages (except the ones on the SAS channels) should include $s$ in the cleartext. Recall that a session identifier between parties $P_i$ and $P_j$ can be agreed upon by exchanging random long-enough values $s_i$ and $s_j$, respectively, and setting $s = (s_i, s_j)$, which guarantees, except of negligible probability, that $s$ is locally unique to both $P_i$ and $P_j$.

We do not want to include the session identifiers in the SAS messages because we want to keep the bandwidth on the SAS channel as low as possible. However, this means that there is no way for the two parties $P_i, P_j$ to differentiate among several SAS messages sent on *different* sessions between $P_i$ and $P_j$. Therefore, if many instances of the protocol are run concurrently by the same pair $P_i, P_j$, then $P_i$ will keep a set $SAS_{ji}$ of all the SAS messages sent by $P_j$ to $P_i$ within some time window, and $P_i$ will accept in session $s$ of the Enc-mMCA protocol if value ${R_i}^{(s)} \oplus {R_j}^{(s)}$ that it computes at the end of this protocol instance matches *some* value in $SAS_{ji}$. Similarly, $P_j$ accepts in session $s$ if the value ${R_i}^{(s)} \oplus {R_j}^{(s)}$ it computes at the end of this protocol instance matches some value in $SAS_{ij}$.

| $\mathbf{P}_i(SK_i, PK_i, m_i^{(s)}, s)$ | $\mathbf{P}_j(m_j^{(s)}, s)$ |
|---|---|
| Run Enc-MCA of Figure 4.2 as initiator on input $(SK_i, PK_i, (s, m_i^{(s)}))$ | Run Enc-MCA of Figure 4.2 as responder on input $((s, m_j^{(s)}))$ |

Figure 4.5: Enc-mMCA: Encryption-based SAS-MCA (multi-party, multi-session setting)

**Theorem 2 (Security of *Encryption-based SAS-MCA (multi-player, multi-session setting)*).** *If a $(n, R, \bar{R})$-attacker against the protocol bounded by a time $T$ wins with a probability $p$ and if the commitment scheme is $(T_C, \epsilon_C)$-OW-ExA and the encryption scheme is $(T_E, \epsilon_E)$-OW-R-CCA, then $p \leq nR\bar{R}2^{-k+1} + max(2nR\bar{R}\epsilon_C, 2nR\bar{R}\epsilon_E)$ or $T \geq min(T_C, T_E) - \mu$, for a small constant $\mu$.*

We show the proof in Appendix D.

## 4.5 Encryption-based SAS Authenticated Key Agreement Protocol SAS-AKA

We show the protocol in Figure 4.6 and call it Enc-AKA. The protocol is based on the SAS-MCA protocol of Figure 4.5. It runs on input of the session identifier, and on successful termination outputs the same key on both sides. $SAS_{ji}$ and $SAS_{ij}$ denote the set of all SAS messages transmitted from $P_j$ to $P_i$ and from $P_i$ to $P_j$, respectively.

| $\mathbf{P}_i(SK_i, PK_i, s)$ | $\mathbf{P}_j(s)$ |
|---|---|
| Run Enc-mMCA as initiator on input $(SK_i, PK_i, \perp, s)$ | Pick $K^{(s)} \in \{0,1\}^l$<br>Run Enc-mMCA as responder on input $(K^{(s)}, s)$ |

Figure 4.6: Enc-AKA: Encryption-based SAS-AKA

**Theorem 3 (Security of Encryption-based *SAS-AKA*).** *If a $(n, R, \bar{R})$-attacker bounded by time $T$ has advantage $\epsilon$ in a game against the* SAS-AKA *protocol, and if the commitment scheme is $(T_C, \epsilon_C)$-OW-ExA and the encryption scheme is $(T_E, \epsilon_E)$-OW-R-CCA, then $\epsilon \leq nR\bar{R}2^{-k+1} + max(2nR\bar{R}\epsilon_C, 2nR\bar{R}\epsilon_E)$ or $T \geq min(T_C, T_E) - \mu$, for a small constant $\mu$.*

*Proof. (Sketch)* We show that if there exists a $(n, R, \bar{R})$-adversary $\mathcal{A}$ which can attack the proposed protocol in time $T \geq min(T_C, T_E) - \mu$ with probability $p$ better than $1/2 + nR\bar{R}2^{-k+1} + max(2nR\bar{R}\epsilon_C, 2nR\bar{R}\epsilon_E)$, then there exists an adversary $\mathcal{B}_C$ which can win the hiding game in case of OW-ExA commitments with a probability better than $2^{-k} + \epsilon_C$ or there exists an adversary $\mathcal{B}_E$ which can win the OW-R-CCA challenger game of the encryption scheme with a probability significantly better than $2^{-k} + \epsilon_E$.

$\mathcal{A}$ succeeds if it can find a pair of players $P_i$ (initiator) and $P_j$ (responder) both running sessions with same id $s$, and can distinguish the session key computed by either of them, from random. We'll consider the case where $\mathcal{A}$ tests the initiator and the case when $\mathcal{A}$ tests the responder separately below.

Both reduction algorithms, $\mathcal{B}_C$ and $\mathcal{B}_E$ start by guessing some session initialized as $(P_i, init, P_j, s)$ (there are at most $nR/2$ of these). We'll call this a $(P_i, s)$ session, but this choice determines $P_j$. Both reductions also pick one session at random among all

sessions of the form $(P_j, resp, P_i, s')$, for the above $P_i, P_j$ pair (that's additional $\bar{R}$ guesses). Additionally, each reduction guesses whether it's $(P_i, s)$ or $(P_j, s')$ that will be tested. If $\mathcal{A}$ tests some other session than the one guessed by $\mathcal{B}_C$ or $\mathcal{B}_E$, either reduction outputs a random bit. Therefore, as in the reduction of MCA protocol security, Theorem 2, the success probability of this reduction deteriorates by a factor of $nR\bar{R}$. In either case (initiator or responder) considered below, the reduction considers two sub-cases, and if it guesses which sub-case it is prepared to handle, this results in additional factor of 2 in the security degradation, thus leading to the $p \leq 1/2 + 2nR\bar{R} * [2^{-k} + max(\epsilon_C, \epsilon_E)]$ bound on $p$.

1. We first argue that $\mathcal{A}$ cannot make the initiator $P_i$ accept a key $\hat{K^{(s)}}$ different from $K^{(s)}$ picked by $P_j$ on the session $s$. This is because the success of $\mathcal{A}$ in doing so is clearly equivalent to an attack against $P_j$ to $P_i$ direction of the Enc-mMCA protocol shown in Figure 4.5 and follows directly from the reductions $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$, $\mathcal{B}_C[3]$, $\mathcal{B}_C[4]$, $\mathcal{B}_E[1]$ and $\mathcal{B}_E[2]$ shown in the proof Theorem 2. Note that these reductions will also need to simulate the responses to the "reveal" queries issued by $\mathcal{A}$. In the first four reductions, our algorithm is able to perfectly simulate them by responding with the session keys that it simply picks itself or it obtains by following the protocol. While in the last two reductions, to answer the reveal queries corresponding to sessions of the initiator $P_i$, the reduction makes use of the CCA decryption oracle; for any other session, where $P_i$ is not an initiator, "revelation" of keys is done by following the protocol.

   From the above argument, it follows that $P_i$ must output the same key $K^{(s)}$ which was picked by $P_j$ on session $s$. If $\mathcal{A}$ now succeeds in distinguishing this key from random, we reduce it to an attacker $\mathcal{C}_E$ against the IND-CCA game of the encryption scheme, as shown in Figure 4.7. The simulation and "revelation" of keys of the sessions other than the "tested" one, other than the ones corresponding to $P_i$ and the ones where $P_i$ is not an initiator, are done by following the protocol. While to simulate and answer the "reveal" queries corresponding to sessions of the initiator $P_i$, the reduction makes use of the CCA decryption oracle.

2. Consider the case when $\mathcal{A}$ attacks the responder $P_j$ by succeeding in sending a public key $\hat{PK_i}$ different from $PK_i$. In this case, we reduce $\mathcal{A}$ to an attack $\mathcal{B}_C$ which executes sub-algorithms $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$ and $\mathcal{B}_C[3]$, based on the message interleaving patterns. This follows directly from the constructions $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$ and $\mathcal{B}_C[3]$, of the proof of the

Theorem 2. Note that on any session except the tested session, the reduction simply follows the protocol and is therefore able to respond to the "reveal" queries by $\mathcal{A}$ with the session keys that it outputs.

Now, consider the case when $\mathcal{A}$ attacks the responder $P_i$, but sets $PK_i = \hat{PK}_i$. In this case, we reduce $\mathcal{A}$ to a CCA attacker similarly as shown in Figure 4.7 and as we argued above for the case of $\mathcal{A}$ attacking the initiator.

□

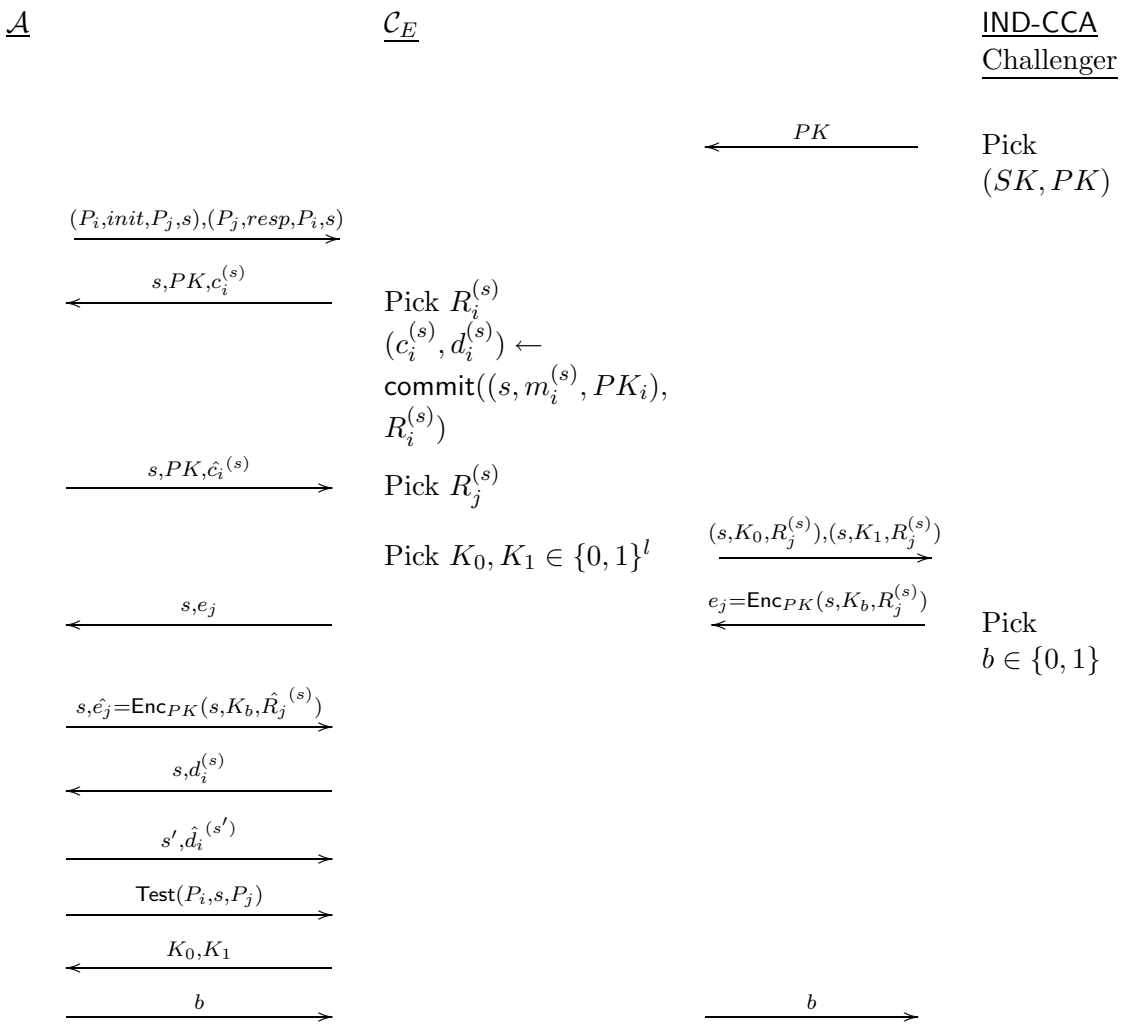



Figure 4.7: Construction of $\mathcal{C}_E$ from $\mathcal{A}$ for interleaving case I

## 4.6 Implications on the Bandwidth of the SAS Channel

From our security analysis of the SAS-MCA and SAS-AKA protocols it follows that the bandwidth $k$ needed on the SAS channel should be $log_2(2\underline{n}R\bar{R}/p)$, where $p$ is the desired upper bound on the attack probability. Since $\bar{R} \leq R$, we have $k \geq log_2(2nR^2/p)$. In contrast, based on the analysis presented in [Vau05] (Lemma 6), $k \geq log_2(\underline{n^2}R^2/4p)$. Therefore, for $n \geq 8$, we require $log_2(n) - 3$ bits fewer on the SAS channel than that in prior proposals. Using the same numbers for the parameters as in [Vau05], i.e., $n = 2^{20}, R = 2^{10}, p = 2^{-10}$, we need only 51-bits on the SAS channel, while [Vau05] needs 68-bits. Similarly, if we consider an attack against a *target verifier node*, i.e., when the attacked player is fixed, $k \geq log_2(2R\bar{R}/p) \geq log_2(2R^2/p)$ for us, while $k \geq log_2(nR^2/p)$ from the analysis in [Vau05].

# Part II

# Multi-Party Setting

# Chapter 5

# Background on Threshold Cryptography

## 5.1 Introduction

In many scenarios, there exists a *group authority* that can be trusted to manage the group security. Typical responsibilities include access control, logging of traffic and usage, and key management. However, basing the security of the entire group on a single entity makes the system more vulnerable. Thus, it is beneficial, in general, to distribute the security tasks as much as possible.

The idea of distributing a cryptosystem so as to secure it against corruption of some threshold, e.g. a minority, of participating players is known as *threshold cryptography*. It was introduced in the proposals of Desmedt [Des87], Boyd [Boy89], Croft and Harris [CH89], and Desmedt and Frankel [DF90]. Threshold cryptography offers better fault tolerance than a centrally managed system : even if some entities are unavailable or compromised, others can still perform the task (see Figure 5.1). Threshold cryptography also offers better security since no single entity is trusted to perform a critical task in its entirety.

**Definition 5.1 (Threshold Secret Sharing).** *Let $t$, $n$ be positive integers where $t < n$. A $(t+1, n)$-**threshold secret sharing** is a method for sharing a secret value $\boldsymbol{x}$ among a set of $n$ entities, in such a way that any $t+1$ entities can reconstruct the value of $\boldsymbol{x}$, but no group of $t$ or fewer entities can do so.*

Digital Signatures is a common cryptographic operation that can be distributed

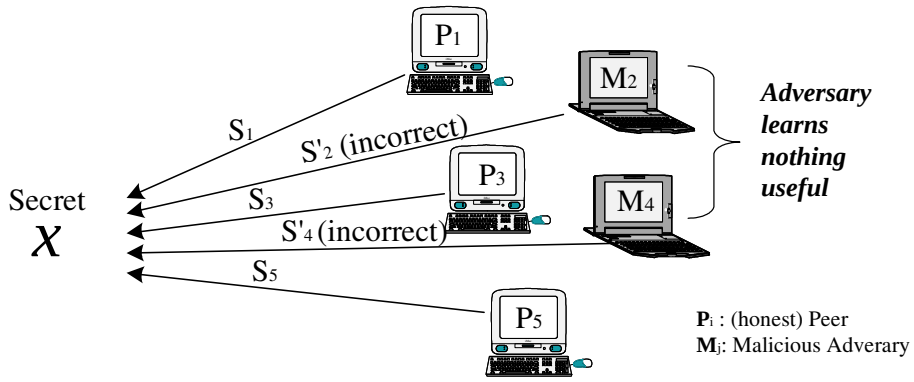



Figure 5.1: Concept of Threshold Cryptography

using threshold cryptography.

**Definition 5.2 (Threshold Signature Scheme).** *Let $\mathcal{S}$=(*KeyGen*,* Sig*,* Ver*) be a signature scheme. A $(t+1, n)$-**threshold signature scheme** $\mathcal{TS}$ for $\mathcal{S}$ is a set of protocols* TS-KeyGen*,* TS-Sig*,*TS-Ver *for the set of $n$ entities, where the followings are satisfied:*

1. TS-KeyGen *is the threshold key generation algorithm such that public/private key pairs $(y, x)$ are generated as if they were produced by the* KeyGen *algorithm of the regular signature scheme $\mathcal{S}$, but the private key $x$ is shared using a $(t+1, n)$-threshold secret sharing scheme.*

2. TS-Sig *is the distributed signing algorithm that allows any $t+1$ entities to collectively generate a valid signature as if it was produced by* KeyGen *algorithm for $\mathcal{S}$, but prevents $(t)$ or fewer entities can to do so. The verification algorithm is, therefore, the same as in the regular signature scheme $\mathcal{S}$.*

The security of a threshold scheme is defined as follows.

**Definition 5.3 ($t$-security and $t$-robustness).** *A $(t+1, n)$-threshold schemes (i.e., both threshold secret sharing and threshold signature) are said to be **$t$-secure** if any coalition of at most $t$ corrupt entities is unable to reconstruct the secret (or forge a valid signature) and **$t$-robust** if honest entities can efficiently reconstruct the secret (or produce a signature) even in the presence of at most $t$ malicious/corrupted entities.*

In threshold schemes, an adversary needs to compromise $t+1$ entities in order to expose the secret. Gradual break-ins into $t+1$ entities over a long period of time might be

possible since the secret shares that have been distributed remain unchanged. Therefore, traditional threshold secret sharing is not sufficient for long-lived secrets. This prompts the need to refresh individual secret shares, periodically, without changing the existing group secret. To this end, *proactive threshold secret sharing schemes* [OY91, HJKY95] are necessary to update individual secret shares under such constraints.

In the proactive cryptographic schemes, time is divided into time periods, called *update rounds*, which are determined by the common global clock (e.g., a day, a week, etc.). During an update round, the adversary can corrupt at most $t$ entities. We now introduce some definitions [HJKY95] related to proactive threshold schemes.

**Definition 5.4.** *A **static adversary** is an active adversary that corrupts at most $t$ entities in the entire lifetime of the system. A **mobile adversary** is an active adversary that corrupts a potentially new set of up to $t$ entities in each update round.*

**Definition 5.5.** *A **proactive** $(t+1, n)$**-threshold signature scheme**, based on techniques of proactive secret sharing, is a $(t+1, n)$-threshold signature scheme which remains $t$-secure and $t$-robust even in the presence of **mobile adversaries**.*

A proactive threshold signature scheme is achieved by the entities periodically updating their sharing of the secret signature key via a *distributed share update protocol*. Such an update protocol destroys the correlation between secret shares learned by corrupted entities in different time periods, so that the scheme can tolerate any number of corruptions throughout its lifetime as long as the number of simultaneously corrupted entities in any single time period does not exceed $t$.

## 5.2 Threshold Secret Sharing

In this section, we present Shamir's secret sharing scheme [Sha79] which is based on polynomial interpolation and is information theoretically secure. We will refer to it as TSS. To distribute shares of a secret $x$ among $n$ entities, a trusted dealer $TD$ chooses a polynomial $f(z)$ over $\mathbb{Z}_q$ of degree $(t)$:

$$f(z) = \sum_{i=0}^{t} a_i z^i \pmod{q} \tag{5.1}$$

where the constant term $a_0$ is set to the group secret $x$; $f(0) = a_0 = x$. *TD* computes each entity's share $x_i$ such that $x_i = f(id_i)$, where $id_i$ is an identifier of entity $P_i$, and securely transfers $x_i$ to $P_i$.

Then, any group[1] of $t+1$ entities who have their shares can recover the secret using the Lagrange interpolation formula:

$$\begin{aligned} f(z) &= \sum_{i=1}^{t+1} x_i \ \lambda_i(z) \pmod{q}, \\ \text{where } \lambda_i(z) &= \prod_{j=1, j\neq i}^{t+1} \frac{z - id_j}{id_i - id_j} \pmod{q} \end{aligned} \tag{5.2}$$

Since $f(0) = x$, the shared secret may be expressed as:

$$x = f(0) = \sum_{i=1}^{t+1} x_i \ \lambda_i(0) \pmod{q} \tag{5.3}$$

Thus, the secret $x$ can be recovered only if at least $t+1$ shares are combined. In other words, no coalition of less than $t+1$ entities yields any information about $x$.

## 5.3 Verifiable Secret Sharing

**Feldman's Verifiable Secret Sharing (VSS).** If we suppose that some entities can become malicious or compromised by an adversary, they may attempt to "cheat" by using incorrect secret shares in order to deny/disrupt the service. To remedy the situation, a more advanced technique, *Verifiable Secret Sharing* due to Feldman [Fel87], denoted by VSS, can be used. It basically provides a means to detect incorrect secret shares.

To be more specific, VSS setup involves two large primes $p$ and $q$, and an element $g \in \mathbb{Z}_p^*$ chosen in a way that $q$ divides $p-1$ and $g$ is an element of $\mathbb{Z}_p^*$ which has order $q$. The procedure for the *TD* to distribute the shares is the same as in Section 5.2. VSS is achieved by the following procedure:

1. *Secret Sharing and Witness generation.* The *TD* randomly selects a polynomial $f(z) = \sum_{i=0}^{t} a_i z^i$ (secret being shared is $a_0$), computes secret shares $x_i = f(id_i)$, and transfers them to each entity securely. Also, $TD$ chooses an element $g \in \mathbb{Z}_p^*$ of order $q$, and

[1] W.l.o.g. and for the ease of description, we consider a group consisting of first $t+1$ players.

computes $W_i$, for $i \in [0, t]$, called **witness**, such that $W_i = g^{a_i}$. Then, $TD$ publishes these $W_i$-s in some public domain (e.g., a directory server) [2].

2. *Share verification.* When each entity $P_i$ receives its share $x_i$, it verifies $x_i$ by checking:

$$g^{x_i} \stackrel{?}{=} \prod_{j=0}^{t} [W_j]^{(id_i)^j} \pmod{p}. \tag{5.4}$$

Feldman's VSS is secure under the discrete logarithm assumption; an adversary who statically corrupts at most $t$ entities and learns the secret key in the protocol, can be converted to compute a discrete logarithm.

**Pedersen's Verifiable Secret Sharing (Pedersen-VSS).** An information theoretically secure version of VSS was proposed by Pedersen based on Pedersen commitment scheme [Ped91a] (we described this commitment scheme in Section 2.7). We denote this scheme by Pedersen-VSS.

The scheme uses parameters $p$, $q$, and $g$ as in VSS and an additional element $h$ in the subgroup of $\mathbb{Z}_p^*$ generated by $g$. It is assumed that the adversary cannot compute $log_g(h)$. Pedersen-VSS has the following procedures:

1. *Secret Sharing and Witness generation.* The $TD$ randomly selects two polynomials $f(z) = \sum_{i=0}^{t} a_i z^i$ and $f'(z) = \sum_{i=0}^{t} b_i z^i$ (secret being shared is $a_0$). $TD$ computes secret shares $x_i = f(id_i)$, $x_i' = f'(id_i)$, and transfers them to each entity securely. Also, $TD$ broadcasts/publishes the witnesses $W_i = g^{a_i} h^{b_i}$, for $i = 0, 1, \cdots, t$.

2. *Share verification.* When each entity $P_i$ receives its shares $x_i, x_i'$, it verifies them by checking:

$$g^{x_i} h^{x_i'} \stackrel{?}{=} \prod_{j=0}^{t} [W_j]^{(id_i)^j} \pmod{p}. \tag{5.5}$$

## 5.4 Distributed Key Generation

In certain decentralized applications, a $TD$ can not be assumed, not even at the time of initialization and secret sharing. In such applications, the entities in the system

[2] In case of Pedersen-DKG, where the group polynomial is jointly selected by the entities, this step is carried out by each of the entities individually

themselves need to initialize the cryptosystem and generate its private and public keys. This distributed initialization is referred to as *distributed key generation*.

A simple approach, due to Pederson [Ped91b] and denoted by Pedersen-DKG, is to run $n$ parallel instances of VSS such that in each instance, each party $P_i$ behaves as dealer to distribute shares of its secret $a_{i0}$. The private key $x$ is set to be the sum of $a_{i0}$ values, and the corresponding public key $y = g^x$ becomes the product of $g^{a_{i0}}$ values broadcast by entities during VSS. Note that the private key $x$ is not known to any entity taking part in the protocol.

However, Pedersen-DKG is shown to be insecure [GJKR99b] – it does not guarantee that in the presence of an adversary private key $x$ remains uniformly distributed. In other words, an adversary can manipulate the protocol execution in such a manner that $x$ does not turn out to be uniformly distributed. Refer to [GJKR99b] for details regarding this insecurity of Pedersen-DKG. Note that in certain protocols that use distributed key generation, e.g., threshold DSS signatures [GJKR96b], the distribution of the output key $x$ needs to be uniform.[3]

Next we present the distributed key generation protocol of [GJKR99b] that guarantees a uniform distribution for the output key and is secure under the discrete logarithm assumption. We call it GJKR-DKG. The protocol consists of following steps.

1. Each $P_i$ runs an instance of Pedersen-VSS to share its secret $a_{i0}$. That is, $P_i$ chooses at random two polynomials $f_i(z)$, $f'_i(z) \in \mathbb{Z}_q$ of degree $t$ such that $f_i(0) = a_{i0}$, where $a_{i0}$ is a random secret that $P_i$ selects. Let $f_i(z) = a_{i0} + a_{i1}z + \cdots + a_{i,t}z^t \pmod q$ and $f'_i(z) = b_{i0} + b_{i1}z + \cdots + b_{i,t}z^t \pmod q$.

   $P_i$ broadcasts witnesses $W_{ik} = g^{a_{i0}}h^{b_{i0}} \pmod p$, computes each $P_j$'s shares $x_{ij} = f_i(id_j)$ and $x'_{ij} = f'_i(id_j)$ $(j \in [1, n], j \neq i)$, and securely sends it to $P_j$.

2. Each $P_j$ verifies each share that it received from other parties using the following equation:

$$g^{x_{ij}}h^{x'_{ij}} \stackrel{?}{=} \prod_{k=0}^{t} [W_{ik}]^{(id_j)^k} \pmod p$$

   If the above verification fails, $P_j$ broadcasts a complaint against $P_j$.

[3] On the other hand, some protocols can still be proven secure irrespective of the fact that the output distribution is not uniform. As an example, refer to [GJKR03] for a variant of threshold Schnorr signature scheme that remains secure when initialized with Pedersen-DKG.

3. Each party $P_i$ who received a complaint from party $P_j$, broadcasts the values $x_{ij}$ and $x'_{ij}$.

4. Each party marks as disqualified a party that either received at least $t+1$ complaints or answered a complaint with false values.

5. Each party builds a set $Q$ consisting of all the players which were not disqualified above. Each $P_i$ sets its secret shares as $x_i = \sum_{j \in Q} x_{ij}$ and $x'_i = \sum_{j \in Q} x'_{ij}$. The cryptosystem secret becomes $x = \sum_{i \in Q} a_{i0}$, but it is not known to anyone.

6. Each qualified party $P_i$ now broadcasts $W'_{ik} = g^{a_{ik}}$, for $k = 0, 1, \cdots, t$.

7. Each party $P_j$ verifies the values broadcast by other qualified party $P_i$ as follows:

$$g^{x_{ij}} \stackrel{?}{=} \prod_{k=0}^{t} \left[W'_{ik}\right]^{(id_j)^k} \pmod{p}$$

   If the verification fails, $P_j$ launches a complaint against $P_i$ and broadcasts value $x_{ij}$.

8. For each party $P_i$ against whom at least one complaint was valid, any set of $t+1$ parties reconstruct its corresponding secret $a_{i0}$ and compute $y_i = g^{a_{i0}} \pmod{p}$; for all others, $y_i = W_{i0}$. Finally, each party computes the public key of the cryptosystem $y$ such that $y = \sum_{i \in Q} y_i$. Note that $y = g^x \pmod{p}$.

## 5.5 Joint Zero Secret Sharing

This scheme, which appeared in [HJKY95], is a variant of distributed key generation where the generated key is zero. In other words, this scheme is the same as Pedersen-DKG except that in first step, each entity picks a random $t$-degree polynomial $f_i(z) \in \mathbb{Z}_q$ such that $f_i(0) = 0$. We refer to it as a *Joint Zero Secret Sharing*, denoted by JZSS.

It is used in proactive secret sharing, as we describe next.

## 5.6 Proactive Share Update

In order to protect threshold sharing schemes against *mobile adversaries*, it is required that the secret share holders run a *proactive share update* protocol [HJKY95],

denoted by PSU, in which the *secret shares* get re-randomized but the *cryptosystem secret* remains the same. The basic technique for re-randomization is the JZSS described above.

As shown in [HJJ+97a], an appropriate proactive update protocol can be integrated with a threshold signature scheme resulting in a *proactive signature scheme*.

The proactive update protocol proceeds as follows:

1. Every $P_i$ chooses a random *partial update polynomial* $\delta_i(z)$ over $\mathbb{Z}_q$ of degree $t$ with the constant term being zero as in the JZSS protocol. The sum of these partial update polynomials defines the *update polynomial*:

$$\delta(z) = \sum_{i=1}^{n} \delta_i(z) \pmod{q} \tag{5.6}$$

   Note that $\delta(0) = 0$.

2. For each pair of entities $(P_i, P_j)$, $P_i$ gives a share $\delta_i(id_j)$ of its partial update polynomial to $P_j$. Each $P_i$ computes then its new secret share $\tilde{s}_i$ as

$$\tilde{x}_i = x_i + \sum_{j=1}^{n} \delta_j(id_i) = x_i + \delta(id_i) \pmod{q} \tag{5.7}$$

   where $x_i$ is $P_i$'s existing share.

   Note that all the information pertaining to this protocol except of the new share $\tilde{x}_i$ is then erased.

3. Since $x_i = f(id_i)$ for all $P_i$, the new secret-sharing polynomial $\tilde{f}(z)$ such that $\tilde{f}(id_i) = \tilde{x}_i$ for all $P_i$ is defined as

$$\tilde{f}(z) = f(z) + \delta(z) \pmod{q} \tag{5.8}$$

   and therefore it is a random $t$-degree polynomial such that $\tilde{f}(0) = f(0) = x$.

The scheme can be made robust against active adversaries using Feldman's VSS.

## 5.7 Distributed Share Generation

In certain applications, such as in dynamic ad hoc groups, one needs to add (in a distributed manner) new entities to the group over a period of time. This can be achieved

if the existing entities $P_i$ (possessing secret share $x_i$) in the group can generate the secret share $x_{n+1}$ for the new entity $P_{n+1}$ in a distributed manner. We call this *Distributed Share Generation*, referring it to as DSG.

A naive solution to distributed share generation is that any set of $t+1$ entities simply provide their secret shares to $P_{n+1}$. Using these secret shares, $P_{n+1}$ can interpolate the polynomial $f(z)$ and thus compute its secret share $x_{n+1} = f(id_{n+1})$. Clearly, this solution is **insecure** as $P_{n+1}$ is exposed to not only the secret shares of other entities but also the secret sharing polynomial and thus the cryptosystem secret.

We now present a secure protocol for distributed share generation. Its variant appeared in [HJKY95]. This protocol is executed between the new entity $P_{n+1}$ and existing entities $P_i$, for $i = 1, 2, \cdots, n$. The protocol starts by $P_{n+1}$ sending a "join-request" to the group. A set of $m$ $(t+1 \leq m \leq n)$ entities respond with a message containing their identifiers. $P_{n+1}$ chooses a set of $t+1$ out of $m$ entities that responded, forms a "sponsors list" $SL$ consisting of corresponding identifiers, and sends it to each one of them. For simplicity and w.l.o.g., let us assume that $SL = 0, 1, \cdots, t$. Rest of the protocol involves following steps.[4]

1. All of the $t+1$ sponsors perform the JZSS, as in Section 5.5, by setting the constant term of their respective polynomials to zero. Also, the witness values of the polynomials are broadcast to enable VSS.

2. At the end of JZSS, every $P_j \in SL$ possesses a random share $R_j$ of the shared secret zero.

3. Now, $P_j$ provides the *shuffled* partial secret share (over a secure channel) $\tilde{x}_{n+1}^{(j)}$ for $P_{n+1}$:
$$\tilde{x}_{n+1}^{(j)} = x_j \lambda_j(id_{n+1}) + R_j \lambda_j(0) \pmod{q} \tag{5.9}$$

4. $P_{n+1}$ adds up the partial secret shares $\tilde{x}_{n+1}^{(j)}$ to obtain its secret share $x_{n+1}$. Note that $\sum_{j=0}^{t} x_j \lambda_j(id_{n+1}) = f(id_{n+1})$, $\sum_{j=1}^{t} R_j \lambda_j(0) = 0$, and thus $x_{n+1} = \sum_{j=0}^{t} \tilde{x}_{n+1}^{(j)}$.

The idea of VSS can be easily extended to verify correctness of partial shares that $P_{n+1}$ receives from sponsors. Since $\tilde{x}_{n+1}^{(j)} = x_j \lambda_j(id_{n+1}) + R_j \lambda_j(0)$, as described above, to

[4]The messages in Steps 1 and 2 can be routed via the new node $P_{n+1}$, as was proposed in [LL00]. This avoids any direct communication among the $t+1$ sponsors, however, at the cost of $P_{n+1}$ becoming the bottleneck.

check if $\tilde{x}_{n+1}^{(j)}$ is correctly computed and trace which of the $P_j$-s sent back false values, if any, following equation is verified:

$$g^{\tilde{x}_{n+1}^{(j)}} \stackrel{?}{=} \left[\prod_{k=0}^{t} (W_k)^{{id_j}^k}\right]^{\lambda_j(id_{n+1})} g^{R_j \lambda_j(0)} \pmod{p} \tag{5.10}$$

Here, $g^{R_j}$ is computed using the broadcast witness values of the shared polynomial among the sponsors.

As outlined in [HJKY95], the above DSG protocol remains secure under the discrete logarithm assumption; an adversary (acting on behalf of the new entity $P_{n+1}$) who corrupts at most $t$ entities in the group and learns the secret key from the execution of the protocol, can be converted to solve the discrete logarithm problem.

# Chapter 6

# Distributed Signatures

*In this chapter, we focus upon RSA-based distributed (threshold/proactive) signatures. We first present an efficient attack on a recently proposed proactive RSA scheme [LKZ+04], in which an admissible threshold of malicious group members can completely recover the group RSA secret key in the course of the lifetime of this scheme. We then carry on to construct a new (provably secure) proactive RSA scheme based on a corrected use of the scheme of [LKZ+04]. The new scheme offers a simpler alternative to the best previously known proactive RSA scheme given by Tal Rabin, and is applicable to build online certification, revocation and timestamping services.*

## 6.1 Introduction: Background and Motivation

**Threshold Cryptography, Threshold and Proactive Signature Schemes.** Recall that the idea of distributing a cryptosystem so as to secure it against corruption of some threshold, e.g. a minority, of participating players is known as *threshold cryptography*. It was introduced in the proposals of Desmedt [Des87], Boyd [Boy89], Croft and Harris [CH89], and Desmedt and Frankel [DF90], which were based on Shamir's *polynomial secret-sharing* technique [Sha79].

A $(t+1, n)$ threshold signature scheme [DF90] enables any subgroup of $t+1$ members in a group consisting of $n > t$ members, to collaboratively sign a message on

behalf of that group. This is achieved by secret-sharing the signature key, e.g. the RSA secret key, among the group members, and allowing them to compute a signature on some message via a distributed protocol in which the members use the shares of the signature key instead of the key itself. The scheme is said to be *t-secure* if any coalition of at most $t$ corrupt members is unable to forge a valid threshold signature on any message which honest members would not sign, and *t-robust* if honest group members can efficiently produce a valid signature even in the presence of at most $t$ malicious members. To achieve $t$-security, a threshold signature scheme must in particular protect the secrecy of the signature key as long as no more than $t$ of the group members are corrupt.

A *proactive* signature scheme [HJJ+97a], based on techniques of proactive secret sharing [OY91, HJKY95], is a threshold signature scheme which remains secure and robust even if in every *time period*, called "share update interval", a possibly different set of $t$ group members is corrupted. This is achieved by the members periodically updating their shares of the secret signature key via a distributed share update protocol. Such an update protocol should destroy the correlation between secret shares learned by corrupted members in different time periods, so that the scheme can tolerate any number of corruptions throughout its lifetime as long as in any single time period the number of simultaneously corrupted members does not exceed $t$. A proactive signature scheme offers stronger security guarantee then a threshold scheme, especially in an application which might come under repeated attacks, like a certification authority or a timestamping service. Moreover, a proactive scheme offers more secure management of a system whose size and make-up need to change throughout its lifetime. Efficiency of the distributed signature protocol involved in a proactive signature scheme is very important in some applications, like in a timestamping service, or in the decentralized control of peer-to-peer groups, ad-hoc groups, or sensor networks [KZL+01, STY03]. An efficient proactive scheme for RSA signatures is especially important because RSA signatures are widely used in practice, and because verification of RSA signatures is several orders of magnitude faster than verification of other signatures.

**Prior Work on Threshold and Proactive RSA.** While the work of Herzberg et al. [HJKY95, HJJ+97b] and Gennaro et al. [GJKR96b, GJKR99b] quickly yielded efficient secure proactive signature schemes for discrete-log based schemes like Schnorr [Sch91] or DSS [NIS91] signatures, the work on secure proactive RSA schemes progressed more slowly, and the initial threshold RSA scheme of Desmedt and Frankel [DF90] was robust only

against crashes and not malicious faults, and had only heuristic security. The difficulty in adopting Shamir's polynomial secret-sharing technique to threshold RSA was caused by the fact that the RSA private key $d$ is an element of a group $\mathbb{Z}_{\phi(N)}$, where $\phi(N)$ needs to remain hidden from all players because it allows immediate computation of the private key $d$ from the RSA public exponent $e$. This difficulty was overcome by the schemes of Frankel et al. [FD92, DDFY94] which provided a proof of security but used secret shares which were elements of a polynomial extension field of $\mathbb{Z}_n$, which increased the cost of the signature operation by a factor of at least $t$. These schemes were then extended to provide robustness against malicious faults by [FGY96, GJKR96a]. Subsequently, Victor Shoup [Sho00] presented a threshold RSA signature scheme which was robust and provably secure with optimal adversarial threshold $t < n/2$, and which did away with the extension field representation of the shares, thus making the cost of the signature operation for each participating player comparable to the standard RSA signature generation.

Proactive RSA scheme is a harder problem because it requires the players to re-share the private key $d$ in each update round even if no single player is allowed to know the secret modulus $\phi(N)$. The first proactive RSA scheme of Frankel et al. [FGMY97b] solved this problem using additive secret sharing over integers in conjunction with combinatorial techniques which divide the group of $n$ players into two levels of families and sub-families. However, the resulting proactive protocol did not achieve optimal adversarial threshold $t < n/2$ and did not scale well with the group size $n$. These shortcomings were later overcome by the same authors [FGMY97a], who showed that the RSA private key $d$ can be shared over integers using polynomials with specially chosen large integer coefficients that simultaneously allowed interpolation without knowing $\phi(N)$ and unpredictability of the value $d$ given any $t$ polynomial shares. In this solution, even though the underlying secret sharing was polynomial, the players need to create a one-time additive sharing for every group of players participating in threshold signature generation. A simpler and more efficient proactive RSA scheme was then given by Tal Rabin [Rab98]. Her solution also used sharing of the private key over integers, and employed shares of size about twice the length of the private key. The new idea was that the secret $d$ was shared additively among the players, every share was backed-up by a secondary level of polynomial secret sharing, and the proactive update consisted of shuffling and re-sharing of the additive shares.

**Limitations and Open Problems in Proactive RSA.** The proactive RSA schemes of

[FGMY97b, FGMY97a, Rab98] leave at least two important problems unaddressed. While the new proactive RSA scheme we present in this chapter does not solve these problems either, the techniques we present might help solve these problems in the future. The first problem is that of handling *adaptive* rather than *static* adversaries. The static adversary model assumes that the adversary decides which player to corrupt obliviously to the execution of the protocol, while the adaptive model allows the adversary to decide which player to corrupt based on his view of the protocol execution. This difference in the adversarial model is not known to be crucial for the security of the above protocols in practice. However, the above protocols are not known to be adaptively secure, while the known adaptively secure RSA schemes [FMY99a, CGJ+99, FMY99b, FMY01] are significantly less efficient.

The second problem is that of requiring some form of additive rather than polynomial secret-sharing. The additive sharing implies that the shares of all temporarily unavailable players need to be reconstructed by the active players that participate in the signature generation protocol. This hurts both the efficiency and the resilience of a scheme in applications where one player might be temporarily unavailable to another without an actual corruption by the adversary. Since the threshold (but not proactive) RSA signature schemes discussed above do not resort to additive sharing, this is a disadvantage of the currently known proactive RSA schemes. COCA (a distributed on-line certification authority) [ZSvR02] employs a modified version of Rabin's RSA scheme which overcomes this problem of availability.[1] However, this scheme, which is based on combinatorial secret sharing as opposed to the additive sharing of Rabin, is applicable only for small groups, because in large groups the number of combinations $\binom{n}{t}$ becomes intractable.

**Insecurity of the Proactive RSA Scheme Proposed in the URSA Ad Hoc Network Access Control Protocol.** In an effort to mitigate the aforementioned (second) problem of the known proactive RSA signatures, Luo, et al. proposed a new proactive RSA scheme, geared to wards providing a security service, called "URSA", in mobile ad hoc

[1] COCA signing example: for $(2, 4)$ signing, the secret $d$ is split in shares $s_1$, $s_2$, $s_3$, $s_4$ corresponding to $l = \binom{n}{t} = \binom{4}{1} = 4$ sets, $P_1 = \{p_1\}, P_2 = \{p_2\}, P_3 = \{p_3\}, P_4 = \{p_4\}$. Each player $p_i$ gets a share corresponding to the set it does not belong to. So, $p_1$ gets $\{s_2, s_3, s_4\}$; $p_2$ gets $\{s_1, s_3, s_4\}$; $p_3$ gets $\{s_1, s_2, s_4\}$ and $p_4$ gets $\{s_1, s_2, s_3\}$. The shares $s_i$-s $\in [-lN^2, lN^2]$ as in Rabin's, where $N$ denotes the RSA modulus. In order to sign, every player creates a signature using all the secret shares it possesses. So, $p_1$'s signature is $m^{s_2}$ $(mod\ N)$, $m^{s_3}$ $(mod\ N)$, $m^{s_4}$ $(mod\ N)$ and so on for other players. Now the signatures from any two players can be accumulated, which will yield the RSA signature $m^d$ $(mod\ N)$.

networks. The original description of this proactive RSA scheme and the URSA application can be found in [LL00]. Subsequently both the proactive RSA scheme and URSA were described in [KZL+01, KLX+02, LZK+02], and most recently in a journal version [LKZ+04]. The URSA proactive RSA scheme can be applicable to MANETs because it avoids the need to access all shares during the threshold signature protocol. This is because it relies solely on Shamir's *polynomial* secret sharing scheme [Sha79], as opposed to resorting to an additional layer of *additive* secret sharing, as is done by the two most efficient provably secure proactive RSA schemes [FGMY97a, Rab98] discussed above. The core of the URSA proactive RSA scheme is the so-called *t-bounded offsetting algorithm* which is used to reconstruct the RSA signature $m^d \pmod N$ from $t+1$ *partial signatures* produced individually by the $t+1$ members participating in the signing protocol.

The first problem with this scheme was pointed out in [NTY03]. Namely, contrary to what the authors of the proposal claimed, their scheme does not provide robustness in signature generation in the presence of $t$ malicious members. Simply speaking, the robustness mechanisms proposed by the authors are faulty because they require certain verification equations to hold even though they in fact do not hold, because the equations involve computation in two different groups (see [NTY03] for more details). Hence, the set of $t$ malicious members can prevent the honest members from efficiently creating a valid signature. However, this robustness problem in the *t-bounded* proactive RSA scheme can be solved, if the secret sharing is performed over a large prime number, instead over the RSA modulus $N$ of the original proposal, coupled with special-purpose zero-knowledge proof protocols for proving equality of discrete logarithms in two different groups [CM99b] and range of a discrete logarithm [Bou00]. Such proof protocols are not very fast, but their expense can be tolerated because they would need to be executed only in the (rare) case of a corrupted member providing an incorrect partial signature to other members.[2]

Since the $t$-robustness of this scheme can be ensured by the above modifications, there remains a question if the (modified) URSA proactive RSA scheme is *secure* against a coalition of corrupt $t$ members[3] whose goal is not to prevent members from is-

[2]This will be a rare occurrence because a malicious member behaving in such a manner would be detected by the honest players, and therefore would be subsequently revoked from the group.

[3]In the standard $(t+1, n)$ threshold cryptographic model, any set of $t+1$ out of a total of $n$ members share the ability to perform a cryptographic operation (e.g., signing) in the presence of atmost $t$ corruptions. All the earlier versions of the URSA papers [LL00, KZL+01, KLX+02, LZK+02] work in this standard model. However, the latest journal version [LKZ+04] (without any stated reasoning whatsoever) employs a

suing signatures but to learn the secret-shared RSA signature key and to be thus able to forge signatures on the group's behalf. The question is interesting because the proposed URSA proactive RSA scheme, amended as described above, would provide efficiency and functionality advantages over the best known provably secure proactive RSA schemes [FGMY97b, FGMY97a, Rab98]. However, the answer turns out to be negative.[4]

**Our Contributions: Explicit Attack on the Proactive RSA Scheme in the URSA Protocol and a New Simpler Proactive RSA.** We make a two-fold contribution in this chapter of the thesis.

1. We demonstrate the insecurity of the URSA proactive RSA signature scheme by constructing an explicit attack in which the admissible group of $t$ corrupted members colludes in the proactive protocol in such a way so that they reconstruct the whole RSA secret key $d$ after a realistic number of runs of the proactive update protocol and the threshold signature protocol.

   Our attack exploits the fact that the *t-bounded offsetting* threshold RSA signature protocol, which is employed in the URSA proactive RSA scheme, leaks certain seemingly innocuous information about the secret signature key. The information that the adversary learns about the secret key in a run of the signature protocol depends on the current sharing of the secret key and on which group of members participates in the protocol. While it is not clear how dangerous this released information is for a single secret sharing, in a *proactive* signature scheme the secret sharing is refreshed with every proactive update, and therefore the released information about the secret key can be different in each update interval. It turns out that the corrupted members can influence an execution of the *update* protocol in such a way that the executions of the *signature* protocol during the subsequent update interval will release information which is both new and correlated with the information the adversary has gained so far. Thus our attack can be seen as a simple search algorithm, where the information learned in a signature protocol tells the adversary which branch to pick next, and the

slightly modified and a non-standard model, wherein any $t+1$ members share the signing operation, but only a maximum of $t-1$ corruptions are allowed. The attack we describe in this chapter is based on the standard model. However, as we discuss later in Section 6.4, the URSA schemes appears insecure even in the non-standard model.

[4] In particular, although Luo, et al. claim that their scheme is provably secure, the security proofs that appear in [LL00] are incorrect.

proactive update protocol allows the adversary to pick that branch.

The attack poses a realistic threat. For example, for the threshold size $t = 7$ and for the RSA public key of $e = 65537$ (utilized in the implementation of URSA [KZL+01]) and 1024-bit RSA modulus $N$, our attack needs 163 executions of the proactive update protocol and 1148 runs of the signature protocol to succeed. The attack succeeds assuming that throughout these 163 update periods, the $t$ corrupted players belong to the so-called "update group" of players which play an active role in the proactive update protocol.

However, the URSA proactive RSA protocol is more vulnerable than what is immediately implied by the above attack. First, our attack does not make use of all the information leaked in the signature protocol, thus it is quite possible that another attack, which does utilize all the available information succeeds in recovering the private RSA key even faster and/or succeeds in recovering the key even if, say, only a smaller subset of the special "update group" of players is corrupted. Moreover, even if the attack we describe is slowed down, for example by slowing down the rate of the proactive updates, it recovers $512 + (r - log_{t+1}(e)) * log_2(t + 1)$ most significant bits of $d$ after $r > log_{t+1}(e)$ update rounds, which gives $512 + 3(r - 5)$ MSB bits of $d$ for the above $e = 65537$ and $t = 7$. Therefore our attack raises doubts about the security of the URSA proactive RSA scheme even for smaller number of rounds. It is hard to say much about the security of this protocol, for example after $r = 34$ rounds, because while it is not currently known how to recover the whole RSA key knowing 600 of the MSBs of $d$, we also do not know any arguments that RSA remains secure with this side information about $d$ revealed, and it would be rather surprising if such arguments existed.

2. Based on the corrected use of a technique discovered by the authors of the URSA scheme, we present a new robust and provably secure optimal-threshold proactive RSA scheme. Our scheme is known to be secure only in the static model, and it employs top-level additive sharing similarly as the Rabin's scheme [Rab98], but it is interesting for the following reasons: (1) It is simpler than the previous schemes; (2) It offers factor of 2 improvement in share size and signature protocol efficiency for general RSA public keys, and factor of 4 improvement for the common case of public exponents like 3, 17, or 65537, over the most efficient previously known proactive

RSA scheme [Rab98] *as originally analyzed* by [Rab98]; (3) The new scheme led us to a tighter security analysis of the [Rab98] scheme, which resulted in similar, up to a logarithmic factor, efficiency improvements for the [Rab98] scheme; (4) The new scheme offers an interesting case of a technique, invented by Lu and Luo [LL00], which makes a distributed protocol run faster but leaks some partial information about the shared secret. This partial information leakage led to an efficient key-recovery attack [JSY04] on the original scheme of [LL00]. Yet, with some fixes, this partial information leakage can be provably neutralized and the same technique results in a provably secure scheme presented here; (5) Finally, our scheme offers new techniques which could aid in overcoming the two problems that still haunt proactive RSA solutions, namely achieving efficient adaptive security and the removal of additive sharing.

However, it is important to note that because of the additive sharing, while the new scheme is applicable to on-line certification authorities (such as COCA [ZSvR02]) or on-line time-stamping services, the scheme is *not* applicable to group access control for groups like MANETs, on-line peer-to-peer groups, sensor nets, etc. Thus the efficient provably secure proactive RSA signature which would avoid additive sharing and be applicable in such contexts, remains an open problem.

**Organization.** The rest of this chapter is organized as follows. We begin by describing the URSA proactive RSA signature scheme in Section 6.2. We then present an attack on this scheme in Section 6.3. The implications of our attack for the security of the URSA scheme are discussed in Section 6.4. Section 6.5 presents our new proactive RSA scheme followed by its security proof in Section 6.6. Finally, in Section 6.7 we show an efficiency improvement for the proactive RSA scheme of [Rab98].

## 6.2 The Proactive RSA Signature Scheme in URSA

In this section we describe the proactive RSA signature scheme of [LL00, KZL+01, KLX+02, LZK+02, LKZ+04] used in the URSA ad hoc network access control protocol. We will refer to this scheme as an "URSA proactive RSA signature scheme". The description includes the system set-up, the "*t-bounded offsetting*" threshold signature generation protocol, and the proactive share update protocol.

### 6.2.1 The Setup Procedure

A trusted dealer $TD$ is involved in a one-time setup to bootstrap the system. The dealer is not required hereafter and in fact is assumed to vanish from the scene, or, equivalently, to erase his memory. $TD$ generates the standard RSA private/public key pair, i.e. it picks two random primes $p$ and $q$, sets $N = pq$, sets $(e, N)$ as a public key where $gcd(e, N) = 1$, and as a private key it sets a number $d < N$ such that $ed = 1 \bmod \phi(N)$, where $\phi(N) = (p-1)(q-1)$.

Once the standard RSA key pair is chosen, $TD$ secret-shares the RSA secret key $d$ using a slight modification of Shamir secret sharing [Sha79]. Namely, $TD$ selects a random polynomial $f(z)$ over $\mathbb{Z}_N$ of degree $t$, such that the group secret is $f(0) = d \pmod N$. Next, $TD$ gives to each member $M_i$, for $i = 1, \cdots, n$, a secret share $ss_i = f(i) \pmod N$. Notice that the secret $d$ is shared over a composite modulus $N$ as opposed to a prime modulus as in the original scheme of Shamir, but our attack does not depend on what modulus is used in the secret sharing.

### 6.2.2 The Threshold Signature Protocol

The goal of the threshold RSA signature protocol is to generate in a distributed manner an RSA signature $s = m^d \pmod N$ under the secret-shared key $d$. The URSA threshold RSA signature protocol consists of two phases: First each participating member creates its *partial signature* on the intended message and sends it to the signature recipient, and then the recipient locally reconstructs the RSA signature from these partial signatures.

**Partial Signature Generation:** Let $G$ denote the set of identifiers of the $t+1$ members in the group who participate in the threshold signature protocol. Using polynomial interpolation we can write the secret key $d$ as

$$d = \sum_{j \in G} ss_j \; l_j^{(G)} \pmod N$$

where $l_j^{(G)} = \prod_{i \in G, i \neq j} \frac{(-i)}{j-i} \pmod N$ Notice that $N = pq$ has only two very large factors, and therefore all the elements $(j-i)$ for $i, j \in G$ will have inverses modulo $N$. Each member $M_j$, for $j \in G$, outputs his partial signature $s_j^{(G)}$ on $m$ as

$$s_j^{(G)} = m^{d_j^{(G)}} \pmod N \text{ , where } d_j^{(G)} = ss_j \; l_j^{(G)} \pmod N \tag{6.1}$$

**Signature Reconstruction:** On receiving $t+1$ partial signatures $s_j^{(G)}$ from the $t+1$ group members $M_j$ in $G$, the signature recipient reconstructs the RSA signature $s$ using the *"t-bounded-offsetting"* algorithm which works as follows. Since $\sum_{j\in G} d_j^{(G)} = d \pmod N$ and $0 \leq d_j^{(G)} \leq N-1$ for all $j$'s, therefore

$$d = \sum_{j\in G} d_j^{(G)} - \alpha^{(G)} N \quad \text{(over the integers)} \tag{6.2}$$

for some integer $\alpha^{(G)} \in [0,t]$. Equation (6.2) implies that

$$s = m^d = (\prod_{j\in G} s_j^{(G)}) m^{-\alpha^{(G)} N} \pmod N$$

for some integer $\alpha^{(G)} \in [0,t]$. Since there can be at most $t+1$ possible values of $\alpha^{(G)}$, the signature recipient can recover $s = m^d \pmod N$ by trying each of the $t+1$ possible values $Y_\alpha = Y(m^{-N})^\alpha \pmod N$ for $Y = \prod_{j\in G} s_j^{(G)}$ and $\alpha = 0, \cdots, t$, and returning $s = Y_\alpha$ if $(Y_\alpha)^e = m \pmod N$. The most significant cost factor in this procedure is an exponentiation $m^{-N} \pmod N$, and therefore the computational cost of the URSA threshold RSA signature protocol for each of the signers and for the recipient is about one full (1024 bit) exponentiation modulo $N$.

**Remark:** It is important to note that the above *t-bounded offsetting* threshold RSA signature algorithm **reveals the value of** $\alpha^{(G)}$, which, as will be described in section 6.3, leaks some information about the secret-shared private key $d$ to an adversary who corrupts $t$ of the players participating in group $G$. This information leakage in fact exposes the whole proactive RSA scheme to an efficient key-recovery attack.

### 6.2.3 The Proactive Share Update Protocol

The goal of the proactive share update protocol is to re-randomize the secret sharing of the private RSA key $d$ held by the group members. This protocol was first proposed for both proactive secret sharing and for proactive cryptosystems and signature schemes by Herzberg, et al. [HJKY95, HJJ+97a]. The URSA proactive share update protocol is a variant of this protocol which is more efficient, especially in settings where some players can be inactive or temporarily disconnected from others, as in MANETs.

**The "Classic" Share Update Protocol.** The proactive share update protocol of [HJKY95, HJJ+97a] proceeds as follows (when sharing is done modulo $N$): Every member $M_j$ chooses

a random *partial update polynomial* $\delta_j(z)$ over $\mathbb{Z}_N$ of degree $t$ with the constant term being zero. The sum of these partial update polynomials defines the *update polynomial* $\delta(z) = \sum_{j=1,n} \delta_j(z) \pmod N$. Note that $\delta(0) = 0$. For each pair of members $(M_j, M_i)$, player $M_j$ gives a share $\delta_j(i)$ of his partial update polynomial to $M_i$. Each member $M_i$ then computes his new secret share (to be used in the threshold signature protocol in the subsequent update interval) as

$$ss_i = ss'_i + \sum_{j=1}^{n} \delta_j(i) = ss'_i + \delta(i) \pmod N$$

where $ss'_i$ is $M_i$'s existing share, i.e. a share this player used in the previous interval. All the information pertaining to this protocol except of the new share $ss_i$ is then erased. Note that if $ss' = f'(i) \pmod N$ for all $i$ then the new secret-sharing polynomial $f(z)$ is defined as $f(z) = f'(z) + \delta(z) \pmod N$, and it is therefore a $t$-degree polynomial s.t. $f(0) = f'(0) = d \pmod N$.

However, it is important to notice that the new secret-sharing polynomial $f(z)$ is not necessarily a *random* $t$-degree polynomial s.t. $f(0) = d \pmod N$. This is because a corrupt player $M_j$ can distribute its update polynomial $\delta_j(z)$ only after all the other corrupt players $M_i$ see their shares of all the other update polynomials. In this way, the corrupt players can control the new secret-sharing polynomial $f(z)$ to some degree, by controlling the shares of $f(z)$ held by the corrupted players. In provably-secure proactive schemes that employ this proactive share update protocol, like the proactive DSS or BLS signatures [GJKR99a, Bol03], this adversarial ability does not pose any harm. However, as we will see in section 6.3, this control ability means trouble for the URSA proactive RSA scheme since the information about shared secret $d$ leaked in the threshold signature protocol depends precisely on the shares held by the corrupted players.

**The URSA Two-Stage Modification of this Protocol.** The URSA proactive RSA scheme utilizes a simple modification of the above share update protocol which improves the protocol's efficiency. This modification can indeed be used to speed up all proactive cryptosystems that use the [HJKY95] protocol, as the above mentioned schemes of [GJKR99a, Bol03]. The URSA proactive share update protocol consists of two stages. The protocol relies on an existence of a designated group of players $\Omega$, which we will call an "update group", consisting of $t$ group members.[5] The first stage of the protocol proceeds

[5] In some URSA descriptions it seems that the $\Omega$ group needs to have $t+1$ and not $t$ members. We believe

exactly like the above protocol of Herzberg et al., except that only the players $\Omega$ participate in it. In other words, players $M_j \in \Omega$ create their update polynomials $\delta_j(z)$, send their shares $\delta_j(i)$ to other members $M_i \in \Omega$, and thus the players in $\Omega$ can be said to hold in secret-shared form the update polynomial $\delta(z)$ defined as $\delta(z) = \sum_{j \in \Omega} \delta_j(z) \pmod N$. In the second stage of the URSA proactive share update protocol, the members of the update group $\Omega$ provide shares of this update polynomial $\delta(i)$ to *all* remaining (and non-revoked) group members $M_i \notin \Omega$. In this way all group members $M_i$ will get their update share and can compute the new share $f(i) = f'(i) + \delta(i) \pmod N$ as before.

The URSA papers describe two protocols for how these $\delta(i)$ update shares are transferred from the $\Omega$ players to their final destinations. Even though the details of the second version of this protocol are a little unclear, these details do not affect the attack we describe in this chapter. For completeness, we sketch the two variants as follows: In the first version, each $M_i$ gets its $\delta(i)$ share by communicating with each of the players $M_i \in \Omega$ directly. The players in $\Omega$ jointly reconstruct the $\delta(i)$ value by first sharing masking random values among themselves. In the second version, either the $\delta(i)$ update shares or the whole $\delta(z)$ polynomial (this version is not very clear to us), appear to be distributed to the rest of the group members encrypted under the group public key $(e, N)$. Presumably, each member $M_i$ reconstructs his share of this update polynomial $\delta(i)$ by contacting her $t+1$ neighbors who either decrypt her encrypted share or decrypt the whole polynomial $\delta(z)$ and evaluate it at point $i$ at the same time. It is not clear how this second version can be implemented securely, but the first version is standard and we see no vulnerabilities in it.

## 6.3 An Attack on the URSA Proactive RSA Scheme

In this section we present an efficient key-recovery attack on the URSA proactive RSA signature scheme summarized in the previous section. The roots of this attack lie in the *t-bounded offsetting algorithm* which is the core of the URSA threshold signature protocol.

---

that $t$ members is enough, and that the issue does not have a significant bearing on anything considered in this chapter.

### 6.3.1 Overview of the Attack

An adversary against a proactive signature scheme, and thus also against a MANET group access control mechanism like URSA which utilizes a proactive signature scheme, is able to compromise any set of at most $t$ members in the group in every *update round* (see below). After compromising a member, the adversary learns its corresponding secret share and can force this member to behave arbitrarily in the protocol. We assume the worst case where all the $t$ corrupted members collude, and in fact all corruptions are simply scheduled and controlled by a single entity, called an "adversary" and denoted by $\mathcal{A}$.

The goal of the adversary in our attack is to recover the secret-shared private RSA key $d$. The full attack holds as long as these $t$ members form the "update group" $\Omega$ (see section 6.2.3 above), and it holds regardless of what indices these players hold. However, for the sake of simplicity in the exposition, we will assume that the adversary corrupts members $M_1, \cdots, M_t$ throughout the lifetime of the scheme, and that these players also always form the $\Omega$ update group. We note, however, that our attack does *not* depend on the ability of the adversary to corrupt a *different* set of members every update period. This means that for example, as long as the $\Omega$ group is allowed not to change between the updates, the adversary can recover the private RSA key quite quickly if only he corrupts that subset and otherwise follows the protocol so as to avoid detection and revocation of these corrupted members from the group.

**Information Leakage in the Signature Protocol.** Assume that $\mathcal{A}$ participates in the threshold signature protocol on some message in which the set of participating members $G$ (see section 6.2.2 above) is made of all the corrupted members $M_1, \cdots, M_t$ and a single honest member $M_p$, for some $p \in [t+1, \cdots, 2t]$. (Here too, the attack works for other players $M_p$, but we fix the above $t$ values of $p$ to simplify the presentation.) Let $G_p$ represent the set of identifiers $\{1, \cdots, t, p\}$ corresponding to the members participating in this run of the threshold signature protocol. By equation (6.2), the secret key $d$ satisfies the following equation for some integer $\alpha^{(G_p)} \in [0, t]$:

$$d = \sum_{j \in G_p, j \neq p} d_j^{(G_p)} + d_p^{(G_p)} - \alpha^{(G_p)} N \quad \text{(over the integers)}$$

Let us denote $S_p = \sum_{j \in G_p, j \neq p} d_j^{(G_p)}$ (over integers) and $D_p = S_p \pmod N$. Note that since $\mathcal{A}$ knows $ss_1, ss_2, \cdots, ss_t$, he can compute $S_p$ and $D_p$.

By employing the reconstruction using the *t-bounded offsetting algorithm*, $\mathcal{A}$ learns the value of $\alpha^{(G_p)}$ corresponding to this signing group $G_p$. Now, note that from values $\alpha^{(G_p)}$ and $S_p$, the adversary also learns whether the shared secret $d \in [0, \cdots, N-1]$ is less than or greater than $D_p$. This is because $S_p < \alpha^{(G_p)}N$ if and only if $d < D_p$; and $S_p \geq \alpha^{(G_p)}N$ if and only if $d \geq D_p$.

**Utilizing the Information Leakage to Recover the Key.** At first sight, the information of whether the secret RSA key $d$ is left or right of some value $D_p$ in the $[0, \cdots, N-1]$ range seems to provide only information on the few most significant bits of $d$. However, recall that over the lifespan of the system, the members update their secret shares by performing the proactive share update procedure. As we will see below, it turns out that during this procedure, as long as the "update group" $\Omega$ is formed by the corrupted players $\{M_1, \cdots, M_t\}$, the adversary can choose the values of his new shares $ss_1, ss_2, \cdots, ss_t$, which gives him complete freedom in specifying the resulting values $D_p$, for $p = t+1, \cdots, 2t$, to be any values that he wants (we describe this process in subsections 6.3.2 and 6.3.3 below). Since in any subsequent run of the threshold signature protocol involving members $M_1, \cdots, M_t, M_p$ the adversary learns whether the secret $d$ lies to the left or to the right of the corresponding value $D_p$ (for $p = t+1, \cdots, 2t$), the adversary can learn most about $d$ if the chosen values $D_{t+1}, \cdots, D_{2t}$ divide the range $[0, N-1]$ into $t+1$ equally spaced intervals $\{[0, D_{t+1}-1], [D_{t+1}, D_{t+2}-1], \cdots, [D_{2t}, N-1]\}$.

In this case, $\mathcal{A}$ learns from $t$ instances of the threshold signature protocol, for $t$ different values $p = t+1, \cdots, 2t$, whether $d$ lies to the left or to the right of each of these $D_p$'s. Consequently $\mathcal{A}$ shrinks the search interval for the secret $d$ from $[0, N-1]$ to some interval $[D_p, D_{p+1}-1]$ which is smaller than the original interval by the factor of $t+1$. If the adversary repeats this attack recursively, then with every share update protocol his search range narrows by the factor of $t+1$. This is equivalent to saying that in every update interval, the adversary learns the $lg(t+1)$ new most significant bits (MSBs) of the secret $d$. Therefore, this search procedure will end and the secret key $d$ will be completely recovered after $\lceil \frac{|N|}{lg(t+1)} \rceil$ share update rounds. We refer to this search procedure as a *"(t+1)-ary search"*.

**Example with $t$=1:** Consider a simple example with $t = 1$. Assume that the adversary $\mathcal{A}$ compromises member $M_1$. Assume also that $M_1$ collaborates with member $M_2$ in a threshold signature protocol. The signing group is therefore $G_2 = \{1, 2\}$, which yields the

following equation:

$$d = d_1^{(G_2)} + d_2^{(G_2)} - \alpha^{(G_2)} N$$

Here $S_2 = D_2 = d_1^{(G_2)}$. $\mathcal{A}$ now employs the *t-bounded offsetting algorithm* and learns the value of $\alpha^{(G_2)}$. If $\alpha^{(G_2)} = 0$, $\mathcal{A}$ learns that $d \geq d_1^{(G_2)} \pmod N$; otherwise if $\alpha^{(G_2)} = 1$, he learns that $d < d_1^{(G_2)} \pmod N$. Assuming that in the share update procedure $\mathcal{A}$ can pick his secret share $ss_1$ so that the resulting $D_2 = d_1^{(G_2)}$ is whatever $\mathcal{A}$ wants, $\mathcal{A}$ can set $D_2 = \lceil \frac{N-1}{2} \rceil$. Then, with every share update round, $\mathcal{A}$ halves the search interval, and thus he performs a *binary search* which recovers the secret $d$ completely in $lg(N)[= \frac{|N|}{lg(2)}]$ rounds.

**Attack Speed-ups:** Since in many cases the RSA secret key $d$ can be efficiently recovered once some number of the most significant bits are recovered, the number of rounds in the attack can be further reduced. Moreover, for the commonly used small values of $e$, like $e = 3$, 17, or 65537, the attack can be sped-up by the factor of two because the first few MSB bits of $d$ enable $\mathcal{A}$ to efficiently compute the first half of the MSBs of $d$. We describe such speed-up mechanisms in subsection 6.3.4 below.

### 6.3.2 Optimal Choice of New Secret Shares

In each share update protocol the adversary's goal is to set the $t$ values $D_{t+1}, \cdots, D_{2t}$ which will hold in the subsequent update period in a manner described above. In order to do that, the adversary first solves for the optimal new secret shares $ss_1, ss_2, \cdots, ss_t$ which would result in values $D_{t+1}, \cdots, D_{2t}$ he desires, using the following system of $t$ modular linear equations:

$$ss_1 l_1^{(G_{t+1})} + ss_2 l_2^{(G_{t+1})} + \cdots + ss_t l_t^{(G_{t+1})} = D_{t+1} \pmod N$$
$$ss_1 l_1^{(G_{t+2})} + ss_2 l_2^{(G_{t+2})} + \cdots + ss_t l_t^{(G_{t+2})} = D_{t+2} \pmod N$$
$$\cdots\cdots$$
$$\cdots\cdots$$
$$ss_1 l_1^{(G_{2t})} + ss_2 l_2^{(G_{2t})} + \cdots + ss_t l_t^{(G_{2t})} = D_{2t} \pmod N$$

These equations are linearly independent as the following matrix $\mathbb{L}$ is invertible:

$$\mathbb{L} = \begin{pmatrix} l_1^{(G_{t+1})} & l_2^{(G_{t+1})} & \cdots & l_t^{(G_{t+1})} \\ l_1^{(G_{t+2})} & l_2^{(G_{t+2})} & \cdots & l_t^{(G_{t+2})} \\ \cdots & & & \\ l_1^{(G_{2t})} & l_2^{(G_{2t})} & \cdots & l_t^{(G_{2t})} \end{pmatrix} = (-1)^t \prod_{i=1}^{t} \left(\frac{\prod_{j=1, j\neq i}^{t}(-j)}{\prod_{j=1, j\neq i}^{t}(i-j)}\right) \begin{pmatrix} \frac{1}{t} & \frac{1}{t-1} & \cdots & 1 \\ \frac{1}{t+1} & \frac{1}{t} & \cdots & \frac{1}{2} \\ \cdots & & & \\ \frac{1}{2t-1} & \frac{1}{2t-2} & \cdots & \frac{1}{t} \end{pmatrix}$$

Thus, by inverting the above matrix the adversary can compute the optimal secret share values $ss_1, \cdots, ss_t$ he needs in the next update round, in order to achieve his optimal values $S_{t+1}, \cdots, S_{2t}$ in the signature protocols performed in that update round. The adversary is now left with the task of forcing the proactive update protocol to actually arrive at these optimal secret shares $ss_1, \cdots, ss_t$ for the corrupted members $M_1, \cdots, M_t$.

### 6.3.3 Adversarial Behavior in the Update

We show that as long as the adversary controls the players in the update group $\Omega$, the adversary can easily influence the proactive share update protocol to arrive at the optimal shares $ss_1, \cdots, ss_n$ he computed above. In the case of larger $\Omega$ groups, the attack succeeds as long as the adversary corrupts $t$ members in $\Omega$, and as long as some of these members can "speak last" in the first phase of the URSA share update protocol (see section 6.2.3).

Let us describe the attack assuming the most general case that $|\Omega| \geq t$. Let $B \subseteq \Omega$ denotes the subset of $t$ members corrupted by the adversary, and let $M_b \in B$ be the corrupted member who "speaks last" in the first phase of the share update protocol. Since there is no established sequence or order in which the members in $\Omega$ take part in the secret share update procedure, the adversary can wait until each member in $\Omega$ except of $M_b$ distributes their shares of the random update polynomials $\delta_j$, $j \in \Omega \setminus \{M_b\}$, before distributing the shares of his polynomial $\delta_b(z)$ as the last one. (If the order is somehow fixed, although it's not clear how it could be without heavy performance penalty for the protocol, the adversary would still win assuming that he corrupts the player who is entitled to speak last.) Recall that the update polynomial is equal to

$$\delta(z) = \sum_{j \in \Omega \setminus \{M_b\}} \delta_j(z) + \delta_b(z) \pmod{N}$$

and that the new shares of each members are computed as $ss_i = ss'_i + \delta(i)$ where $ss'_i$ is the current share of $M_i$.

To fix the resulting shares of the corrupted members to come out as the optimal

values $ss_1, \cdots, ss_t$ specified above, member $M_b$ chooses his partial update polynomial $\delta_b(z)$ in such a way that the resulting update polynomial $\delta(z)$ satisfies $\delta(i) = ss_i - ss'_i \pmod N$ for $i = 1, \cdots, t$. To do that, $M_b$ sets values $\delta_b(i)$ for $i = 1, \cdots, t$ as

$$\delta_b(i) = ss_i - ss'_i - \sum_{j \in G \setminus \{M_b\}} \delta_j(i) \pmod N$$

The $M_b$ player then interpolates these values to recover the $\delta_b(z)$ update polynomial he should use.

Importantly, note that this adversarial behavior is indistinguishable to outside observers from prescribed behavior an honest player exhibits in the protocol. The attack succeeds if $M_b$ picks his partial update protocol in the above way instead of the prescribed way of picking this polynomial at random, but the difference cannot be observed by the honest players, and thus this attack would be undetected.

### 6.3.4 Speeding-up the Attack

By following the above attack procedure, in every update round the adversary $\mathcal{A}$ learns new $lg(t+1)$ most significant bits (MSBs) of $d$. Assuming that $\mathcal{A}$ needs to discover all the $|N|$-bits of the RSA secret key $d$, $\mathcal{A}$ needs $\lceil \frac{|N|}{lg(t+1)} \rceil$ update rounds, and $t$ signature protocol instances within each update interval as described above, to complete the attack. However, there are several ways in which $\mathcal{A}$ can speed up this search. First, we can assume that at least the last 40-bits of the secret $d$ can be obtained by a brute-force search once all the other bits are found, because the candidate $d$ can be efficiently tested given the public key $(e, N)$. This reduces the number of rounds in the attack to $\lceil \frac{1}{lg(t+1)}(|N| - 40) \rceil$. Second, we can speed up this search by making a simple observation about half of the MSBs of $d$ for small $e$ values, and by utilizing several known results regarding the security of the RSA cryptosystem under partial key exposure [BDF98, BM03]. Below we explain the speed up for small $e$'s, and we list the other applicable results and explain how they speed up our search algorithm. The graph in Figure (6.1) summarizes this discussion by showing the number of rounds required in the attack w.r.t the range of the public exponent $e$ for 1024-bit RSA modulus $N$, taking as an example a threshold value $t = 7$.

**Theorem 4.** *Given only the first $log_2(e)$ MSBs of $d$, the first half of MSBs of $d$ can be efficiently computed.*

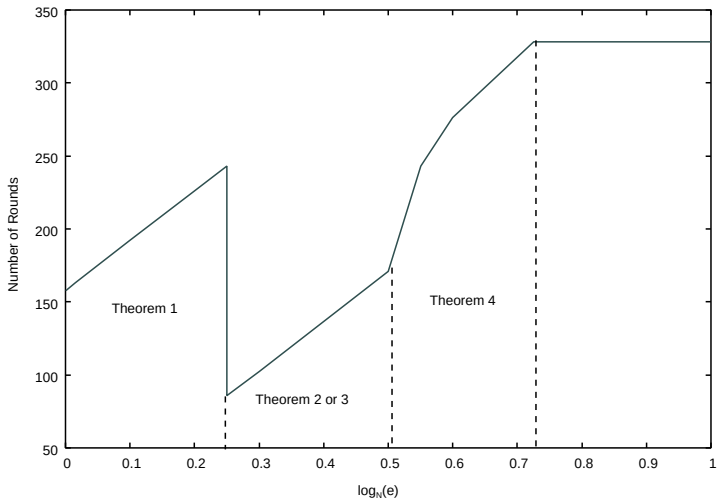



Figure 6.1: Currently required # of proactive update rounds to recover $d$ for a given value of $log_N(e)$, assuming $|N| = 1024, t = 7$.

*Proof.* Note that $ed = 1 \pmod{\phi(N)}$ implies that $d = 1/e(1 + k\phi(N))$ for some integer $k = 1, \cdots, e-1$. Therefore, since $N - \phi(N) < \sqrt{N}$, it follows that $0 \leq \hat{d}_k - d < \sqrt{N}$ for $\hat{d}_k = \lfloor 1/e(1+kN) \rfloor$ for one of the $e-1$ choices of $k$. Note that the $\hat{d}_k$ values can be publicly computed, and note that $\hat{d}_{k+1} - \hat{d}_k \approx N/e$ for every $k$, and therefore that the $log_2(e)$ MSBs of $d$ determine the appropriate $k$ (and $\hat{d}_k$) values, and therefore, since $|\hat{d}_k - d| < \sqrt{N}$ for that $k$, they also determine the 512 MSBs of $d$. □

The import of the above observation for our attack is very simple: Since the above choices of the $e-1$ values $\hat{d}_1, \cdots, \hat{d}_{e-1}$, are neatly spread in the $[0, N-1]$ interval in distances of $N/e$ apart from each other, in our attack based on the $(t+1)$-ary search, the adversary can identify the appropriate $\hat{d}_k$ (and $k$) value, and thus recover the first $|N|/2$ MSBs of $d$ by the above theorem, after just $\lceil \frac{lg(e)}{lg(t+1)} \rceil$ rounds of the share update protocol.

Therefore, for small $e$'s, the attack requires only

$$r > \frac{1}{lg(t+1)} \left( lg(e) + \frac{|N|}{2} - 40 \right) \tag{6.3}$$

rounds of share updates for the adversary to learn the whole secret $d$. This means that the current implementation of URSA which uses the well-known value $e = 65537$ [KZL+01] can be attacked in just 163 update rounds for a modest threshold of $t = 7$.

For larger values of $e$, the results of [BDF98, BM03] on the RSA key security with partial key exposure imply the following speed-ups in our RSA key recovery attack:

**Theorem 5.** [BDF98] *If $e$ is a prime in the range $[2^m, 2^{m+1}]$, with $\frac{|N|}{4} \leq m \leq \frac{|N|}{2}$, then given $m$ MSBs of $d$, there is a polynomial time algorithm to compute $d$.*

**Theorem 6.** [BDF98] *If $e$ is in range $[2^m, 2^{m+1}]$ and is a product of at most $r$ primes, with $\frac{|N|}{4} \leq m \leq \frac{|N|}{2}$, then given the factorization of $e$ and $m$ MSBs of $d$, there is a polynomial time algorithm to compute all of $d$.*

If $e$ meets either of the above criteria, the number of rounds $r$ required in our attack reduces to within the range

$$\lceil \frac{|N|}{4lg(t+1)} \rceil \leq r \leq \lceil \frac{|N|}{2lg(t+1)} \rceil$$

**Theorem 7.** [BM03] *If $e$ is in range $[N^{0.5}, N^{0.725}]$, then the number of MSBs needed to completely recover $d$ is given by*
$\frac{|N|}{8}(3 + 2\alpha + \sqrt{36\alpha^2 + 12\alpha - 15})$ *where $\alpha = log_N(e)$.*

If $e$ meets the above criteria, the number of rounds required to recover the secret key $d$ is given by $\lceil \frac{|N|}{8lg(t+1)}(3 + 2\alpha + \sqrt{36\alpha^2 + 12\alpha - 15}) \rceil$

## 6.4 Discussion: Efficiency of Our Attack and Insecurity of the URSA scheme

While the above results for general $e$ values are interesting, in practice people want to use RSA with small $e$ values, like $e = 3$, 17, or 65537 (all of which are prime numbers of the form $2^i + 1$ for a small value of $i$, which makes the exponentiation $s^e \pmod N$ involved in the RSA signature verification take only $i + 1$ modular multiplications), and for these values our attack holds if (1) $t$ members of the update group $\Omega$ are corrupted and one of the corrupted player speaks last, (2) if in every update interval some $t$ chosen honest players $M_p$ are coaxed into participating in a signature protocol (note that the adversary's attack does not depend on what message is used in this protocol!), and (3) if the system lasts for $r = 163$ update rounds, which given the twice a day rate of updates gives only two months.

We think that these are quite reasonable assumptions. There's certainly nothing in the adversarial model of a proactive signature scheme and of the URSA group access control scheme, which would disallow the adversary from satisfying each of the above criteria.

Since our attack depends crucially on all of the $t$ players in the share update group $\Omega$ to be the corrupted players, it is therefore important for the practical feasibility of the

attack how this $\Omega$ group is decided. From the initial reply of the URSA authors to the attack presented in this chapter [Lu], it appears that the details of how the $\Omega$ group is decided are not set in stone in the design of the URSA scheme. This is not surprising since the idea of modifying the Herzberg et al. protocol by delegating the update work to a smaller set of players $\Omega$ was introduced for the reasons of efficiency, not security.

The fixes proposed in [Lu], e.g. choosing the $\Omega$ group as the $t$ players with smallest IDs, seem problematic: First, such players would need to be identified in some distributed protocol, and second, the resulting protocol would now be still under the attack, but only if the $t$ players with lowest IDs are corrupted. Moreover, the resulting protocol would then need all these $\Omega$ players to be always present and connected, which goes against the URSA philosophy of providing group access control in environments where some players can become inactive and disconnected. Possibly, the tweak that would slow the attack we describe in this chapter the most would choose the $\Omega$ group differently in every round. This tweak, however, has similar problems: (1) It is not clear how to make sure that the $\Omega$ membership rotates without the global knowledge of the current membership list, which would reduce the applicability of the resulting protocol, and (2) the resulting protocol would again need the scheduled players to be up and connected at the right times. In our understanding, the attractive idea of the original philosophy of URSA was that the $\Omega$ group can be formed by *any* $t$ players, and moreover that several such groups can be independently created in disconnected fragments of the network.

Unfortunately, while the ability for any group of $t$ currently active and connected players to form the $\Omega$ update group would make the URSA scheme most reliable and attractive, because of the security vulnerability of this scheme which we describe in this chapter, such freedom in choosing $\Omega$ would also lead to the fastest key-recovery attack on the resulting scheme. While it is still possible that there exists a smart tweak of the $\Omega$-choosing process which would both slow down in practice the attack we describe here, and would not impact the applicability of the URSA scheme to the "on/off presence, on/off communication links" setting it targets, we believe that what is really needed is a replacement of the URSA proactive RSA scheme with a *provably secure* proactive signature scheme.

The attack can be probably slowed down in practice if some modifications are employed in the process of choosing the "update group" $\Omega$, to make it harder for the adversary to corrupt this group, or, equivalently, to make it harder for the adversary to have whatever players he does corrupt be chosen as the $\Omega$ group. However, summing up the

above discussion, it is not clear how to modify the protocol in order to make the adversary's success significantly harder, and at the same time not to severely limit the applicability of the URSA scheme.

Perhaps more importantly, the attack we exhibit shows that even if the adversary does not satisfy all the above criteria, the adversary still learns meaningful information about the RSA private key $d$, which makes the security of the resulting system doubtful. For example, as we discussed in the introduction, if the adversary successfully participates in just $r = 34$ instead of $r = 163$ update rounds (for 1024-bit $N$, $e = 65537$ and $t = 7$), the adversary will learn 600 most significant bits of $d$. Given the steady progress in the ability to recover the full $d$ from partial knowledge [BDF98, BM03], there is little hope that such partial information can be shown not to weaken the RSA system.

Finally, our attack uses only very specific *part* of the information leaked in the URSA threshold RSA signature protocol. We showed that if the adversary corrupts $t$ of the signing $t+1$ players, and if the remaining $(t+1)$-th player $M_p$ can be known beforehand, the leaked information is equivalent to whether the shared secret $d$ lies to the left or to the right of some value $D_p$ which can be computed and in fact controlled by the adversary. However, information about $d$ leaks also if (1) other players $M_p$, $p \notin [t+1, \cdots, 2t]$ participate in the threshold signature protocol, and if (2) less than $t$ corrupted players participate in this protocol. The information revealed about $d$ in these cases is more complicated than the $d <$ or $> D_p$ information we used in our attack, but it is nevertheless easy to define as well, and it can very well be used to either speed up the attack we propose even more, or to extend it to adversaries who (1) corrupt less than $t$ (e.g., $t-1$ as in the non-standard model of [LKZ+04]) the players participating in the threshold signature protocol, or (2) corrupt less than $t$ players in the update group $\Omega$, or (3) fail to predict which honest players will participate in the threshold signature protocol.

## 6.5 The New Proactive RSA Signature Scheme

In this section, we build a new proactive RSA signature scheme based on a corrected use of the URSA proactive RSA.

### 6.5.1 Computational and Adversarial Model

We work in the standard model of threshold cryptography and distributed algorithms known as synchronous, secure links, reliable broadcast, trusted dealer, static, and proactive adversary model. This is the same model as employed for example in [HJKY95, HJJ+97b, FGMY97b, FGMY97a, Rab98] discussed in the introduction, with the exception that the first two did not need a trusted dealer (but did not handle RSA).

This model involves $n$ players $M_1, ..., M_n$ equipped with synchronized clocks and an ability to erase information. The players are connected by weakly synchronous communication network offering secure point-to-point channels and a reliable broadcast. The time is apriori divided into evenly spaced update rounds, say of length of one day. We assume the presence of the so-called "mobile" adversary, modeled by a probabilistic polynomial time algorithm, who can *statically*, i.e., at the beginning of the life time of the scheme, schedule up to $t < n/2$ arbitrarily malicious faults among these $n$ players, independently for every update round. We also assume a trusted dealer who initializes the distributed scheme by picking an RSA key and securely sharing the private key among the players. Since the adversary attacks a proactive *signature* scheme, the adversary can also stage a chosen-message attack [CMA], i.e. it can ask any of the $n$ players to run a signature protocol on any message it chooses. The adversary's goal is to either (1) forge a signature on a message he did not request a signature on, exactly as in the CMA attack against a standard (non-threshold) signature scheme, or (2) to prevent the efficient generation of signatures on messages which at least $t + 1$ uncorrupted players want to sign.

### 6.5.2 Overview of the Proposed Scheme

The sharing of the private RSA key $d$ is done additively modulo a *prime* $q$ s.t. $q \geq r2^{|N|+\tau}$, where $r$ is the maximal number of rounds in the lifetime of the system, $|N|$ is the bit length of the RSA modulus $N$, and $\tau$ is a security parameter, e.g. $\tau = 80$. Namely each player $M_i$ holds a share $d_i$ which is a random number in $\mathbb{Z}_q$ s.t. $d_1 + ... + d_n = d \bmod q$. Each of these top-level additive shares is also polynomially shared for backup reconstruction of $d_i$ in case $M_i$ is corrupted, using the information-theoretically secret verifiable secret sharing (Pedersen-VSS) of Pedersen [Ped91a] (refer to Section 2.7 for details), similarly as in the proactive RSA scheme of Rabin [Rab98]. In order to handle the common case of a small public RSA exponent $e$ more efficiently, the most significant $l = \frac{|N|}{2}$ bits of the private

key $d$ can be publicly revealed as $d_{pub}$, and only the remaining portion of the private key $d$, namely $d - 2^{|N|-l}d_{pub}$, is shared as above modulo $q$, for any $q \geq r2^{|N|-l+\tau}$.

The proactive update is very easy in this setting, adopting the original proactive secret sharing of [HJKY95] to additive sharing. Such method was used before e.g. in [CGJ+99]. To re-randomize the the sharing, each $M_i$ picks random partial shares $d_{ij}$ in $\mathbb{Z}_q$ s.t. $d_i = d_{i1} + ... + d_{in} \bmod q$, and sends $d_{ij}$ to $M_j$. Each $M_j$ computes then his new share as $d'_j = d_{1j} + ... + d_{nj} \bmod q$, and shares it polynomially for backup again. All this can be easily verified using Pedersen-VSS, and the new shares sum to the same secret $d$ modulo $q$.

For the threshold signature protocol, we use the observation (as in the URSA scheme) that if $\sum_{j=1}^{n} d_j = d \pmod q$ and $0 \leq d_j \leq q-1$ for all $j$'s, then

$$d = \sum_{j=1}^{n} d_j - \alpha q \quad \text{(over the integers)} \tag{6.4}$$

for some integer $\alpha \in \{0, ..., n-1\}$. Consequently, if $\forall_j$, $s_j = m^{d_j} \bmod N$ then

$$m^d = (\prod_{j=1}^{n} s_j) m^{-\alpha q} \pmod N$$

Therefore the signature $m^d \bmod N$ can be reconstructed if players submit their partial signatures as $s_j = m^{d_j} \pmod N$, and the correct value of $\alpha$ is publicly reconstructed by cycling over the possible $n$ choices of $\alpha$, which adds at most $2n$ modular exponentiations to the cost of the signature generation protocol. (Note that in most applications $n < 100$.) In the (rare) case of a malicious fault causing a failure in this procedure, each player has to prove in zero-knowledge that it used a correct value $d_i$ in its partial signature, i.e. the value committed in the Pedersen-VSS that shares this $d_i$. Efficient zero-knowledge proofs to handle such statement were given by Camenisch and Michels [CM99a], and Boudot [Bou00], and while not blazing fast, they have constant number of exponentiations, and they are practical. This procedure is more expensive than the robustness procedure in [Rab98], but we believe that this efficiency difference does not matter since active corruptions of this type should be unlikely, as active faults are rare in general and the adversary would not gain much by using his corrupted player in this way.

Recall (from section 6.3) that in our attack on the URSA scheme involving *polynomial* rather than additive top-level sharing, the adversary uses the fact that the above procedure reveals whether $d$ is greater or smaller than some value in the $[0, q]$ interval which

the adversary can easily compute from his shares. Since the adversary can perfectly control his shares in the proactive update protocol for this (top-level) polynomial secret sharing scheme, the adversary can use this partial information leakage to stage a binary search for the shared secret $d$.

However, the scheme we present fixes the above problem. Assume that the adversary corrupts players $M_1, ..., M_t$. Giving the adversary the extra knowledge of shares $d_{t+1}, ..., d_{n-1}$, the only information about the secret key revealed by value $\alpha$ is, by equation (6.4), whether or not the secret $d$ is smaller or larger than $R = (D \bmod q)$ where $D = d_1 + ... + d_{n-1}$. Since the adversary does not have enough control over the shares created by our "additive" proactive update protocol, shares $d_{t+1}, ..., d_{n-1}$ are random in $\mathbb{Z}_q$, and hence so is value $R$. Therefore, if $q$ is significantly larger than the maximal value of $d$, then the $\alpha$ value almost never reveals anything about $d$, because $d$ is almost always smaller than $R$. For this reason, if $q \geq r2^{|N|+\tau}$ then the modified scheme keeps $d$ indistinguishable from a value uniform in $\mathbb{Z}_n$, with the statistical difference of $2^{-\tau}$. The additional factor $r$ in the bound on $q$ appears because of the linear increase in the statistical difference with every update round. This captures the security proof of our scheme in a nutshell. The security and robustness of our new scheme is based on Discrete Logarithm, RSA and Strong RSA assumptions, which we described in Section 2.2.

### 6.5.3 Setup Procedure

We require a trusted dealer to securely set up the system. The dealer generates RSA private/public key pair, i.e. an RSA modulus $N$, public exponent $e$, and private key $d = e^{-1} \bmod \phi(N)$. Optionally, $l \leq \frac{|N|}{2}$ most significant bits of $d$ can be publicly revealed as $d_{pub}$ (otherwise $d_{pub} = 0$ and $l = 0$). The dealer also chooses an instance of Pedersen commitment [Ped91a], i.e. primes $p$ and $q$ s.t. $q|(p-1)$, and two random elements $g, h$ of order $q$ in $\mathbb{Z}_p^*$, for $|q| = \log_2 r + |N| - l + \tau + 1$, where $\tau$ is a security parameter ($\tau \geq 80$) and $r$ is the number of rounds the system is expected to run. The dealer then runs the sharing protocol of Figure 6.2.

### 6.5.4 Threshold Signature Protocol

The goal of the threshold RSA signature protocol is to generate in a distributed manner an RSA signature $s = m^d \pmod N$ under the secret-shared key $d$, where $m \in \mathbb{Z}_n^*$ is

**Input**: private key $d \in \mathbb{Z}_{\phi(n)}$, public value $d_{pub}$ corresponding to $l$ MSBs of $d$, public RSA modulus $N$, Pedersen commitment instance $(p, q, g, h)$.

1. Select shares $d_j \in \mathbb{Z}_q$ uniformly at random for $j = 1, \ldots, n-1$ and set $d_n = d - 2^{|N|-l} d_{pub} - \sum_{j=1}^{n-1} d_j \pmod{q}$.

2. Share each $d_j$ using Pedersen-VSS protocol [Ped91a]. Namely, select random polynomials $f_j(z) = d_j + f_{j1}z + \cdots + f_{jt}z^t$ and $f'_j(z) = d'_j + f'_{j1}z + \cdots + f'_{jt}z^t$ over $\mathbb{Z}_q$ of degree $t$ s.t. $f_j(0) = d_j$. Compute and publish the witnesses $w_{j0} = g^{d_j} h^{d'_j} \pmod{p}$ and $w_{jk} = g^{f_{jk}} h^{f'_{jk}} \pmod{p}$ for $k = 1, \ldots, t$.

3. Compute the secret shares $ss_{ij}$ and $ss'_{ij}$ as $ss_{ij} = f_j(i) \pmod{q}$ and $ss'_{ij} = f'_j(i)$, deliver $d_i, d'_i, ss_{ij}$ and $ss'_{ij}$ $(\forall j)$ to each $M_i$ over a secure channel.

Figure 6.2: Trusted Dealer's Protocol: Sharing of the Private Key $d$

some hashed/padded function of the signed message, e.g. $m = H(M)$ for the Full Domain Hash RSA [BR93]. Our protocol consists of two parts. First each player $M_j$ creates its *partial signature* on the intended message $s_j = m^{d_j} \bmod N$, and sends it to the signature recipient. The recipient then locally reconstructs the RSA signature from these partial signatures using the $n$-bounded reconstruction algorithm of [LL00]. The threshold signature generation and reconstruction protocol is summarized in Figure 6.3, and we explain the details of the reconstruction algorithm below.

**Signature Reconstruction with $n$-Bounded Offsetting.** On receiving $n$ partial signatures $s_j$ from the $n$ players, the signature recipient reconstructs the RSA signature $s$ using the *$n$-bounded-offsetting* algorithm [KZL+01] which works as follows. Since $\sum_{j=1}^{n} d_j = d - 2^{|N|-l} d_{pub} \pmod{q}$ and $0 \le d_j \le q-1$ for all $j$'s, therefore

$$d = 2^{|N|-l} d_{pub} + \sum_{j=1}^{n} d_j - \alpha q \quad \text{(over the integers)} \tag{6.5}$$

for some integer $\alpha \in \{0, \ldots, n-1\}$, which implies that

$$s = m^d = m^{2^{|N|-l} d_{pub}} (\prod_{j=1}^{n} s_j) m^{-\alpha q} \pmod{N}$$

Since there can be at most $n$ possible values of $\alpha$, the signature recipient can recover $s = m^d \pmod{N}$ by trying each of the $n$ possible values $Y_\alpha = Y(m^{-q})^\alpha \pmod{N}$ for

**Input**: (hashed) message $m \in \mathbb{Z}_n^*$, outputs of the Setup procedure

1. Player $M_i$ broadcasts its partial signature $s_i = m^{d_i} \pmod N$.
2. If $M_i$ fails to provide its partial signature, all players reconstruct $d_i$ and compute $s_i = m^{d_i} \pmod N$.
3. Reconstruct RSA signature using the *n-bounded offsetting* algorithm (see below).
4. If signature reconstruction fails, trace the faulty signer(s) by executing the protocol $ZKPK(d_i : w_{i0} = g^{d_i} h^{d'_i} \pmod p \ \wedge \ s_i = m^{d_i} \pmod N) \ \wedge \ d_i \in [0, q-1]))$ with each $M_i$ (see Appendix A).
5. If $M_i$ fails this proof, any set of $t+1$ players reconstruct $d_i$ and compute and broadcast $s_i = m^{d_i} \pmod N$.

Figure 6.3: Signature Generation and Reconstruction

$Y = m^{2^{|N|-l} d_{pub}} (\prod_{j=1}^{n} s_j)$ and $\alpha = 0, ..., n-1$, and returning $s = Y_\alpha$ if $(Y_\alpha)^e = m \pmod N$. The decisive factor in the cost of this procedure is the cost of the full exponentiation $m^q \bmod N$, where $q$ can be e.g. 613-bit long for $N = 1024$, $e = 3$, $l = |N|/2$, $\tau = 80$, and $r \leq 2^{20}$.

As discussed in the overview subsection above, this procedure reveals value $\alpha$ which contains some partial information on the shared secret $d$. Namely, granting to the adversary some extra knowledge and assuming he knows shares $d_1, ..., d_{n-1}$, the $\alpha$ value reveals whether $d \in \mathbb{Z}_{\phi(n)}$ lies in the interval $[0, R[$ or in $[R, N]$, where $R = (D \bmod q)$ and $D = d_1 + \ldots + d_{n-1}$, if $l = 0$. More generally, $\alpha$ reveals if $d$ is smaller or larger than $R + 2^{|N|-l} d_{pub}$.

**Robustness Mechanisms.**

In case some player $M_u$ does not issue a partial signature, share $d_u$ of $M_u$ needs to be reconstructed to recover partial signature $s_u = m^{d_u} \pmod N$. In reconstruct $d_u$, every player $M_i$ broadcasts its shares $ss_{iu}, ss'_{iu}$ of $d_u$. The validity of these shares can be ascertained by checking

$$g^{ss_{iu}} h^{ss'_{iu}} = \prod_{k=0}^{t} (w_{uk})^{i^k} \pmod p.$$

Share $d_u$ can then be recovered using the interpolation

$$d_u = \sum_{j \in G} ss_{ju} l_j(u) \pmod{q}$$

where $G$ is a subgroup of $t+1$ players who broadcast valid shares and $l_j(u) = \prod_{j \in G, j \neq i} \frac{(u-j)}{i-j} \bmod q$ is the Lagrange interpolation polynomial computed at $u$.

If all the partial signatures are present but the above $n$-bounded signature reconstruction algorithm fails, then at least one out of $n$ players did not issue a correct partial signature. The signature recipient must then trace the faulty players(s) by verifying the correctness of each partial signature. Once a player is detected as faulty, the share(s) of the faulty player(s) can be reconstructed as above. To prove correctness of its partial signature, each $M_i$ proves in zero-knowledge that there is a pair of integers $(d_i, d_i')$ s.t.

$$w_{i0} = g^{d_i} h^{d_i'} \bmod p \quad , \quad s_i = m^{d_i} \bmod N \quad , \quad 0 \leq d_i < q$$

It is crucial that the range of $d_i$ is checked because otherwise player $M_i$ can submit its partial signature as $m^{d_i'} \bmod N$ where $d_i' = d_i + kq$ for some $k$. An efficient zero-knowledge proof system for the proof of equality of discrete logarithms (and representations) in two different groups was given in [BT99, CM99b], and the efficient proof that a committed number lies in a given range appeared in [Bou00]. The resulting ZKPK proof system is shown in Appendix A. It is non-interactive in the random oracle model and involves a (small) constant amount of exponentiations.

### 6.5.5 Proactive Update Protocol

At the beginning of every update round, the players perform the share update protocol of Figure 6.4 to re-randomize the sharing of $d$.

## 6.6 Security Analysis of the New Proactive RSA Scheme

**Theorem 8 (Security).** *If there is a $t$-threshold proactive adversary for $t < n/2$, which in time $T$ succeeds with probability $\beta$ in a chosen-message attack against our new proactive (full domain hash) RSA signature scheme running for up to $r$ rounds, for any $l \leq |N|$ and prime $q \geq r2^{|N|-l+\tau}$, then there is a CMA attack against the standard (full domain hash) RSA signature scheme, which succeeds in time $T + poly(n, |N|)$ with probability $\beta - 2^{-\tau}$ given the $l$ most significant bits of the secret key $d$ as an additional public input.*

**Input:** Outputs of the Setup procedure or the previous Update protocol.
Let $r \geq 1$ be the round number. Denote current values $d_i^{(r-1)}, d_i'^{(r-1)}, w_{ij}^{(r-1)}$, etc.

1. Each player $M_i$ selects (sub)shares $d_{ij}$ and $d'_{ij} \in \mathbb{Z}_q$, uniformly at random for $j = 1, \ldots, n-1$, and sets $d_{in} = d_i^{(r-1)} - \sum_{k=1}^{n-1} d_{ik} \pmod q$ and $d'_{in} = d_i'^{(r-1)} - \sum_{k=1}^{n-1} d'_{ik} \pmod q$. $M_i$ broadcasts witness values $w_{ij}^{(r)} = g^{d_{ij}} h^{d'_{ij}} \pmod p$, and hands $(d_{ij}, d'_{ij})$ to $M_j$ $(\forall j)$ over a secure channel.

2. $M_j$ verifies the validity of the received shares using witness values as $w_{ij}^{(r)} = g^{d_{ij}} h^{d'_{ij}} \pmod p$, and ascertains whether the sub-shares in fact sum up to the previous share of $M_i$ by checking that $\prod_{j=1}^{n} w_{ij}^{(r)} = w_{i0}^{(r-1)} \pmod p$.

3. $M_j$ computes its new additive shares as $d_j^{(r)} = \sum_{i=1}^{n} d_{ij} \pmod q$ and $d_j'^{(r)} = \sum_{i=1}^{n} d'_{ij} \pmod q$. (Note that $\sum_{j=1}^{n} d_j^{(r)} = d - 2^{|N|-l} d_{pub} \pmod q$.)

4. $M_j$ shares its new additive shares $d_j^{(r)}, d_j'^{(r)}$ using Pedersen-VSS, as in the setup phase described in Section 6.5.3. In order to check if $M_j$ is indeed sharing its new additive share, every player checks that the witness value in this VSSinstance corresponding to the shares $d_j^{(r)}, d_j'^{(r)}$ equals to $\prod_{i=1}^{n} w_{ij}^{(r)} \pmod p$.

Figure 6.4: Proactive Share Update

*Proof.* We show that if the adversary succeeds in staging the CMA attack on our (Full Domain Hash) proactive RSA signature scheme in time $T$ with probability $\beta$, then there is also an efficient CMA attack against the standard (non-threshold) FDH-RSA signature which given the $l$ most significant bits of $d$ succeeds in time comparable to $T$ by an amount polynomial in $|N|$ and $n$, with probability no worse than $\beta - 2^{-\tau}$. We show it by exhibiting a very simple simulator, which the adversary against the standard FDH-RSA scheme can run to interact with the proactive adversary which $(T, \beta)$-succeeds in attacking the proactive scheme. We will argue that the statistical difference between the view presented by this simulator on input of the public RSA parameters, $l$ MSBs of $d$, and (message,signature) pairs acquired by the CMA attacker from the CMA signature oracle, and the adversarial view of the run of the real protocol on these parameters, for any value of the private key $d$ with these $l$ most significant bits, is at most $2^{-\tau}$, which will complete the proof.

The simulator $\mathcal{SIM}$ is described in Figure 6.5. The simulation procedure is very simple. The simulator picks a random value $\hat{d}$ in $\mathbb{Z}_n$ with the given $l$ most-significant bits, and runs the secret-sharing protocol in the setup stage using this $\hat{d}$. Similarly in every update, the simulator just runs the actual protocol, but on the simulated values which we denote $\hat{d}_i$, $\hat{d}_{ij}$, etc. The only deviation from the protocol is that in the simulation of the threshold signature protocol, assuming w.l.o.g. that the $M_n$ is an uncorrupted player, the simulator runs the actual protocol for all uncorrupted players except of $M_n$, i.e. it outputs $\hat{s}_j = m^{\hat{d}_j}$ for each uncorrupted $M_j$, $j \neq n$. The simulator then determines the $\hat{\alpha}$ value, which is an approximation to the actual value $\alpha$ the adversary would see in the protocol, by computing $D = \sum_{j=1}^{n-1} \hat{d}_j$, and taking $\hat{\alpha} = \lfloor D/q \rfloor + 1$. In this way we have $D = (\hat{\alpha} - 1)q + R$ where $R = (D \bmod q)$. Finally, the simulator computes the missing partial signature $\hat{s}_n$ corresponding to the player $M_n$ as $\hat{s}_n = s * m^{\hat{\alpha} q} / (m^{2^{|N|-l} d_{pub}} \prod_{j=1}^{n-1} \hat{s}_j) \pmod N$. In this way, partial signatures $\hat{s}_j$ add up to a valid RSA signature $\hat{s}$, and value $\hat{\alpha}$ the adversary sees in the *simulation* of the signature reconstruction algorithm is equal to the above $\alpha$ with an overwhelmingly high probability.

For ease of the argument, assume that the adversary corrupts players $M_1, ..., M_t$ throughout the lifetime of the scheme. We will argue that the adversarial views of the protocol and the simulation are indistinguishable with the statistical difference no more than $2^{-\tau}$, even if the adversary additionally sees shares $d_{t+1}, ..., d_{n-1}$ *and* the shared secret key $d$.

**Input**: Pedersen commitment instance $(p, q, g, h)$, RSA public parameters $(N, e)$, optional values $l > 0$ and $d_{pub} < 2^l$ (otherwise set $l = d_{pub} = 0$).

**Setup Procedure**

Pick random $\hat{d} \in \mathbb{Z}_n$ and proceed as in the Setup of the actual protocol:

1. Select random shares $\hat{d}_j, \hat{d}'_j \in \mathbb{Z}_q$, for $j = 1, \ldots, n-1$, and set $\hat{d}_n = \hat{d} - 2^{|N|-l} d_{pub} - \sum_{i=1}^{n-1} \hat{d}_j \pmod{q}$, as in step 1 of the Setup procedure.

2. Share each $\hat{d}_j$ and $\hat{d}'_j$ using the Pedersen-VSS: Choose random polynomials $\hat{f}_j(z) = \hat{d}_j + \hat{f}_{j1} z + \cdots + \hat{f}_{jt} z^t$ and $\hat{f}'_j(z) = \hat{d}'_j + \hat{f}'_{j1} z + \cdots + \hat{f}'_{jt} z^t$ over $\mathbb{Z}_q$ of degree $t$; compute and publish the witnesses $\hat{w}_{j0} = g^{\hat{d}_j} h^{\hat{d}'_j} \pmod{p}$ and $\hat{w}_{jk} = g^{\hat{f}_{jk}} h^{\hat{f}'_{jk}} \pmod{p}$ for $k = 1, \ldots, t$.

3. Compute the secret shares $\hat{ss}_{ij}$ and $\hat{ss}'_{ij}$ as $\hat{ss}_{ij} = \hat{f}_j(i) \pmod{q}$ and $\hat{ss}'_{ij} = \hat{f}'_j(i)$ and distribute $\hat{d}_i, \hat{d}'_i, \hat{ss}_{ij}$ and $\hat{ss}'_{ij}$ $(\forall j)$ to each $M_i$ over a secure channel.

**Threshold Signature Protocol** (on additional input $(m, s)$, where $s = m^d \bmod N$):

1. Generate partial signatures $\hat{s}_i$ for $i = 1, \ldots, n-1$ as $\hat{s}_i = m^{\hat{d}_i} \pmod{N}$. Compute $D = \hat{d}_1 + \ldots + \hat{d}_{n-1}$, and $\hat{\alpha} = \lfloor D/q \rfloor + 1$. Compute $\hat{s}_n = s * m^{\hat{\alpha} q} / (m^{2^{|N|-l} d_{pub}} \prod_{j=1}^{n-1} \hat{s}_j) \pmod{N}$.

2. Output values $\hat{s}_i$ on behalf of the uncorrupted players $M_i$.

3. If needed, execute the ZKPK proof for $M_i \neq M_n$, and simulate it for $M_n$.

**Proactive Update**

Proceed in exactly the same manner as the Proactive Update protocol:

1. At the beginning of round r, for all uncorrupted players $M_i$, select (sub)shares $\hat{d}_{ij}$ and $\hat{d}'_{ij}$ uniformly in $\mathbb{Z}_q$ for $j = 1, \ldots, n-1$, and set $\hat{d}_{in} = \hat{d}_i^{(r-1)} - \sum_{k=1}^{n-1} \hat{d}_{ik} \pmod{q}$ and $\hat{d}'_{in} = \hat{d}'^{(r-1)}_i - \sum_{k=1}^{n-1} \hat{d}'_{ik} \pmod{q}$. Broadcast witness values $\hat{w}_{ij}^{(r)} = g^{\hat{d}_{ij}} h^{\hat{d}'_{ij}} \pmod{p}$, and hand $(\hat{d}_{ij}, \hat{d}'_{ij})$ to $M_j$ $(\forall j)$ over a secure channel.

2. Compute $M_j$'s new secret shares $\hat{d}_j^{(r)} = \sum_{i=1}^{n} \hat{d}_{ij} \pmod{q}$ and $\hat{d}'^{(r)}_j = \sum_{i=1}^{n} \hat{d}'_{ij} \pmod{q}$, as in the Proactive Update protocol.

3. Re-share the new additive share $\hat{d}_j^{(r)}, \hat{d}'^{(r)}_j$ using Pedersen-VSS.

Figure 6.5: Simulator Construction ($\mathcal{SIM}$)

*Setup Procedure*: Since $d_i$ and $d'_i$ in the protocol and $\hat{d}_i$ and $\hat{d}'_i$ in the simulation are all picked uniformly from $\mathbb{Z}_q$ for $i = 1, \ldots, n-1$, the two ensembles $(d, \{d_i, d'_i\}_{i=1,\ldots,n-1})$ and $(d, \{\hat{d}_i, \hat{d}'_i\}_{i=1,\ldots,n-1})$ have identical distributions.

By the information theoretic secrecy of Pedersen-VSS, the second-layer shares and the associated verification values visible to the adversary are also distributed identically in the protocol and in the simulation.

*Threshold Signature Protocol*: Since $d_i$ and $\hat{d}_i$, for $i = 1, \ldots, n-1$, have the identical distributions, therefore distributions of the corresponding partial signatures $s_i$ and $\hat{s}_i$, are also identical. However, values $s_n$ and $\hat{s}_n$ are the same only in the event that value $\alpha$ in the protocol and value $\hat{\alpha}$ in the simulation are the same. Recall that $\hat{\alpha}$ in the simulation is computed as $\hat{\alpha} = \lfloor D/q \rfloor + 1$ where $D = \sum_{j=1}^{n-1} \hat{d}_j$. Note that $D = (\hat{\alpha} - 1)q + R$ where $R = (D \bmod q)$. By equation (6.5), value $\alpha$ computed by the protocol would satisfy equation

$$d = 2^{|N|-l} d_{pub} + D + d_n - \alpha q = 2^{|N|-l} d_{pub} + R + d_n + (\hat{\alpha} - \alpha - 1)q$$

because $d_1, ..., d_{n-1}$ are distributed identically to $\hat{d}_1, ..., \hat{d}_{n-1}$.

Since $d_n$ and $R$ are elements in $\mathbb{Z}_q$ for $q \geq 2^{|N|-l+\tau+\log r}$, and since $d \in [2^{|N|-l} d_{pub}, 2^{|N|-l} d_{pub} + 2^{|N|-l}]$, the above equation implies that there are only two possible cases: $\alpha = \hat{\alpha} - 1$ and $\alpha = \hat{\alpha}$. The first case happens if $d \geq 2^{|N|-l} d_{pub} + R$ and the second if $d < 2^{|N|-l} d_{pub} + R$. However, the probability that $d < 2^{|N|-l} d_{pub} + R$, and hence that $\alpha = \hat{\alpha}$, is at least $1 - 2^{-(\tau+\log r)}$ because the probability of the other case is at most the probability that $R$ is less than $2^{|N|-l}$, which, given that $R$ is a uniformly distributed element in $[0, q]$, is at most $2^{-(\tau+\log r)}$.

Note that value $\alpha$ stays the same in all instances of the threshold signature protocol in any given update round. Since the same holds for the $\hat{\alpha}$ value in the simulation, the probability that the adversary's view of all these protocol instances is different from the view of all the simulation instances remains at most $2^{-(\tau+\log r)}$. In other words, the statistical difference between the adversary's view of the real execution and the simulation in any update round, is at most $(1/r)2^{-\tau}$.

*Proactive Update Protocol*: Since values $\{d_i\}_{i=1..n-1}$ and $\{\hat{d}_i\}_{i=1..n-1}$ are distributed identically, the only difference in the execution and the simulation of the update protocol can come from sharing of the $d_n$ value in the protocol and $\hat{d}_n$ in the simulation. However, since this sharing is a "additive" equivalent of Pedersen-VSS, and the second-layer sharing of the

shares of the $d_n$ or $\hat{d}_n$ value is done with Pedersen-VSS too, the whole protocol hides the shared value $d_n$ perfectly, and hence the adversarial view in the simulation of the update protocol is identical to the adversarial view of the actual protocol.

Since the statistical difference between the protocol and the simulation is zero in the setup stage and in any proactive update stage, and at most $(1/r)2^{-\tau}$ in any single update round, given $r$ rounds the overall difference between adversarial view of the protocol execution and its simulation is at most $2^{-\tau}$, which completes our argument.

□

**Theorem 9 (Robustness).** *Under the Discrete Logarithm and Strong RSA assumptions, our proactive signature scheme is robust against a t-threshold proactive adversary for $t < n/2$.*

*Proof.* Note that the only way robustness can be broken is if some malicious player $M_i$ cheats either in the proactive update protocol, by re-sharing a value different than its proper current share $d_i$ committed in Pedersen commitment $w_i = g^{d_i}h^{d_i} \bmod p$, or $M_i$ cheats in the signature protocol, by proving correct the wrong partial signature $s_i \neq m^{d_i} \bmod N$. Since the first type of cheating is infeasible under the discrete logarithm assumption and the second type is infeasible under the strong RSA assumption, the claim follows. □

### 6.6.1 Security Implications

Taking $l = 0$, Theorem 8 implies that the new proactive signature scheme is as secure as the standard RSA:

**Corollary 1.** *Under the RSA assumption in the Random Oracle Model, our scheme is a secure t-threshold proactive signature, for $l = 0$ and $q \geq r2^{|N|+80}$.*

On the other hand, note that the RSA adversary can always correctly guess the most significant half of the bits of $d$ with probability $1/(e-1)$.[6] Together with theorem 8, this implies the following corollary:

[6]Note that $ed = 1 \pmod{\phi(N)}$ implies that $d = 1/e(1 + k\phi(N))$ for some integer $k = 1, ..., e-1$. Therefore, since $N - \phi(N) < \sqrt{N}$, it follows that $0 \leq \hat{d}_k - d < \sqrt{N}$ for $\hat{d}_k = \lfloor 1/e(1 + kN) \rfloor$ for one of the $e-1$ choices of $k$. Thus any adversary facing the RSA cryptosystem can with probability $1/(e-1)$ guess the $|N|/2$ most significant bits of $d$ by picking the right $k$ and computing $\hat{d}_k$ as above.

**Corollary 2.** *Under the RSA assumption (in the Random Oracle Model), the time $T_{PRSA}$ to break the new proactive signature scheme for $e = 2^i + 1$, $l = |N|/2$ and $q \geq r2^{|N|/2+80}$, is at least $T_{PRSA} \geq 2^{-i}T_{RSA}$, where $T_{RSA}$ is the time required to break the CMA security of the standard (FDH) RSA signature scheme for modulus of length $|N|$.*

For the most popular value of $e = 3$, this implies that if the 1024-bit modulus RSA has a $2^{80}$ security then our proactive RSA scheme running on the same modulus for $l = 512$ and $q \geq r2^{512+80}$ would have at least $2^{79}$ security. For $e = 17$ the provable security would be $2^{76}$. Of course, our scheme could be executed with slightly larger $N$ to compensate for the $2^i$ factor in security degradation, but with key shares sizes still limited by $q < r2^{|N|/2+80}$. The efficiency of the resulting schemes resulting from Corollary 2 should be compared with the straightforward settings implied by Corollary 1, where same $2^{80}$ security is given by 1024 bit $N$ but with larger bound of $r2^{|N|+80}$ on the share size $q$.

However, since there are no known attacks against RSA which speed up the factorization of $N$ when half of the most significant bits of $d$ are revealed for small values of $e$, it can be plausibly hypothesized that for small $e$'s, the proposed proactive RSA scheme remains as secure as standard RSA for the same modulus size even with half of the most significant bits of $d$ are revealed.

Finally we remark that the security analysis of our scheme given in Theorem 8 grants the adversary the knowledge of $n-1$ shares instead of just $t$ shares he can see in the protocol, which suggests that our security analysis can be improved and that our scheme is possibly secure using smaller share sizes than our analysis recommends.

## 6.7 Improved Security Analysis of Rabin's Proactive RSA

**Overview of Proactive RSA Scheme of [Rab98].** During the setup, a trusted dealer generates the RSA public $(N, e)$ and private $(d, \hat{p}, \hat{q})$ key pairs. The signature key $d$ is shared additively among the players. Each $M_i$ gets a share $d_i$, chosen uniformly in $[-R, R]$ where $R = nN^2$, and the dealer publishes public value $d_{public}$ such that

$$d_{public} = d - \sum_{i=1}^{n} d_i \quad (\text{over } \mathbb{Z}) \tag{6.6}$$

This can be easily extended, so that like our new scheme, $l$ most significant bits of $d$ are publicly revealed and added to the $d_{pub}$ value, and only the remaining $(|N| - l)$-bit value

$d - 2^{|N|-l} d_{pub}$ is shared as above. The witness value $w_i = g^{d_i} \pmod N$ corresponding to each $d_i$ is published, where $g$ is an element of high order in $\mathbb{Z}_n^*$. Each share $d_i$ is then itself shared using the Feldman VSS[Fel87] over $\mathbb{Z}_n$. To sign a message $m$, each player $M_i$, generates a partial signature $s_i = m^{d_i} \pmod N$. Since the signature key $d$ is shared over integers, the RSA signature can be easily reconstructed by simply multiplying $n$ partial signatures, i.e.,

$$s = m^{d_{public}} \prod_{i=1}^{n} s_i \pmod N$$

The detection of faults during the signing process can be performed using the protocols of [GJKR96a, FGY96]. The secret share of the faulty player is then reconstructed by pooling in the shares of any $t+1$ players using a special variant of polynomial interpolation (refer to [Rab98] for details). In the share update protocol each $M_i$ additively re-shares its secret share $d_i$ with (sub)shares $d_{ij} \in [-R/n, R/n]$ and

$$d_{i,public} = d_i - \sum_{j=1}^{n} d_{ij} \text{ (over } \mathbb{Z})$$

is made a public value. The new secret share for $d_i^{(r)}$ of $M_i$ is then computed as $d_i^{(r)} = \sum_{j=1}^{n} d_{ji}$, and $M_i$ shares it using Feldman VSSover $\mathbb{Z}_n$.

**Improved Security Analysis and Improved Performance.** First, we note that the simulator for the setup phase presented in [Rab98] has a small error. That simulator for the key distribution protocol picks random shares $\hat{d}_i \in [-R, R]$, for $i = 1, \ldots, n-1$, and it picks $\hat{d}_{public}$ *uniformly* at random in $[-nR, nR + N]$. However, values generated in this way are not statistically indistinguishable from the values in the protocol, because if the $d_i$ values are chosen uniformly in $[-R, R]$, then by equation (6.6), value $d_{public}$ has a normal probability distribution, which is immediately *distinguishable* from the uniform distribution of $\hat{d}_{public}$.

The corrected simulation of the key distribution (and the subsequent update protocols) works exactly in the same manner as the actual protocol. The simulator should choose some secret value $\hat{d} \in [0, N-1]$ at random, and share this new value in exactly the same manner as in the protocol. After $r$ update rounds, the overall statistical difference between the view of the adversary interacting with the protocol and the view of the adversary interacting with the (new) simulator is at most $rN/R$. This difference is negligible if $R = rN2^{\tau}$, where $\tau \geq 80$, instead of the $R = nN^2$ value recommended in [Rab98].

This shows that secret shares can be picked from range $[-rN2^{\tau}, rN2^{\tau}]$, instead of range $[-nN^2, nN^2]$ of the original scheme, which means an almost factor of 2 improvement in the share size. Since the computational cost of this scheme is driven by cost of the exponentiation $s_i = m^{d_i} \bmod N$ done by each player, factor of 2 improvement in the size of $d_i$ speeds the signature generation by the same factor.

# Chapter 7

# Decentralized Admission in Ad Hoc Groups

*In this chapter, we present a secure, efficient and a fully non-interactive protocol to securely extend an ad hoc group. The main feature of our protocol is that it is non-interactive and requires only a single round of asynchronous communication. The protocol is developed using secret sharing techniques based on bi-variate polynomials. We evaluate the proposed protocol in a real MANET setting and show that it compares favorably to previously proposed protocols.*

## 7.1 Introduction

Ad hoc groups, such as mobile ad hoc networks (MANETs), have many well-known applications in military settings as well as in emergency and rescue operations. However, lack of infrastructure and lack of centralized control make ad hoc groups inherently insecure, and therefore specialized security services are needed for their deployment. *Admission Control* (or secure node admission) is a fundamental security service in ad hoc groups; it is required to ascertain membership eligibility and to bootstrap other important security services, such as secure routing (e.g., [HPJ02, HJP02]) and secure group communication (e.g., [STW00b, STW00a]).

Node admission in ad hoc groups cannot be performed centrally. Since, requiring

constant presence (availability) of a central fixed entity which is charged with admission control is not realistic for many types of ad hoc groups. First, such an entity is a single point of failure. Second, it represents an attractive and high-payoff target for attacks. Third, topology changes due to mobility and node outages may cause the central entity to be unreachable and thus unable to perform its duties in the parts of a ad hoc group not connected to it. This motivates us to investigate admission control techniques that function in a distributed or decentralized manner. Since our emphasis is on security, the natural technology to consider is threshold cryptography.

Recall that the concept of threshold cryptography involves distributing cryptographic primitives (such as decryption or digital signatures) in order to secure them against corruption of a certain number of parties, i.e., a threshold. For example, a $(t+1, n)$ threshold signature scheme [DF90] allows, in a group of $n$ parties, to share the ability to digitally sign messages in such a way that any $t+1$ parties can do so jointly, whereas, no coalition of up to $(t)$ parties can. Such a threshold signature scheme is resilient against the so-called *static adversary* who corrupts at most $(t)$ parties in the entire lifetime of the system.

More advanced *proactive* cryptographic schemes [HJKY95] offer improved resistance against corruptions. Time is divided into *update rounds*, and the proactive scheme offers the same combination of security and robustness even in the presence of so-called *mobile adversaries* [OY91], whereby a potentially new set of up to $(t)$ parties becomes corrupted in each update round. This is done by the *proactive update* procedure which involves parties randomly re-sharing the shared secret at the start of each update round.

Two features of ad hoc groups make decentralized node admission a very challenging problem. *First*, the devices of ad hoc groups often have very weak computational facilities and battery power. *Second*, these devices usually function in an asynchronous (on/off) manner, often becoming temporarily unavailable. Therefore, an ideal admission control protocol must be efficient in terms of both computation and communication[1]. It must also involve minimal (ideally, *none* at all) interaction among the nodes of the network.

A number of admission control techniques have been proposed in recent years [KZL+01, KLX+02, LZK+02, NTY03, STY03, STY04]. (See the following section for more details on prior work.) Most are based on $(t+1, n)$ threshold cryptography and allow any set of $t+1$-out-of-$n$ nodes (called sponsors) to admit a new node by issuing to it:

[1] Communication is directly related to the consumption of battery power in most mobile devices [BA03].

(1) a share of a group secret (to be used in future admissions), and

(2) a membership certificate (used for secure communication)

In order to ensure the robustness of the secret sharing and secret reconstruction protocols in the presence of malicious nodes, Feldman's verifiable secret sharing (VSS as described in Section 5.3) [Fel87] is employed.

Unfortunately, all previous schemes are far from ideal. They are **heavily interactive** among the sponsors as far as either item (1) or (2) above. Furthermore, they are very computationally expensive in performing item (2). This severely limits their practicality.

**Our Contributions.** In this chapter, we first show that as long as each node is able to obtain an updated VSS information, there is no need for node-specific certificates. This implies that admission control for ad hoc groups can be realized by only issuing node-specific secret shares (item (1) above) and obviating the need for expensive membership certificate issuance.

Second, we construct an efficient and a **fully non-interactive** admission control protocol and evaluate it in the context of MANETs. In contrast with prior work, our protocol does not require any interaction among the nodes sponsoring admission and has a single round of asynchronous communication between the new node and each sponsoring node. We thoroughly analyze our protocol and show that it compares favorably to previous protocols.

The proposed protocol is the result of a collaborative work which first appeared in [STY05], and is also included in the dissertation by Jeong Hyun Yi [Yi05].

**Organization.** The rest of the chapter is organized as follows: we first review prior work in Section 7.2. The generic admission protocol ad hoc groups is presented in Section 7.3, followed by the overview of prior admission control protocols based on uni-variate polynomial secret sharing (UniAC) in Section 7.4. We then describe, in Section 7.5, the proposed admission control protocol based on bi-variate polynomial secret sharing (BiAC). The detailed performance results, analysis and comparison of BiAC with UniAC are presented in Section 7.7. Finally, some we discuss the security of the proposed scheme in Section 7.6.

## 7.2 Related Work

We now review relevant prior work in ad hoc group security. Zhou and Haas [ZH99] first suggested the use of threshold cryptography to secure mobile ad hoc networks. Their idea was to distribute the trust among the nodes of the network such that no less than a certain threshold of nodes are trusted. They proposed a distributed certification authority (CA) which issues certificates (using some threshold signature [DF90] protocol) to nodes joining the network. These certificates enable nodes to communicate with each other in a confidential and authenticated manner. This work also led to the development of COCA [ZSvR02], an on-line certification authority for wired networks. Although quite attractive, this idea is not directly applicable for the purposes of admission control in ad hoc groups. The proposed approach is hierarchical in the sense that only select nodes can serve as parts of the certification authority, i.e., take part in admission decisions. Moreover, contacting distributed CA nodes in a multi-hop and ever-changing ad hoc group might not always be possible.

Kong, et al. considered the same problem in series of papers [KZL+01, KLX+02, LZK+02, LKZ+04] and proposed a set of protocols for providing ubiquitous and robust admission control for ad hoc groups. They adapted the model of Zhou and Haas so that any node can participate in admission control decisions, thus maintaining the true "peer" nature of a ad hoc groups and providing increased availability. The security of their admission mechanism relies upon a specific variant of the proactive threshold RSA signature scheme. Unfortunately, this scheme is neither robust [NTY03] (i.e., it can not tolerate malicious nodes) nor secure [JSY04].

Recently, Narasimha, et al. [NTY03] and Saxena, et al. [STY04] proposed similar admission control protocols based on threshold DSA [GJKR96b] and threshold BLS [Bol03] signatures, respectively. While provably secure, both solutions are quite inefficient.

As pointed out in the previous section, all of the above techniques require admitting nodes to interact in order to issue a new node its secret share. Both heavy interaction and costly cryptographic computation make these techniques overly expensive for most ad hoc group applications.

The admission control technique developed in this thesis is completely non-interactive. It uses secret sharing based on so-called bi-variate polynomials which have been employed for related purposes in the literature [BOGW88, NPR99, BSH+92]. In particular, [LN03]

presents a key pre-distribution scheme for sensor networks using bi-variate polynomials [BSH+92] *in the presence of a centralized authority.* The protocol we propose is fully distributed and allows nodes in a ad hoc group to readily and efficiently share pairwise secret keys without any centralized support.

## 7.3 Generic Admission Control Protocol

We claimed earlier (in Section 7.1) that admission control for ad hoc groups can be realized by *only* issuing node-specific secret shares. Whereas, in prior proposals, it is also necessary to issue individual node membership certificates. We now discuss the reasoning behind this claim.

In prior solutions, certificates are needed to achieve secure communication among the nodes, and secret shares are needed to take part in the admission protocols. We observe that secure communication among the nodes can be achieved if each node just has a secret share and, in addition, the updated VSS information. (How to achieve such public-key based secure communication using shares and VSS information, is the topic of the next chapter.) The updated VSS information can be provided to a new node during the admission protocol by the existing nodes. Indeed, the VSS information needs to be signed (via a distributed signature protocol) by a set of $t + 1$ nodes. However, it only needs to be signed once per update interval (since the VSS remains constant per interval), and the signed responses can simply be copied to the new node during the admission protocol.

Based on the above description, we define an admission control mechanism for ad hoc groups as a set of three components:

1. *Initialization:* The group is initialized by either a trusted dealer or a set of founding members. The dealer or founding members initialize the group by choosing a group secret key, and computing and publishing the corresponding public parameters in the group certificate [KMT03]. The group secret is shared among the founding member(s) in such a way that any set of $t+1$ members can reconstruct it. The share of the group secret possessed by each member is referred to as its *secret share.*

2. *Admission:* A prospective member $P_{n+1}$ who wishes to join the group must be issued its secret share by current member nodes. $P_{n+1}$ initiates the admission protocol by sending a `JOIN_REQ` message to the network. A member node, that receives this

JOIN_REQ message and approves the admission of $P_{n+1}$, replies, over a secure channel, with a partial secret share (derived from its secret share) for $P_{n+1}$. Once $P_{n+1}$ receives partial secret shares from at least $t+1$ different nodes, it uses them to compute its secret share.

During the above process, a malicious node can easily preclude a prospective node from being admitted by inserting incorrect partial secret shares, i.e., a denial-of-service (DoS) attack. To prevent this, a prospective node must be able to verify the validity of its reconstructed secret share before using them. This feature is called *verifiability* in the rest of the chapter. Also, when the node detects that its secret share is invalid, it must be able to trace the bogus shares and the malicious node(s) in the ad hoc group. This functionality is provided by the so-called *traceability* feature. Note that *verifiability* is always required, whereas, *traceability* is only necessary when a node detects (via verifiability) that its reconstructed secrets are not valid.

3. *Pairwise Key Establishment:* Each node can use its secret share and/or the public parameters to compute pairwise keys with any other node. This allows nodes to securely communicate with each other.

## 7.4 UniAC: Admission Control using Uni-variate Polynomial Secret Sharing

In this section, we briefly describe previously proposed admission control methods [KZL+01, KLX+02, LZK+02, NTY03, STY03, STY04] (adapted with our optimization that obviates certificates). These methods are based on uni-variate polynomial secret sharing; we refer to them collectively as: UniVariate Admission Control (UniAC). UniAC involves the following steps (for protocol message flows, see Figure 7.2).

1. *Initialization:* The system can be initialized by a trusted dealer $TD$ or a set of founding nodes. As in Shamir's secret sharing [Sha79] based on a uni-variate polynomials, the $TD$ (or founding members) choose(s) a large prime $q$, and select(s) a polynomial

$$f(x) = \sum_{i=0}^{t} a_i x^i \pmod{q}$$

such that $f(0) = S$, where $a_i$-s are the coefficients of the polynomial, $q$ is a large prime, and $S$ is the group secret. The $TD$ computes each node's secret share $ss_i$

such that $ss_i = f(id_i) \pmod q$, and securely transfers $ss_i$ to node $P_i$. [Recall that any group of $t+1$ nodes, say first $t+1$ nodes, who have their shares can recover the secret using Lagrange interpolation: $f(0) = \sum_{i=1}^{t+1} ss_i \ \lambda_i(0) \pmod q$, where $\lambda_i(x) = \prod_{j=1, j\neq i}^{t+1} \frac{x - id_j}{id_i - id_j} \pmod q$.]

$TD$ also publishes a commitment to the polynomial as in VSS. VSS setup involves a large prime $p$ such that $q$ divides $p-1$ and a generator $g$ which is an element of $\mathbb{Z}_p^*$ of order $q$. $TD$ computes $W_i$, called the **witness**, such that $W_i = g^{a_i} \pmod p$ for all $i \in [0, t]$, and publishes these $W_i$-s in the group certificate.

2. *Admission:* We described this procedure in Section 5.7. During this procedure (which we call *random shuffling* hereafter), a new node $P_{n+1}$ is given a *shuffled* partial secret share as $pss_j(n+1) = ss_j \lambda_j(id_{n+1}) + R_j \lambda_j(0) \pmod q$ by each sponsoring node $P_j$. Here, $R_j \in \mathbb{Z}_q$ denotes $P_j$'s random secret share of "zero", and is generated during the *joint zero secret sharing* protocol (JZSS as described in Section 5.5) among the sponsoring nodes. Upon receiving these partial share values from $t+1$ nodes, $P_{n+1}$ obtains its secret share $ss_{n+1}$ by simply adding them. Note that $\sum_{j=1}^{t+1} R_j \lambda_j(0) = 0$. (This random shuffling process is illustrated in Figure 7.1(a).)

   $P_{n+1}$ then performs the verifiability checking and, if needed, the traceability procedure. (See [CSY05] for details regarding the actual computations involved in these procedures.)

   *We note that, due to the above random shuffling procedure, this admission protocol becomes heavily interactive among the $t+1$ sponsoring nodes – it requires $O(t^2)$ point-to-point messages as well as extremely expensive $O(t)$ reliable broadcast messages [Bra84]. All this makes it impractical for most ad hoc group settings.*[2]

3. *Pairwise Key Establishment:* Any pair of nodes $P_i$ and $P_j$ can establish shared keys using their respective secret shares $ss_i$, $ss_j$ and public VSS information. $P_i$ computes:

$$g^{ss_j} = \prod_{k=0}^{t} (W_k)^{{id_j}^k} \pmod p$$

[2]The messages in the random shuffling procedure can be routed via the new node $P_{n+1}$, as was proposed in [LL00]. This avoids any direct communication among the $t+1$ sponsors, however, at the cost of $P_{n+1}$ becoming the bottleneck.

from the public witness values, and exponentiates it with its share $ss_i$ to get a key $K_{ij} = (g^{ss_j})^{ss_i} \pmod p$. Similarly, $P_j$ computes:

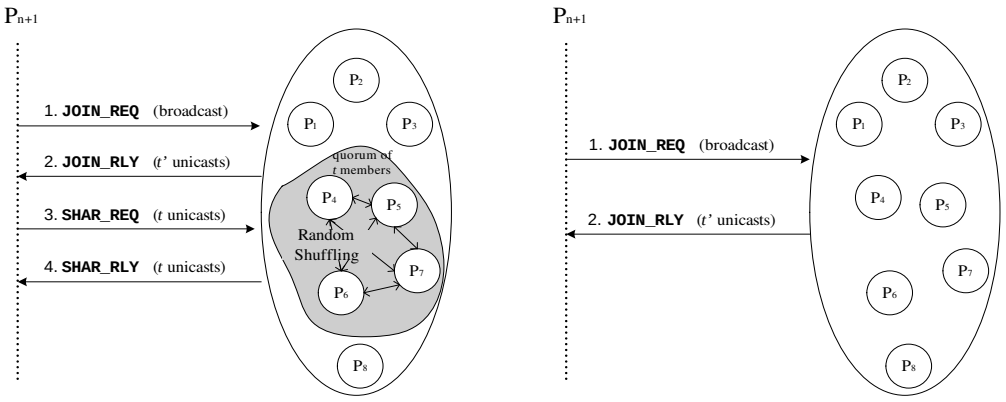
$$g^{ss_i} = \prod_{k=0}^{t} (W_k)^{id_i{}^k} \pmod p$$

and exponentiates it with its share $ss_j$ to get a key $K_{ji} = (g^{ss_i})^{ss_j} \pmod p$. Since, $K_{ij} = K_{ji}$, $P_i$ and $P_j$ now have a shared secret key.

This key establishment procedure remains secure under the computational Diffie-Hellman (CDH) assumption in the random oracle model (ROM). (The security proof appears in the next chapter).

## 7.5 BiAC: Non-interactive Admission Control

We now describe a new admission technique for ad hoc groups. It is based on secret sharing using bi-variate polynomials and is fully non-interactive. We call the protocol BiVariate Admission Control (BiAC).




(a) UniAC (b) BiAC

Figure 7.1: Comparison of UniAC and BiAC

### 7.5.1 Overview

As shown in Figure 7.1(b), we avoid interaction among sponsoring nodes by using a *bi-variate* polynomial $f(x, y)$. Bi-variate polynomials have been previously used for related purposes [BOGW88, NPR99, BSH+92].

The secret sharing based on bi-variate polynomial works as follows. To distribute shares among $n$ nodes, a trusted dealer chooses a large prime $q$, and selects a random symmetric bi-variate polynomial $f(x, y) = \sum_{\alpha=0}^{t} \sum_{\beta=0}^{t} f_{\alpha\beta} x^{\alpha} y^{\beta} \pmod{q}$ such that $f(0, 0) = S$, where the constants $f_{\alpha\beta}$-s are the coefficients of the polynomial and $S$ is the group secret. Since the polynomial is symmetric, $f_{\alpha\beta} = f_{\beta\alpha}$ for each $\alpha, \beta$ and $f(x, y) = f(y, x)$. For each node $P_i$, the dealer computes a uni-variate polynomial, called a *share-polynomial*, $b_i(x)$ of degree $(t)$ such that $b_i(x) = f(x, id_i) \pmod{q}$, and securely transfers $b_i(x)$ to each node $P_i$. Note that, after initializing at least $t + 1$ nodes, the dealer is no longer needed.

In order to admit a new node $P_{n+1}$, the current member nodes must issue it a *share-polynomial* $b_{n+1}(x)$ in a distributed manner. This can be achieved if at least $t + 1$ member nodes provide $P_{n+1}$ with partial shares $b_j(id_{n+1})$ such that $b_j(id_{n+1}) = f(id_{n+1}, id_j)$ for some $j \in [1, n]$. $P_{n+1}$ can then use the standard Gaussian elimination procedure [PFTV92] to compute $f(id_{n+1}, x)$, which is the same as $f(x, id_{n+1})$ (since the polynomial $f(x, y)$ is symmetric) and thus obtain its share-polynomial $b_{n+1}(x) = f(x, id_{n+1})$ from $t + 1$ partial shares $b_j(id_{n+1})$.

Unlike protocols based on sharing of uni-variate polynomials, this scheme *does not* require any interaction among the admitting member nodes.

### 7.5.2 Initialization

In *BiAC*, the ad hoc group can be initialized by one node (centralized initialization) or a set of nodes (distributed initialization).

**Centralized Initialization:** the trusted dealer $TD$ computes a two-dimensional sharing of the secret by choosing a random bi-variate polynomial:

$$f(x, y) = \sum_{\alpha=0}^{t} \sum_{\beta=0}^{t} f_{\alpha\beta} x^{\alpha} y^{\beta} \pmod{q}$$

such that $f(0, 0) = S$. $TD$ computes $W_{\alpha\beta}$, called a **witnesses**, such that $W_{\alpha\beta} = g^{f_{\alpha\beta}} \pmod{p}$ for all $\alpha, \beta \in [0, t - 1]$, and publishes these $W_{\alpha\beta}$-s as part of the group certificate.

Once $TD$ computes the witness matrix, it sends each node $P_i$ ($i \in [1, n]$) a distinct *share-polynomial:* $b_i(x) = f(x, id_i)$. $TD$'s presence is needed only during this initialization phase in order to bootstrap the system.

**Distributed Initialization:** alternatively, the network can be initialized by a set of $t + 1$ or more founding nodes. These nodes agree on a random bi-variate polynomial $f(x, y)$ using the bi-variate variant of GJKR-DKG, the *Distributed Key Generation (DKG)* technique of [GJKR99b] (refer to Section 5.4 for details).

### 7.5.3 Admission Process

In order to join the network, $P_{n+1}$ must collect at least $t + 1$ partial shares of the polynomial from $t + 1$ current nodes. Figure 7.3 shows the protocol message flow for the node admission process[3]. We assume that the communication between $P_{n+1}$ and each $P_i$ takes place over a secure channel. Such channels can be established, for example, by using out-of-band channels (as was proposed in Chapters 3 and 4).

| | | |
|---|---|---|
| $P_{n+1} \rightarrow P_i$: | $id_{n+1}$ | (1) |
| $P_{n+1} \leftarrow P_i$: | $id_i$ | (2) |
| $P_{n+1} \rightarrow P_j$: | $SL_{n+1}$ | (3) |
| $P_i \longleftrightarrow P_j$ : | *Random Shuffling* | (4) |
| $P_{n+1} \leftarrow P_j$: | $pss_j(n + 1)$ | (5) |

Figure 7.2: UniAC Admission Protocol

| | | |
|---|---|---|
| $P_{n+1} \rightarrow P_i$: | $id_{n+1}$ | (1) |
| $P_{n+1} \leftarrow P_i$: | $id_i, b_i(id_{n+1}),$ | (2) |

Figure 7.3: BiAC Admission Protocol

1. $P_{n+1}$ broadcasts `JOIN_REQ` message which contains its identity $id_{n+1}$.

2. Each receiving node ($P_i$) willing to admit $P_{n+1}$, computes a *partial share* $b_i(id_{n+1})$ using its own *share-polynomial* such that $b_i(id_{n+1}) = f(id_{n+1}, id_i)$. $P_i$ then replies

[3] In order to secure the protocol against common attacks such as *replay*, *impersonation*, and *interleaving* attacks [MvOV97], we note that it is necessary to include additional information such as timestamps, nonces, and identity information of the sender as well as the receiver. However, in order to keep our description simple, we omit these values.

to $P_{n+1}$ with a `SHARE_REP` message. Each message contains $b_i(id_{n+1})$ along with the respective identifiers $id_i$.

Note that, in order to compute their partial shares above, sponsors do not need to be aware of each other and thus no interaction is needed. This is in contrast with UniAC scheme, where each sponsor needs to be aware of and connected with all other sponsors.

3. Upon receiving $m$ ($\geq t+1$) `SHARE_REP` messages, $P_{n+1}$ selects **any** $t+1$ of them and interpolates its own share-polynomial $b_{n+1}(x)$ using standard Gaussian elimination as follows. Let us denote the share-polynomial $b_{n+1}(x)$ reconstructed by $P_{n+1}$ as $\sum_{\alpha=0}^{t} A_\alpha x^\alpha$. Since $b_i(id_{n+1}) = b_{n+1}(id_i)$ (due to symmetry), the problem to interpolate $b_{n+1}(x)$ using $t+1$ $b_i(id_{n+1})$-s is equivalent to the problem to solve the matrix $A$ such that $XA = B$ as follows:

$$\begin{bmatrix} (id_1)^0 & (id_1)^1 & \cdots & (id_1)^t \\ (id_2)^0 & (id_2)^1 & \cdots & (id_2)^t \\ & & \vdots & \\ (id_{t+1})^0 & (id_{t+1})^1 & \cdots & (id_{t+1})^t \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_t \end{bmatrix} = \begin{bmatrix} b_{n+1}(id_1) \\ b_{n+1}(id_2) \\ \vdots \\ b_{n+1}(id_{t+1}) \end{bmatrix}$$

The above system of linear equations yields a unique solution since the $id_i$ values are distinct and the matrix $X = [x_{ij}]$, where $x_{ij} = (id_i)^{j-1}$ for all $i, j \in [0, t]$, is invertible.

*Verifiability:* In order to verify the acquired hare-polynomial $\sum_{\alpha=0}^{t} A_\alpha x^\alpha$, $P_{n+1}$ must perform the verifiability procedure. In order to be a valid share-polynomial, $A_\alpha$ must be equal to $\sum_{\beta=0}^{t} f_{\alpha\beta}(id_{n+1})^\beta$, for all $\alpha \in [0, t]$. Using the public witness values (from the group certificate) $W_{\alpha\beta} = g^{f_{\alpha\beta}} \pmod p$, the polynomial can be verified as follows:

$$g^{A_\alpha} = \prod_{\beta=0}^{t} (W_{\alpha\beta})^{(id_{n+1})^\beta} \pmod p$$

for all $\alpha \in [0, t]$.

Note that the right-hand side in the above equation can be pre-computed by $P_{n+1}$ prior to starting the admission process.

*Traceability:* If verification fails, $P_{n+1}$ can trace the faulty share provider(s) by performing the traceability procedure. This involves verifying the validity of each partial

share $b_i(id_{n+1}) = f(id_{n+1}, id_i)$, that $P_{n+1}$ received. This can be achieved by checking the following equation for each $i$:

$$g^{b_i(id_{n+1})} = \prod_{\alpha=0}^{t} \prod_{\beta=0}^{t} (W_{\alpha\beta})^{(id_{n+1})^\alpha (id_i)^\beta} \pmod{p}$$

Note that $\prod_{\alpha=0}^{t} (W_{\alpha\beta})^{(id_{n+1})^\alpha}$ in the above equation can be pre-computed since $W_{\alpha\beta}$-s and $id_{n+1}$ are known to $P_{n+1}$ in advance.

### 7.5.4 Pairwise Key Establishment

Once every node has its share-polynomial, pairwise key establishment is the same as in [BSH+92] and [LN03]. Any pair of nodes $P_i$ and $P_j$ can establish shared keys as follows: $P_i$ uses its share-polynomial $f(x, id_i)$ to compute

$$K_{ij} = f(id_j, id_i) \pmod{q}$$

and $P_j$ its share-polynomial $f(x, id_j)$ to compute

$$K_{ji} = f(id_i, id_j) \pmod{q}.$$

Since $f(x, y)$ is a symmetric polynomial, $K_{ij} = K_{ji}$. Thus, $P_i$ and $P_j$ now have a shared key that can be used for secure communication.

Unlike the pairwise key establishment in UniAC (security of which is based on the CDH assumption in ROM) as described in Section 7.4, the security of above procedure is unconditional, i.e., not based on any assumption. Refer to [BSH+92] for details regarding the security arguments of this pairwise key establishment.

Table 7.1: Feature Comparison

| Key Features | UniAC | BiAC |
|---|---|---|
| Security Assumption (for Admission) | DL | DL |
| Security Assumption (for Key Comp.) | CDH | Unconditional |
| Decentralized Admission | Yes | Yes |
| DoS Resistance | Yes | Yes |
| Interaction among Sponsors Required | Yes | No |
| Random Shuffling Required | Yes | No |
| Reliable Broadcast Required | Yes | No |

## 7.6 Security Analysis

In this section, we discuss the security of the proposed BiAC scheme. Security of BiAC is based on the discrete logarithm (DL) assumption as long as the adversary is not allowed to corrupt more than $(t)$ $(< n/2$, where $n$ is the total number of nodes) nodes in the network. We only present a sketch of this argument. Basically, as in the Feldman's VSS, we use the idea of *simulated adversarial view* to show that an adversary who corrupts at most $t$ nodes learns nothing extra (other than the witness $g^S \pmod p$) about the secret $S$ during the initialization and admission procedures of the scheme. This is achieved by generating a simulator, which on input $g^S \pmod p$, produces public information and the private information to the adversary which is statistically indistinguishable from the one produced in the actual run of these procedures.

For the security arguments of BiAC pairwise key establishment, we refer the reader to [BSH+92].

## 7.7 Performance Analysis

In this section we discuss the implementation of UniAC and BiAC and compare them in terms of node admission, traceability and pair-wise key establishment costs. We also summarize and compare some salient features in Table 7.1. As expected, BiAC significantly outperforms UniAC in our overall evaluation.

### 7.7.1 Complexity Analysis and Comparison

We summarize computation and communication complexities[4] in Tables 7.2 and 7.3, respectively, where $n \geq m \geq t$. More specifically, BiAC requires each sponsoring node $P_i$ to perform $O(t)$ modular multiplications and the joining node $P_{n+1}$ to perform $O(t^3)$ modular multiplications for Gaussian elimination and $O(t)$ exponentiations for verifiability. On the other hand, UniAC requires each $P_i$ to perform $O(t)$ multiplications, and $P_{n+1}$ to perform $O(t)$ multiplications plus 1 exponentiation for verifiability. For traceability, both the schemes require $O(t^2)$ multiplications and $O(t^2)$ exponentiations with pre-computation. BiAC is significantly more efficient than UniAC for computing pairwise keys, since the former

[4]The costs required for protecting each protocol message are not taken into account since these costs vary with the specific signature scheme.

requires only $O(t)$ multiplications, while the latter needs $O(t)$ exponentiations as well as $O(t)$ multiplications. Note that, pairwise key establishment is a very frequent operation in a MANET, thus, its efficiency is extremely important.

Table 7.2: Computation Complexity

| Category | | | UniAC | BiAC |
|---|---|---|---|---|
| Admission | $P_i$'s view | $\mathcal{M}$ | $O(t)$ | $O(t)$ |
| | | $\mathcal{E}$ | $O(t)$ | 0 |
| | $P_{n+1}$'s view | $\mathcal{M}$ | $O(t)$ | $O(t^3)$ |
| | | $\mathcal{E}$ | 1 | $O(t)$ |
| Traceability | | $\mathcal{M}$ | $O(t^2)$ | $O(t^2)$ |
| | | $\mathcal{E}$ | $O(t^2)$ | $O(t^2)$ |
| Pairwise Key Establishment | | $\mathcal{M}$ | $O(t)$ | $O(t)$ |
| | | $\mathcal{E}$ | $O(t)$ | 0 |

$\mathcal{M}$: modular multiplication $\mathcal{E}$: modular exponentiation

Table 7.3: Communication Complexity

| Category | | UniAC | BiAC |
|---|---|---|---|
| Rounds | broadcast | 1 | 1 |
| | unicast | $O(t^2)$ | $O(t)$ |
| | reliable broadcast | $O(t)$ | 0 |
| Bandwidth | $\log q$-bit | $O(t^2)$ | $O(t)$ |
| | $\log p$-bit | $O(t)$ | $O(t)$ |

As far as overall communication costs[5], BiAC consumes $O(t \log q)$ and $O(t \log p)$ bits, while bandwidth consumption in UniAC is $O(t^2 \log q)$ plus $O(t \log p)$ bits due to the interactive random shuffling procedure.

### 7.7.2 Experimental Setups

UniAC and BiAC protocols have been implemented over the popular OpenSSL library [Operg]. The source is written in C in Linux and consists of about $10,000$ lines of code for each protocol. The code is available at [Peeac].

We used five laptops in our experimental set-up: four with Pentium-3 800MHz CPU-s and 256MB memory and one with Mobile Pentium 1.8 GHz CPU and 512MB memory. Each laptop ran Linux 2.4 and was equipped with a 802.11$b$ interface configured for

[5]We assume that the identity and the public key are $\log q$ bits long and $\log p$ bits long, respectively.

ad-hoc mode. Specifically, for measuring the admission cost, four laptops with the same computing power were used as current member nodes and the high-end laptop was used as the joining/new node. Traceability and pairwise key computation experiments were also performed with this high-end laptop. In our experiments, each node (except the joining node) was emulated by a daemon and each machine was running up to three daemons. The measurements were performed with different threshold values $t$. The size of the parameter $q$ was set to be 160-bits and $p$ 1024-bits.

To measure consumption of battery power, we performed the following experiment: the test machine was an iPAQ (model H5555) running Linux (Familiar-0.7.2). The CPU on iPAQ is a 400 MHz Intel XScale with 48MB of flash memory and 128MB of SDRAM. In order to obtain accurate power measurements, we removed the battery from the iPAQ during the experiment and placed a resistor in series with the power supply. We used a National Instruments PCI DAQ (Data AcQuisition) board to sample the voltage drops across the resistor to calculate current at 1000 samples per second.

### 7.7.3 Experimental Results

We compare our experiment results in terms of admission, energy consumption for admission, traceability, and pairwise key computation.

#### 7.7.3.1 Admission Results

To evaluate admission cost, we measured total processing time between sending of `JOIN_REQ` by the prospective member and receiving (plus verification) of acquired secret shares. Our measurements include the average computation time of the basic operations (such as modular multiplications, exponentiations etc.) as well as communication costs, such as packet en/decoding time, network delay, and so on.

As observed from Figure 7.4, the admission (join) cost with BiAC is much lower than that with UniAC. The difference is even higher for higher threshold values. The reason is quite intuitive: not only is BiAC computationally cheaper than UniAC, but it also requires less communication.

Energy consumption results for admission operation are plotted in Figure 7.5. This experiment is quite tricky to measure fairly. Energy consumption is directly proportional to processing time. It is meaningless to measure energy consumption based on computation

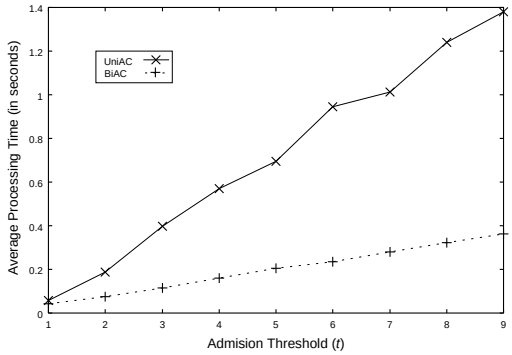



Figure 7.4: Admission Costs

time. However, it is well known that, in many small devices such as low-end MANET nodes or sensors, sending a single bit is roughly equivalent to performing 1,000 32-bit computations in terms of batter power consumption [BA03]. Therefore, we measured the power consumption in terms of communication bandwidth required by each admission protocol. In more detail, we sent some bulk data (e.g., 100 Mbytes) from a single iPAQ PDA, measured the power consumed while sending out this data, and then computed the average power consumption per bit. After that, we calculated the power consumption of each admission protocol by multiplying this measurement result by the bit length of the transmitted data. These results in Figure 7.5 clearly illustrate that BiAC is much more energy-efficient than UniAC.

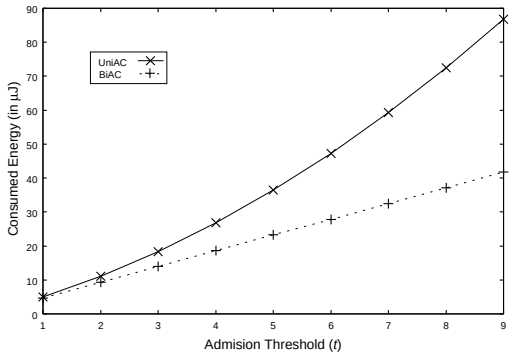



Figure 7.5: Energy Consumption with Communication for Admission (iPAQ-H5555: XScale 400 MHz, 128MB)

#### 7.7.3.2 Traceability Results

Figure 7.6 displays traceability costs for the two approaches. Even in the worst case, BiAC is as good as UniAC for performing the (very infrequent) operation of tracing malicious nodes.

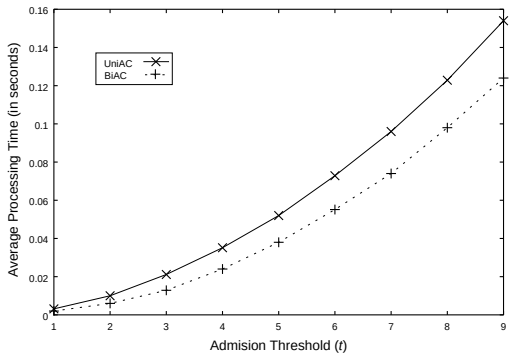



Figure 7.6: Traceability Costs

#### 7.7.3.3 Pairwise Key Establishment Results

Figure 7.7 shows that BiAC is significantly more efficient than UniAC for computing pairwise keys. The achieved gains range approximately from 115 ($t = 1$) to 412 ($t = 9$); in other words, BiAC is 115 to 412 times faster than UniAC when establishing a shared secret key. This result was actually expected because in BiAC the pairwise key computation requires only $O(t)$ multiplications where the modular size is 160 bits. In contrast, UniAC requires $O(t)$ exponentiations with a modular size of 1024 bits as well as $O(t)$ multiplications with 160-bit modulus.

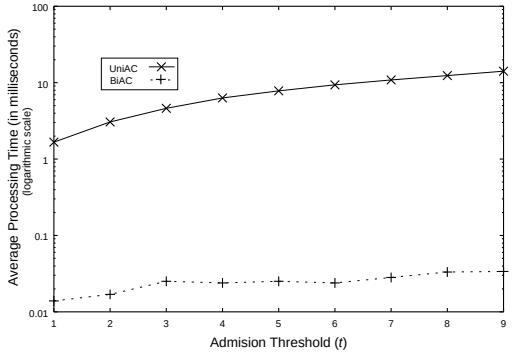



Figure 7.7: Pairwise Key Establishment Costs

# Chapter 8

# Efficient Secure Communication in Ad Hoc Groups

*In this chapter, we show how to perform necessary public key operations without node-specific certificates in ad hoc groups. These operations include pair-wise key establishment, signing, and encryption. Our proposal is based upon the use of secret shares generated in Feldman's VSS as private keys.*

## 8.1 Introduction

We saw in Chapter 7, that as long as each node is able to obtain an updated VSS information, there is no need for node-specific certificates. However, Chapter 7 focuses mainly on how to efficiently admit new nodes, i.e., how to create new secret shares in a distributed manner. In this chapter, we are concerned with the problem of how to enable secure communication among the nodes once they have been admitted. In particular, we show that the secret shares created by Feldman's VSS can be securely and efficiently used as private keys in many standard discrete-log based public-key cryptosystems, namely in a Schnorr signature scheme, in an ElGamal encryption, and in a non-interactive version of the Diffie-Hellman pairwise key establishment protocol. Note that if the VSS share $x_i$ is treated as $P_i$'s private key, the Feldman's VSS public information allows everyone to compute the corresponding public key $y_i$.

**Motivation.** The motivation for establishing pairwise keys is straight-forward – it is needed to secure communication between any pair of nodes, e.g., as required in various secure routing protocols, such as Ariadne [HPJ02]. Signing is required in cases when *non-repudiation* is needed, e.g., as in ARAN secure routing protocol [DLRS01]. Encryption is suitable for scenarios where an authorized node outside the network needs to send a *private* query to a node inside. An example scenario is in a wireless sensor network, where a base station sends a maintenance query to a particular sensor node (e.g., to obtain its reading of nuclear activity in the environment). However, sending the query in clear would leak critical information to an adversary who might be interested in knowing what the sensor network is installed for (e.g., for detecting a nuclear attack [Hil01]).

**Related vs. Independent Keys.** It is not obvious whether the proposed usage of secret shares as private keys is safe. The reason is simple – unlike the standard public-key cryptosystems where every user gets an *independently created* private/public key pair, here the private keys of all parties are related by being values of a $t$-degree polynomial (Note, for example, that any set of $t+1$ such values determines all the others). Recall, for example, that the "text-book RSA" is not secure when public keys of two users are related [MvOV97].

**Our Contributions.** We show that indeed such use of the *secret shares as private keys* is just as secure as the standard discrete-log based signatures, encryption, and key establishment, as long as no more than $t$ of the players in the group collude or are corrupted by an attacker. Note that this is the best that one can hope for because if the private keys are shares in a secret sharing with $t$-degree privacy threshold, any collection of $t+1$ such keys enables reconstruction of the whole secret-sharing and hence also all the other private keys. Our proposal renders necessary public key operations efficiently feasible in ad hoc networks, without the need of certificates.

**Threshold-tolerant ID-based Cryptography.** The proposed scheme is essentially equivalent to an identity-based cryptosystem that tolerates upto a threshold of corruptions/collusions. However, as compared to well-known ID-based cryptographic mechanisms, such as IBE [BF01] and other related schemes, our approach is more efficient and is also based on standard cryptographic assumptions.

**Organization.** Section 8.3 presents our new scheme. In Section 8.4, we compare our proposal to prior identity-based cryptosystems.

## 8.2 Communication and Adversarial Model

We work in the standard model of threshold cryptography and distributed algorithms known as synchronous, reliable broadcast, static adversary model. This model involves nodes equipped with synchronized clocks. We assume some nomenclature system that provides each node in the network with a unique identifier, and also that it's computationally hard for an adversary to forge identities.

We assume the existence of an on-line trusted public repository where the network-wide or group public key is published. The nodes (both within and outside the network) are connected by weakly synchronous communication network offering point-to-point channels and a reliable broadcast. To interact with a node in the network, an outsider must first be able to retrieve the group public key from the repository.

We consider the presence of the so-called "static" adversary, modeled by a probabilistic polynomial time algorithm, who can *statically*, i.e., at the beginning of the life time of the scheme, schedule up to $t < n/2$ arbitrarily malicious faults among $n$ users in the group. Such an adversary is said to break our scheme if it is able to break the underlying key establishment, signature and encryption schemes against the standard notions of security.

## 8.3 Our Proposal: "Secret-Shares-as-Private-Keys"

In this section we present our proposal on using secret VSS shares as private keys that renders public key operations efficiently feasible in ad hoc networks. We begin by providing a brief overview of the scheme.

### 8.3.1 Overview

The idea of the scheme is very simple. Basically, we use Feldman's VSS ( summarized in Section 5.3), to build our scheme.

A dealer (or a set of founding nodes) chooses a secret sharing polynomial $f(z) = a_0 + a_1 z + \cdots a_t z^t$ in $\mathbb{Z}_q$, where $a_0$ (also denoted as $x$) is the group secret key. The dealer

also publishes commitments to the coefficients of the polynomial, as $w_i = g^{a_i} \pmod p$, for $i = 0, \cdots, t$. These witnesses constitute the public key of the group. To join the group, a user $M_i$ with a unique identifier (such as an email address) $id_i$, receives from the dealer (or a set of $t+1$ or more nodes distributedly as described in previous chapter) a secret share $x_i = f(id_i) \pmod q$ over a secure channel. The public key $y_i = g^{x_i} \pmod p$ of $M_i$ can be computed using the public key of the group and its identifier $id_i$ as

$$y_i = \prod_{j=0}^{t} (w_j)^{{id_i}^j} \pmod p$$

Now, any user (within or outside) the group, can send encrypted messages to $M_i$ using its public key $y_i$, which $M_i$ can decrypt using its secret key $x_i$. Similarly, $M_i$ can use $x_i$ to sign messages, which can be publicly verified using $y_i$. Moreover, any two users $M_i$ and $M_j$ can establish pairwise keys in a non-interactive manner: $M_i$ and $M_j$ compute $k_{ij} = (y_j)^{x_i} \pmod p$, and $k_{ji} = {y_i}^{x_j} \pmod p$, respectively. Since $K_{ij} = k_{ij} = k_{ji}$, a hash of $K_{ij}$ can be used as session keys for secure communication between $M_i$ and $M_j$.

We call these secret sharing based pairwise key establishment, signature and encryption procedures as SS-KE, SS-Sig and SS-Enc, respectively. SS-Sig is realized using the Schnorr's signature scheme, and SS-Enc using ElGamal encryption.

**Remark:** The scheme that we present in this chapter is based on the original Feldman's VSS, i.e., the one that uses uni-variate polynomial for secret sharing. However, the scheme is also applicable to the variant of VSS based on bi-variate polynomials, and thus can be used in conjunction with the solutions that we proposed in the last chapter.

### 8.3.2 Setup and Joining

In order to setup the system, a dealer first chooses appropriate parameters $(p, q, g)$ for the group, and selects a polynomial $f(z) = a_0 + a_1 z + \cdots + a_t z^t$ in $\mathbb{Z}_q$, where $a_0$ (also denoted as $x$) is the group secret. The dealer keeps the polynomial secret and publishes commitments to the coefficients of the polynomial, as $w_i = g^{a_i} \pmod p$, for $i = 0, \cdots, t$. These witnesses constitute the public key of the group.

To join the group, a user $M_i$ sends its unique identifier $id_i$ to the dealer, who issues it its secret share $x_i = f(id_i) \pmod q$. (We assume there exists some kind of a unique nomenclature system for the users in the group, and that its computationally hard for anyone to forge the identities.)

**Remark:** In an ad hoc network, the setup and joining are performed in a distributed manner. The setup is performed using GJKR-DKG and the joining is performed using the admission protocol UniAC based on random shuffling (described in the previous chapter).[1]

### 8.3.3 SS-KE: Secret Sharing based Pairwise Key Establishment

Any pair of users $M_i$ and $M_j$ in the group can establish shared keys with each other using their secret keys and the group public key. $M_i$ computes the public key $y_j$ of $M_j$ (knowing its identifier $id_j$ only) as

$$y_j = \prod_{i=0}^{t} (w_i)^{{id_j}^i} \pmod{p}$$

$M_i$ then exponentiates $y_j$ to its own secret key $x_i$, to get $k_{ij} = {y_j}^{x_i} = g^{x_j x_i} \pmod{p}$. Similarly, $M_j$ computes public key $y_i$ of $M_i$ as

$$y_i = \prod_{j=0}^{t} (w_j)^{{id_i}^j} \pmod{p},$$

and exponentiates it to its own secret key $x_j$, to get $k_{ji} = {y_i}^{x_j} = g^{x_i x_j} \pmod{p}$. Since, $k_{ij}$ equals $k_{ji}$, $M_i$ and $M_j$ can use $K_{ij} = H(k_{ij}) = H(k_{ji})$, as a session key for secure communication with each other.

**Computational Complexity.** Each party needs to compute the other party's public key via interpolation, and one exponentiation only. Using the well-known scheme of multi-exponentiation (or Shamir's trick) [Möl01], the cost of interpolation is $O(log(n^t))$ squarings and $O(log(n^t))$ multiplications, where $n$ denotes the total number of parties. For reasonable threshold values and network sizes, the interpolation is fairly efficient.

Next, we present the security argument for the above SS-KE procedure. Basically we show that an adversary, who corrupts $t$ users, can not distinguish a key $K_{IJ}$ for some uncorrupted user pair $(M_I, M_J)$ from random *even* if he learns all other session keys $K_{ij}$ for

[1] Note that the security proofs for the secret sharing based key establishment, signatures and encryption that we present in this chapter are based on Feldman's VSS and require an on-line *TD*. However, same proofs also hold when the setup is performed in a distributed manner using GJKR-DKG and joining using UniAC; the only difference being in their respective simulations, which follow directly from the simulations of GJKR-DKG and of UniAC.

$(i,j) \neq (I,J)$. This is the standard notion for the security of a key establishment protocol and is adopted from [CK01].

**Theorem 10 (Security of SS-KE).** *Under the CDH Assumption in ROM, there exists no probabilistic polynomial time adversary A, which on inputs of secret keys of t corrupted users, and shared keys $K_{ij}$ between every user pair except $K_{IJ}$ $\{(i,j) \neq (I,J)\}$, is able to distinguish with a non-negligible probability $K_{IJ}$ from a random value.*

*Proof.* We prove the above claim by contradiction, i.e, we prove that if a polynomial time adversarial algorithm A, which on inputs of secret keys of $t$ corrupted users, and shared keys $K_{ij}$ between every user pair except $K_{IJ}$ $\{(i,j) \neq (I,J)\}$, is able to distinguish with a non-negligible probability $K_{IJ}$ from a random value, then there exists a polynomial time algorithm B, which is able to break the CDH assumption in the random oracle model.

In order to construct the algorithm B which breaks the CDH assumption, we first construct a polynomial time algorithm C, which breaks the SCDH assumption. The algorithm C runs on input of an SCDH instance $y = g^x \pmod p$, and would translate the adversarial algorithm A into outputting $g^{x^2} \pmod p$.

Without loss of generality, we first assume that the adversary A corrupts $t$ players denoted by $M_1, M_2, \cdots, M_t$. Now, the algorithm C runs as follows:

As in the simulation of Feldman's VSS C picks $x_1, x_2, \cdots, x_t$ values corresponding to the secret keys of corrupted users, uniformly at random from $\mathbb{Z}_q$. It then sets $x_i = F(id_i)$, and employs appropriate Lagrange interpolation coefficients in the exponent to compute the public witnesses $g^{A_1}, \cdots, g^{A_t} \pmod p$, where $F(z) = x + A_1 z + \cdots + A_t z^t \pmod q$.

Corresponding to the shared keys $K_{ij}$ between every user pair, C picks a random value $R_{ij}$, and runs the algorithm A on $x_1, \cdots, x_t$ and $R_{i,j}$ values. Note that the values $x_1, \cdots, x_t$ and the witnesses have an identical distribution to an actual run of the Feldman's secret sharing protocol, and therefore A can not see the difference between C's inputs and actual protocol run. Also, since the $K_{ij}$ values for $(i,j) \neq (I,J)$ are obtained by hashing $g^{x_i x_j}$, the only way A can tell the difference, except with negligible probability, between $K_{i,j}$ and $R_{i,j}$ for $(i,j) \neq (I,J)$, is by querying the random oracle on at least one appropriate $g^{x_i x_j}$ value. If A does tell the difference, then C records $R = g^{x_i x_j}$, and use the following equations to compute $g^{x^2}$,

$$x = \sum_{k=1}^{t} x_k l_k^i + x_i l_i^i \pmod{q}$$

$$x = \sum_{k=1}^{t} x_k l_k^j + x_j l_j^j \pmod{q}$$

($l_k^i$ denotes the lagrange coefficient $l_k^G(0)$, where $G = \{1, \cdots, t, i\}$).
Multiplying above two equations, we get

$$x^2 = (\sum_{k=1}^{t} x_k l_k^i)(\sum_{k=1}^{t} x_k l_k^j) + x_i x_j l_i^i l_j^j \pmod{q}$$

This implies,

$$g^{x^2} = g^{(\sum_{k=1}^{t} x_k l_k^i)(\sum_{k=1}^{t} x_k l_k^j)} R^{l_i^i l_j^j} \pmod{p}$$

If A doesn't tell the difference between $K_{i,j}$ and $R_{i,j}$ for $(i,j) \neq (I,J)$, then it must tell the difference between $K_{I,J}$ and $R_{I,J}$. However, as above, this is only possible, except with negligible probability, if A queries $g^{x_I x_J}$ to the random oracle. Them C records this value (say $K$) and computes $g^{x^2}$ similarly as above, using the following equation

$$g^{x^2} = g^{(\sum_{k=1}^{t} x_k l_k^I)(\sum_{k=1}^{t} x_k l_k^J)} K^{l_I^I l_J^J} \pmod{p}$$

Now, we will use C to construct B to break a CDH instance $(g^u, g^v)$. This is very simple as outlined in [MW96]: B runs C on input $g^u$, then on $g^v$, and finally on $g^{u+v} = g^u g^v$, and receives $g^{u^2}, g^{v^2}, g^{(u+v)^2}$, respectively. Now, since $(u+v)^2 = u^2 + v^2 + 2uv \pmod{q}$, B can easily compute $g^{uv}$ from the outputs of C.

Clearly, $Pr(B) = Pr(C)^3$, where $Pr(B), Pr(C)$, denote the probabilities of success of B and C respectively.

□

### 8.3.4 SS-Sig: Secret Sharing based Signatures

As mentioned previously, we realize SS-Sig using the Schnorr's signature scheme.

**Signing.** To sign a message $m$ , $M_i$ (having secret key $x_i$), picks a random secret $k \in Z_q$ and computes $r = g^k \pmod{p}$. It then outputs the signature as a pair $(c, s)$, where $c = H(m, r)$ and $s = k + rx_i \pmod{q}$.

**Verification.** In order to verify the above signature $(c, s)$, a recipient first computes the public key $y_i$ of the signer $M_i$ using its identity $id_i$ as $y_i = \prod_{j=0}^{t} (w_j)^{{id_i}^j} \pmod{p}$, and then verifies whether $c = H(m, r)$, where $r = g^s {y_i}^{-c} \pmod{p}$.

**Computational Complexity.** The signer needs to compute only one exponentiation, while the verifier requires one interpolation operation, two exponentiations and and two multiplications.

In the following theorem, we argue the security of SS-Sig. More precisely, we argue that SS-Sig remains secure against existential forgery under chosen message attack (CMA) [GMR84] in ROM as long as the discrete logarithm assumption holds. Notice that SS-Sig is different from regular signatures in the sense that the users generate signatures with related (and not independent) secret keys, and the adversary knows at most $t$ of these secret keys.

For clarity of our argument, we first recall the argument for security of the underlying Schnorr's signature scheme against CMA attack in ROM and discrete logarithm assumption; the simulator algorithm, on input $y = g^x$, can produce Schnorr's signatures on any $m$ by picking $s$ and $c$ at random in $\mathbb{Z}_q$, computing $r = g^s y^{-c} \pmod{p}$ and setting $H(m, r) = c$. This simulator can also translate the adversary's forgery into computing $dlog_g y$ as follows. It runs the adversary until the adversary outputs a forgery $(c, s)$ on some message $m$. Note that because $H$ is a random function, except for negligible probability, the adversary must ask to $H$ a query $(m, r)$ where $r = g^s y^{-c} \pmod{p}$, because otherwise it could not have guessed the value of $c = H(m, r)$. The simulator then rewinds the adversary, runs it again by giving the same answers to queries to $H$ until the query $(m, r)$, which it now answers with new randomness $c'$. If the adversary forges a signature on $m$ in this run, then, except for negligible probability, it produces $s'$ s.t. $r = g^{s'} y^{-c'} \pmod{p}$, and hence the simulator can now compute $dlog_g y = (s - s')/(c' - c) \pmod{q}$. One can show that if the adversary's probability of forgery is $\epsilon$, this simulation succeeds with probability $\epsilon^2/4q$: $O(\epsilon)$ probability that the adversary forges in the first run times the $O(\epsilon/qH)$ probability that it will forge on the second run and that it will choose to forge on the same $(m, r)$ query out of its q queries to H. We refer to [PS96] for the full proof.

**Theorem 11 (Security of SS-Sig).** *Under the DL assumption in ROM, as long as the adversary corrupts no more than t users,* SS-Sig *is secure against the chosen-message attack for every remaining uncorrupted user*

*Proof.* We prove the following claim: if there exists a polynomial time algorithm A, which on inputs the secret keys of $t$ corrupted users, is able to create an existential forgery in CMA model corresponding to an uncorrupted user, then there exists a polynomial time algorithm B, which can break the DL assumption in ROM.

We construct an algorithm B, which runs on input of a DL instance $y = g^x \pmod p$, and would translate the adversarial algorithm A into outputting $x$. We first assume that the adversary A corrupts $t$ players denoted by $M_1, M_2, \cdots, M_t$, w.l.o.g.

Note that in our multiple user scenario, the adversary A can request the signature oracle to sign chosen messages corresponding to any honest player. In other words, when A sends $(m, id_i)$ to the signature oracle, the oracle responds with a signature on message $m$ signed with $x_i$.

B picks $x_1, x_2, \cdots, x_t$ values corresponding to the secret keys of corrupted users, uniformly at random from $\mathbb{Z}_q$. It then sets $x_i = F(id_i)$, and employs appropriate Lagrange interpolation coefficients in the exponent to compute the public witnesses $g^{A_1}, \cdots, g^{A_t} \pmod p$, where $F(z) = x + A_1 z + \cdots + A_t z^t \pmod q$. Since, $x = \sum_{k=1}^{t} x_k l_k^i + x_i l_i^i \pmod q$, B can compute the public key $y_i$, corresponding to an honest player $M_i$ ($i \geq t+1$) as

$$y_i = (y / g^{\sum_{k=1}^{t} x_k l_k^i})^{1/l_i^i} \pmod p \tag{8.1}$$

B now runs A on inputs $x_1, x_2, \cdots, x_t$ and simulates the signature oracle on A's query $(m, id_i)$, by picking $s$ and $c$ at random in $\mathbb{Z}_q$, computing $r = g^s {y_i}^{-c} \pmod p$ and setting $H(m, r) = c$. A then outputs a forgery $(C, S)$ on some message $M$ corresponding to user $M_i$. Note that because $H$ is a random function, except for negligible probability, A must have asked to $H$ a query $(M, R)$ where $R = g^S {y_i}^{-C} \pmod p$, because otherwise it could not have guessed the value of $C = H(M, R)$. B then reruns A by giving the same answers to queries to $H$ until the query $(M, R)$, which it now answers with new randomness $C'$. If A outputs the forgery on the same message $M$, but this time for a different user $M_j$ ($i \neq j$) then, except for negligible probability, it produces $S'$ s.t. $R = g^{S'} {y_j}^{-C'} \pmod p$. B can now (using equation 8.1) compute

$$x = (S - S' + (C/l_i^i) \sum_{k=1}^{t} x_k l_k^i - (C'/l_j^j) \sum_{k=1}^{t} x_k l_k^j) / (C/l_i^i - C'/l_j^j) \pmod q$$

As in the security proof of Schnorr's Signatures, the probability of success of B would be $\epsilon^2/4q$, where $\epsilon$ represents the success probability of A and $q$ is the total number

of queries to $H()$.

□

### 8.3.5 SS-Enc: Secret Sharing based Encryption

We use Hashed ElGamal encryption scheme in the SS-Enc procedure.

**Encryption.** In order to encrypt a message $m$ for a user $M_i$ in the group, the encryptor computes the public key of $M_i$ as $y_i = \prod_{j=0}^{t}(w_j)^{id_i{}^j} \pmod p$, chooses a random $r \in \mathbb{Z}_q$ and then sends a pair $(c_1, c_2)$ to $M_i$, where $c_1 = g^r \pmod p$ and $c_2 = m \oplus H(y_i{}^r)$ ($\oplus$ denotes the bit-wise XOR operator).

**Decryption.** $M_i$ recovers the message by computing $c_2 \oplus H(c_1^{x_i})$ from the ciphertext $(c_1, c_2)$.

**Computational Complexity.** In the above procedure, the encryptor performs one interpolation and two exponentiation. The decryptor, on the other hand, needs to compute only a single exponentiation.

Before presenting the security argument for SS-Enc, we briefly discuss the indistinguishability notion [GM89]. Indistinguishability is defined as the following game: the adversary is first run on input of the public key and outputs two messages to be challenged upon. Next, one of these messages is encrypted and given to the adversary. The adversary is said to win this game if he can output which message was encrypted with non-negligible probability greater than half.

The above notion of indistinguishability was designed for a single user scenario, where multiple messages are being encrypted for one user. However, to capture the security of SS-Enc, where there are multiple users in the group and the messages are encrypted using *related* keys, we adopt the *multi-user* indistinguishability notion of Baudron et al. [BPS00] and Bellare et al. [BBM00]. In this notion, the adversarial game is as follows: first the adversary is given as input $n$ public keys $(pk_1, \cdots, pk_n)$ of all the users. The adversary then outputs two vectors of $n$ messages $M_0 = \{m_{01}, \cdots, m_{0n}\}$ and $M_1 = \{m_{11}, \cdots, m_{1n}\}$, which might be related or same, to be challenged upon. One of the message vectors $M_b$ ($b$ is 0 or 1) is then encrypted with $n$ public keys (the order of the encryption is preserved, i.e., $m_{bi}$ is encrypted with $pk_i$). The adversary is said to win the game if he can, with

probability non-negligibly greater than half, output which message was encrypted. It has been shown in [BBM00, BPS00] that an encryption scheme secure in the sense of single-user indistinguishabilty is also secure in the sense of multi-user indistinguishability.

Following is the security argument for SS-Enc based on a slightly modified multi-user indistinguishability notion, as described above (Basically, the adversary is only challenged for the encryptions of $n - t$ honest users in the group).

**Theorem 12 (Security of SS-Enc).** *Under the CDH assumption in ROM, as long as the adversary corrupts no more than t users,* SS-Enc *is secure in the sense of multi-user indistinguishability notion.*

*Proof.* As usual, the proof goes by contradiction, i.e., we proof that if there exists a polynomial time algorithm A, which on inputs the secret keys of $t$ corrupted users, is able to break the multi-user indistinguishability notion, then there exists a polynomial time algorithm B, which can break the CDH assumption in ROM.

We construct an algorithm B, which running on input of a CDH instance $U = g^u, V = g^v$, translates the algorithm A into outputting $g^{uv}$. As usual, we first assume that the adversary A corrupts $t$ players denoted by $M_1, M_2, \cdots, M_t$, w.l.o.g.

As in the security proof of SS-Sig, B picks $x_1, x_2, \cdots, x_t$ values corresponding to the secret keys of corrupted users, uniformly at random from $\mathbb{Z}_q$. It then sets $x_i = F(id_i)$, and employs appropriate Lagrange interpolation coefficients in the exponent to compute the public witnesses $g^{A_1}, \cdots, g^{A_t} \pmod p$, where $F(z) = u + A_1 z + \cdots + A_t z^t \pmod q$. Since, $u = \sum_{k=1}^{t} x_k l_k^i + x_i l_i^i \pmod q$, B can compute the public key $y_i$, corresponding to an honest player $M_i$ ($i \geq t+1$) using Equation 8.1.

To help the reader understand the construction of our translator algorithm B, we first recall the how the translator works in the security proof (under CDH and ROM) of single-user hashed ElGamal. The translator works as follows: on input of a CDH instance ($U = g^u, V = g^v$), it first runs the adversary on input $g^u$. The adversary outputs two messages $m_0, m_1$. The translator picks one message $m_b$ ($b = 0$ or 1) at random, and sends the encryption $(c_1, c_2)$ to the adversary, where $c_1 = V * g^r \pmod p$ and $c_2 = R$ ($r$ is a random value in $\mathbb{Z}_q$ and $R$ is a random pad of same length as the message). In the random oracle model, the only way the adversary can distinguish this encryption is by querying the random oracle on value $O = c_1^u = U^{r+v}$, which will be recorded by the translator, and used to compute $g^{uv} = OU^{-r}$. If there are a total of $q$ queries being made to the oracle, this

means that the probability of success of translator would be $1/q$ times the probability of success of the adversary.

Now, we are ready to describe the reduction based on our multi-user setting: B runs A on inputs the secret keys $x_1, \cdots, x_t$ corresponding to the corrupted users, and the public keys $y_{t+1}, \cdots, y_n$ of all honest ones. A outputs two vectors of $n-t$ messages $M_0 = \{m_{0i}\}$ and $M_1 = \{m_{1i}\}$, where $i = t+1, \cdots, n$, to be challenged upon. B then picks $M_b$ ($b$ is 0 or 1) and sends to A the vector $\{(V * g^{r_i}, R_i)\}$, where $r_i$ is a random value in $\mathbb{Z}_q$, and $R_i$ is a random pad equally long as the message $m_{bi}$, for $i = t+1, \cdots, n$. The only possibility for A to win this game, is by querying the random oracle on at least one of the value $O = (V * g^{r_j})^{x_j}$, for some $j \in \{t+1, \cdots, n\}$. B records this value, and assuming that it corresponds to $M_j$, it computes $g^{uv}$ as follows:

$$u = \sum_{k=1}^{t} x_k l_k^j + x_j l_j^j \pmod{q}$$

This implies that

$$g^{uv} = g^{v \sum_{k=1}^{t} x_k l_k^j} g^{v x_j l_j^j} \pmod{q}$$

and

$$g^{uv} = V^{\sum_{k=1}^{t} x_k l_k^j} V^{x_j l_j^j} \pmod{p}$$

Since, $O = (V * g^{r_j})^{x_j}$, this means $V^{x_j} = O {y_j}^{-r_j}$, and therefore,

$$g^{uv} = V^{\sum_{k=1}^{t} x_k l_k^j} O {y_j}^{-r_j l_j^j} \pmod{p}$$

Given that there are a total of $q$ queries to the random oracle, the probability of success of B would be probability of success of A times $1/q(n-t)$, as only one query will yield correct $g^{uv}$ value and each query might correspond to one $j$ value in $\{t+1, n\}$.

□

**Remark: Extension to Chosen Ciphertext Security.** The hybrid encryption techniques for extending standard hashed ElGamal to chosen ciphertext security (refer to [BBP04], [FO99]) can be used to achieve chosen ciphertext security for the SS-Enc scheme.

## 8.4 Comparison with ID-based Cryptography

As previously pointed out in the introduction section, our proposed scheme can be viewed as an identity-based cryptosystem based on threshold assumption. Refer to Section 2.9, where we provided a brief background of ID-based cryptography. Basically, a trusted center provides each user with a secret value (VSS share in our case) derived from the unique identifier of the user, and publishes the VSS information as its public key. Knowing the identifier of a particular user and also the public key of the trusted center, one can send encrypted messages and verify signatures. This is equivalent to IBE [BF01], and ID-based signatures [CC03], apart from the fact that our scheme becomes insecure if there are more than a threshold of collusions or corruptions. However, unlike other ID-based schemes, our proposal is based on standard (DL-based) assumptions. Moreover, for reasonable group sizes and threshold values, our scheme is much more efficient than these prior ID-based schemes, which require costly computations (such as scalar point multiplications, map-to-point operations and bilinear mappings [BF01]) in elliptic-curves. For example, for a group size of around 100, and threshold of 10 (10% of group size), the encryption in our scheme would require less than 70 squarings, less than 70 modular multiplications, and only 2 modular exponentiations. The decryption would just require 1 exponentiation. On the other hand, IBE requires 1 map-to-point operation, 2 scalar point multiplications, and 1 bilinear mapping, for encryption, and 1 bilinear mapping for decryption. It is well-known that for appropriate security parameters, the IBE computations are extremely costly (e.g., a bilinear mapping takes around 80ms, scalar point multiplication costs around 30 ms, while a single modular exponentiation is only a few milliseconds on fast processors). Refer to, e.g., [STY04] for details regarding these cost comparisons.

# Bibliography


[BA03] Kenneth Barr and Krste Asanovic. Energy Aware Lossless Data Compression. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 259–274, 2000.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid encryption problem. In *EUROCRYPT'04*, volume 3027 of *LNCS*, pages 171–188, 2004.

[BDF98] D. Boneh, G. Durfee, and Y. Frankel. An Attack on RSA Given a Small Fraction of the Private Key Bits. In *ASIACRYPT'98*, volume 1514 of *LNCS*, pages 25–34, 1998.

[BDG+04] Dirk Balfanz, Glenn Durfee, Rebecca E. Grinter, Diana K. Smetters, and Paul Stewart. Network-in-a-box: How to set up a secure wireless network in under a minute. In *USENIX Security Symposium*, pages 207–222, 2004.

[BDS+03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret Handshakes from Pairing-Based Key Agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196, May 2003.

[BEPW02] Peter Biddle, Paul England, Marcus Peinado, and Bryan Willman. The darknet and the future of content distribution. In *ACM Workshop on Digital Rights Management*, November 2002.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *EUROCRYPT '03*, volume 2656 of *LNCS*, pages 416–432, 2003.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *ASIACRYPT'01*, volume 2248 of *LNCS*, pages 514–532. IACR, 2001.

[BM03] J. Blomer and A. May. New Partial Key Exposure Attacks on RSA. In Dan Boneh, editor, *CRYPTO '03*, volume 2729 of *LNCS*, pages 27–43. IACR, 2003.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Annual Symposium on the Theory of Computing*, pages 1–10, 1988.

[Bol03] A. Boldyreva. Efficient Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-group Signature Scheme. In *Proceedings of International Workshop on Practice and Theory in Public Key Cryptography*, volume 2567 of *LNCS*, pages 31–46, 2003.

[Bou00] Fabrice Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 431–444, 2000.

[Boy89] Colin Boyd. Digital Multisignatures. In *Cryptography and Coding*, pages 241–246. Claredon Press, May 1989.

[BPS00] Olivier Baudron, David Pointcheval, and Jacques Stern. Extended notions of security for multicast public key cryptosystems. In *International Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 499–511, 2000.

[BR93] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416, 1996.

[Bra84] Gabriel Bracha. An Asynchronous $\lfloor (n-1)/3 \rfloor$-resilient Consensus Protocol. In *ACM Symposium on Priniciples of Distributed Computing*, pages 154–162, August 1984.

[BSH+92] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-Secure Key Distribution for Dynamic Conferences. In *Advances in Cryptology - CRYPTO'92*, pages 471–486, 1992.

[BSSW02] Dirk Balfanz, Diana Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Network and Distributed System Security Symposium*. The Internet Society, 2002.

[BT99] Fabrice Boudot and Jacques Traor. Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery. In *Second International Conference on Information and Communication Security (ICICS)*, pages 87–102, November 1999.

[CC03] J. Cha and J. Cheon. An ID-based signature from Gap-Diffie-Hellman Groups. In *Proceedings of International Workshop on Practice and Theory in Public Key Cryptography*, volume 2567 of *LNCS*, pages 18–30, 2003.

[CFT98] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy Come - Easy Go Divisible Cash. In *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 561–575, 1998.

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 98–115, 1999.

[CH89] R.A. Croft and S.P. Harris. Public-key Cryptography and Re-usable Shared Secrets. In *Cryptography and Coding*, pages 189–201. Claredon Press, May 1989.

[CiM05] RVSI Acuity CiMatrix. Data Matric Barcodes, 2005. Available at http://www.rvsi.net/.

[CJT04] Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik. Secret handshakes from ca-oblivious encryption. In *ASIACRYPT*, pages 293–307, 2004.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT'01*, pages 453–474. Springer-Verlag, 2001.

[CM99a] Jan Camenisch and Markus Michels. Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 107–122, 1999.

[CM99b] Jan Camenisch and Markus Michels. Separability and Efficiency for Generic Group Signature Schemes. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 106–121, 1999.

[Col] CollabNet Project. http://www.collab.net.

[Cry97] *CRYPTO '97*, volume 1294 of *LNCS*, 1997.

[CS99] Ronald Cramer and Victor Shoup. Signature schemes based on the strong rsa assumption. In *ACM Conference on Computer and Communications Security*, pages 46–51, 1999.

[CSY05] Claude Castelluccia, Nitesh Saxena, and Jeong Hyun Yi. Self-Configurable Key Pre-distribution in Mobile Ad Hoc Networks. In *IFIP Networking Conference*, May 2005.

[DA99] Tim Dierks and Christopher Allen. The TLS protocol version 1.9. Internet Engineering Task Force, RFC 2246, January 1999.

[Dam00] Ivan Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 418–430, 2000.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to Share a Function Securely. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 522–533, Montreal, Canada, 1994.

[Des87] Yvo Desmedt. Society and Group Oriented Cryptosystems. In *CRYPTO '87*, number 293 in LNCS, pages 120–127, 1987.

[DF90] Yvo Desmedt and Yair Frankel. Threshold Cryptosystems. In *CRYPTO '89*, volume 435 of *LNCS*, pages 307–315, 1990.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. In *ASIACRYPT'02*, volume 2501 of *LNCS*, pages 125–142. Springer, 2002.

[DLRS01] B. Dahill, B. Levine, E. Royer, and C. Shields. A secure routing protocol for ad hoc networks. Technical Report UM-CS-2001-037, University of Massachusetts, August 2001.

[ElG99] ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Transactions in Information Theory (IT-31)*, pages 469–472, 1999.

[FD92] Yair Frankel and Yvo Desmedt. Parallel Reliable Threshold Multisignature. Technical Report TR-92-04-02, Dept. of EE and CS, U. of Winsconsin, April 1992.

[Fel87] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *28th Symposium on Foundations of Computer Science (FOCS)*, pages 427–437, 1987.

[FGMY97a] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Optimal-Resilience Proactive Public-Key Cryptosystems. In *38th Symposium on Foundations of Computer Science (FOCS)*, pages 384–393, 1997.

[FGMY97b] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Proactive RSA. In *Crypto'97*, volume 1294 of *LNCS*, pages 440–454, 1997.

[FGY96] Yair Frankel, Peter Gemmell, and Moti Yung. Witness-based Cryptographic Program Checking and Robust Function Sharing. In *Proc. 28th ACM Symp. on Theory of Computing*, pages 499–508, Philadelphia, 1996.

[FMR99] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. In *IEEE Transactions on Information Theory*, volume 45, pages 1717–1719, July 1999.

[FMY99a] Yair Frankel, Philip MacKenzie, and Moti Yung. Adaptively-Secure Distributed Threshold Public Key Systems. In *Proceedings of ESA 99*, 1999.

[FMY99b] Yair Frankel, Philip MacKenzie, and Moti Yung. Adaptively-Secure Optimal-Resilience Proactive RSA. In *ASIACRYPT'99*, volume 1716 of *LNCS*, 1999.

[FMY01] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Adaptive security for the additive-sharing based proactive rsa. In *Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 240–263, 2001.

[FO97] E. Fujisaki and T. Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In Crypto97 [Cry97], pages 16–30.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554, 1999.

[G+02] Christian Gehrmann et al. SHAMAN Deliverable: Detailed Technical Specification of Mobile Terminal System Security, May 2002. Available at www.isrc.rhul.ac.uk/shaman/docs/d10v1.pdf.

[GJKR96a] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust and Efficient Sharing of RSA Functions. In *CRYPTO '96*, volume 1109 of *LNCS*, pages 157–172, 1996.

[GJKR96b] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust Threshold DSS Signatures. In *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 354–371, 1996.

[GJKR99a] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust Threshold DSS Signatures. *Information and Computation*, 164:54–84, 1999.

[GJKR99b] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Secure Distributed Key Generation for Discrete Log Based Cryptosystems. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 295–310, 1999.

[GJKR03] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure applications of pedersen's distributed key generation protocol. In *CT-RSA*, pages 373–390, 2003.

[GKS04] Gunnar Gaubatz, Jens-Peter Kaps, and Berk Sunar. Public key cryptography in sensor networks - revisited. In *ESAS*, pages 2–18, 2004.

[GM89] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences,*, 28:270–299, 1989.

[GMN04] Christian Gehrmann, Chris J. Mitchell, and Kaisa Nyberg. Manual authentication for wireless devices. *RSA CryptoBytes*, 7(1):29 – 37, Spring 2004.

[GMR84] S. Goldwasser, S. Micali, and R. L. Rivest. A "paradoxical" solution to the signature problem. In *IEEE Annual Symposium of Foundations of Computer Science (FOCS'84)*, pages 441–448, 1984.

[GSS+06] Michael T. Goodrich, Michael Sirivianos, John Solis, Gene Tsudik, and Ersin Uzun. Loud and Clear: Human-Verifiable Authentication Based on Audio. In *International Conference on Distributed Computing Systems (ICDCS)*, July 2006. Available at `http://www.ics.uci.edu/ccsp/lac`.

[Hal95] N. Haller. The S/KEY One-Time Password System. RFC 1760, IETF, February 1995.

[Ham03] Anita Hamilton. Playing in the dark: The heat is on, and music swappers are taking their business underground. *TIME Magazine*, September 29 2003.

[Han02] Stephen R. Hanna. Configuring Security Parameters in Small Devices, July 2002. draft-hanna-zeroconf-seccfg-00.

[Hil01] R. Hills. Sensing for danger. Science Technology Report, July/August 2001. Available at http://www.llnl.gov/str/JulAug01/Hills.html.

[HJJ+97a] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive Public Key and Signature Systems. In *ACM Conference on Computers and Communication Security*, 1997.

[HJJ+97b] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive Public Key and Signature Systems. In *ACM Conference on Computers and Communication Security*, pages 100–110, 1997.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive Secret Sharing, Or How To Cope With Perpetual Leakage. In *CRYPTO '95*, volume 963 of *LNCS*, pages 339–352, 1995.

[HJP02] Y.-C. Hu, D. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *IEEE Workshop on Mobile Computing Systems and Applications*, June 2002.

[Hoe04] Jaap-Henk Hoepman. The ephemeral pairing problem. In *Proc. Int. Conf. Financial Cryptography*, number 3110 in Lecture Notes in Computer Science, pages 212–226. Springer, 2004.

[HPJ02] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth ACM International Conference on Mobile Computing and Networking (Mobicom 2002)*, September 2002.

[JS05] Stanisław Jarecki and Nitesh Saxena. Further Simplifications in Proactive RSA Signatures. In *Theory of Cryptography Conference (TCC'05)*, February 2005.

[JS06] Stanisław Jarecki and Nitesh Saxena. Encryption-based Authenticated Key Agreement using Short Authenticated Strings. In *Submission*, July 2006.

[JSY04] Stanislaw Jarecki, Nitesh Saxena, and Jeong Hyun Yi. An Attack on the Proactive RSA Signature Scheme in the URSA Ad Hoc Network Access Control Protocol. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 1–9, October 2004.

[KLX+02] Jiejun Kong, Haiyun Luo, Kaixin Xu, Daniel Lihui Gu, Mario Gerla, and Songwu Lu. Adaptive Security for Multi-level Ad-hoc Networks. In *Journal of Wireless Communications and Mobile Computing (WCMC)*, volume 2, pages 533–547, 2002.

[KMT03] Yongdae Kim, Daniele Mazzocchi, and Gene Tsudik. Admission Control in Peer Groups. In *IEEE International Symposium on Network Computing and Applications (NCA)*, April 2003.

[Kob95] Neal I. Koblitz. *A Course in Number Theory and Cryptography.* Springer-Verlag. 1995. ISBN 0-387-94293-9.

[KZL+01] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing Robust and Ubiquitous Security Support for MANET. In *IEEE 9th International Conference on Network Protocols (ICNP)*, pages 251–260, 2001.

[LAN05] Sven Laur, N. Asokan, and Kaisa Nyberg. Efficient mutual data authentication based on short authenticated strings. IACR Cryptology ePrint Archive: Report 2005/424 available at `http://eprint.iacr.org/2005/424`, November 2005.

[LKZ+04] Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu, and Lixia Zhang. URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks. In *IEEE/ACM Transactions on Networking (ToN)*, to appear 2004. Available on-line at http://www.cs.ucla.edu/wing/publication/publication.html.

[LL00] H. Luo and S. Lu. Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks. Technical Report TR-200030, Dept. of Computer Science, UCLA, 2000. Available online at http://citeseer.ist.psu.edu/luo00ubiquitous.html.

[LN03] Donggang Liu and Peng Ning. Establishing Pairwise Keys in Distributed Sensor Networks. In *ACM Conference on Computers and Communication Security*, pages 52–61, 2003.

[Lu] Songwu Lu. Comments on Recent Advances in Cryptoanalysis of URSA. A draft communicated to the authors by email by Songwu Lu on August 16th, 2004.

[LZK+02] Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, and Lixia Zhang. Self-securing Ad Hoc Wireless Networks. In *Seventh IEEE Symposium on Computers and Communications (ISCC '02)*, 2002.

[MAM+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol - OCSP. RFC 2560, IETF, June 1999.

[MNSS87] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. Technical report, MIT, 1987.

[Möl01] Bodo Möller. Algorithms for multi-exponentiation. In *Selected Areas in Cryptography*, pages 165–180, 2001.

[MP03] D. Micciancio and E. Petrank. Simulatable Commitments and Efficient Concurrent Zero-Knowledge. In *EUROCRYPT'03*, volume 2656 of *LNCS*, pages 140–159, 2003.

[MPR05] Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proc. 2005 IEEE Symposium on Security and Privacy*, pages 110–124. IEEE, 2005.

[MSSU04] Anil Madhavapeddy, David Scott, Richard Sharp, and Eben Upton. Using camera-phones to enhance human-computer interaction. In *Sixth International Conference on Ubiquitous Computing (Adjunct Proceedings: Demos)*, 2004.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press series on discrete mathematics and its applications. 1997. ISBN 0-8493-8523-7.

[MW96] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In *CRYPTO'96*, volume 1109 of *LNCS*, pages 268–282, 1996.

[MY04] Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In *EUROCRYPT*, pages 382–400, 2004.

[NIS91] NIST. Digital signature standard (DSS). Technical Report 169. National Institute for Standards and Technology, August 30, 1991.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distibuted Pseudo-Random Functions and KDCs. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 327–346, May 1999.

[NTY03] Maithili Narasimha, Gene Tsudik, and Jeong Hyun Yi. On the Utility of Distributed Cryptography in P2P and MANETs: The Case of Membership Control. In *IEEE International Conference on Network Protocol (ICNP)*, pages 336–345, November 2003.

[Nyq28] Harry Nyquist. Certain topics in telegraph transmission theory. *Transasctions of the American Institute of Electrical Engineers (AIEE)*, 47:617–644, 1928.

[OMR01] Kazuo Ohta, Silvio Micali, and Leonid Reyzin. Accountable Subgroup Multisignatures. In *ACM Conference on Computer and Communications Security*, pages 245–254, November 2001.

[Operg] OpenSSL Project, `http://www.openssl.org/`.

[OY91] Rafael Ostrovsky and Moti Yung. How to Withstand Mobile Virus Attacks. In *10th ACM Symp. on the Princ. of Distr. Comp.*, pages 51–61, 1991.

[Ped91a] T. Pedersen. Non-interactive and Information-theoretic Secure Verifiable Secret Sharing. In *Crypto 91*, volume 576 of *LNCS*, pages 129–140, 1991.

[Ped91b] Torben P. Pedersen. A threshold cryptosystem without a trusted party. In D.W. Davies, editor, *EUROCRYPT '91*, number 547 in LNCS, pages 552–526. IACR, 1991.

[Peeac] Peer Group Admission Control Project, `http://sconce.ics.uci.edu/gac`.

[PFTV92] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1992. ISBN 0-521-43108-5.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398, 1996.

[PV06] Sylvain Pasini and Serge Vaudenay. An optimal non-interactive message authentication protocol. In *CT-RSA*, 2006.

[PVar] Sylvain Pasini and Serge Vaudenay. An optimal non-interactive message authentication protocol. In *CT-RSA*, 2006 (to appear).

[Rab98] Tal Rabin. A Simplified Approach to Threshold and Proactive RSA. In *CRYPTO '98*, volume 1462 of *LNCS*, pages 89 – 104, 1998.

[RG04] Michael Rohs and Beat Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. In Alois Ferscha, Horst Hoertner, and

Gabriele Kotsis, editors, *Advances in Pervasive Computing*, pages 265–271, Vienna, Austria, April 2004. Austrian Computer Society (OCG).

[RRSW97] Carl Rigney, Allan C. Rubens, William Allen Simpson, and Steve Willens. Remote Authentication Dial In User Service (RADIUS). RFC 2058, IETF, January 1997.

[RS60] Irving S. Reeed and Gustave Solomon. Polynomial codes over certain finite fields. In *Journal of Society for Industrial and Applied Mathematics*, 1960.

[SA99] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols Workshop*, volume 1796 of *Lecture Notes in Computer Science*, pages 172–194. Springer, 1999.

[Sax06] Nitesh Saxena. Public key cryptography sans certificates in ad hoc networks. In *Applied Cryptography and Network Security (ACNS)*, pages 375–389, 2006.

[Sch91] Claus P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[SEKA06] Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Secure device pairing based on a visual channel (short paper). In *IEEE Symposium on Security and Privacy (ISP'06)*, May 2006.

[Sha79] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[Sho00] Victor Shoup. Practical Threshold Signatures. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 207–220, 2000.

[STW00a] M. Steiner, G. Tsudik, and M. Waidner. Cliques: A new approach to group key agreement. *IEEE Transactions on Parallel and Distributed Systems*, August 2000.

[STW00b] Michael Steiner, Gene Tsudik, and Michael Waidner. Key agreement in dynamic peer groups. In *IEEE Transactions on Parallel and Distributed Systems*, July 2000.

[STY03] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Admission Control in Peer-to-Peer: Design and Performance Evaluation. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 104–114, October 2003.

[STY04] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Identity-based Access Control for Ad-Hoc Groups. In *International Conference on Information Security and Cryptology (ICISC)*, December 2004.

[STY05] N. Saxena, G. Tsudik, and J. H. Yi. Efficient node admission for short-lived mobile ad hoc networks. In *International Conference on Networking Protocols (ICNP)*, November 2005.

[Theml] The AUTONET Project, `http://www.its.uci.edu/~mcnally/mgm-autonet.html`.

[Vau05] Serge Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Advances in Cryptology - CRYPTO 2005*, number 3621 in Lecture Notes in Computer Science, pages 309 – 326. Springer Verlag, 2005.

[Woo05] Simon Woodside. Read real-world hyperlinks with a camera phone, February 2005. Available at http://semacode.org.

[Yi05] Jeong Hyun Yi. Membership Management for Dynamic Peer Groups, University of California, Irvine PhD thesis, 2005.

[ZH99] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6):24–30, 1999.

[ZSvR02] Lidong Zhou, Fred Schneider, and Robbert van Renesse. COCA: A secure distributed on-line certification authority. *ACM Transactions on Computer Systems*, 20(4):329–368, November 2002.

# Appendices

## A Zero Knowledge Proof of Partial Signature Correctness in the New Proactive RSA

For the purpose of proving the correctness of partial signatures in the proposed proactive RSA scheme, we apply the zero knowledge proofs for the equality of committed numbers in two different groups and for the range of a committed number. All these proofs are honest verifier zero-knowledge and can be converted either into standard zero-knowledge proof either at the expense of 1-2 extra rounds using techniques of [Dam00, DF02, MP03], or into a non-interactive proof in the random oracle model using the Fiat-Shamir heuristic. We adopt the notation of [CM99a] for representing zero-knowledge proof of knowledge protocols. For example, $\boxed{ZKPK\{x : R(x)\}}$ represents a ZKPK protocol for proving possession of a secret $x$ which satisfies statement $R(x)$. In the protocols to follow, $u$ ($\geq 80$) and $v$ ($\geq 40$) are security parameters.

**Protocol for proving the correctness of a partial signature:**

$$\boxed{ZKPK\{d_i, d_i' : w_{i0} = g^{d_i} h^{d_i'} \pmod{p} \ \wedge \ s_i = m^{d_i} \pmod{N} \ \wedge \ d_i \in [0, q-1]\}}$$

The signer (or prover) $M_i$ proves to the verifier the possession of its correct secret share $d_i$ by using the following zero-knowledge proof system. The verifier can either be one of the players or an outsider who has inputs $w_{i0}, g, h, p, s_i, m, N, q$. All the protocols run in parallel, and failure of these protocols at any stage implies the failure of the whole proof.

1. The verifier follows the setup procedure of the Damgard-Fujisaki-Okamoto commitment scheme [FO97, DF02], e.g. it picks a safe RSA modulus $n$ and two elements $G, H$ in $\mathbb{Z}_n^*$ whose orders are greater than 2. (We refer to [DF02] for the details of this commitment scheme.) If $N$ is a safe RSA modulus then set $n = N$, $G = (G')^2 \bmod N$,

$H = (H')^2 \bmod N$ for random $G', H' \in \mathbb{Z}_n^*$.

2. The prover computes the commitment $C = G^{d_i} H^R \pmod n$, where $R$ is picked randomly from $[0, 2^v(q-1)]$ and uses **Protocol (1)** (see below), by substituting $(x, x'_1, x'_2, g_1, h_1, g_2, h_2, n_1, n_2, w_1, w_2, b, b')$ with $(d_i, R, d'_i, G, H, g, h, n, p, C, w_{i0}, q-1, 2^v(q-1))$, respectively, to execute:
$ZKPK\{d_i, R, d'_i : C = G^{d_i} H^R \pmod n \ \wedge \ w_{i0} = g^{d_i} h^{d'_i} \pmod p\}$.

3. The prover then uses **Protocol (1)** (see below), by substituting $(x, x'_1, x'_2, g_1, h_1, g_2, h_2, n_1, n_2 w_1, w_2, b, b')$ with $(d_i, R, 0, G, H, m, m, n, N, C, s_i, q-1, 2^v(q-1))$, respectively, to execute:
$ZKPK\{d_i, R : C = G^{d_i} H^R \pmod n \ \wedge \ s_i = m^{d_i} \pmod N\}$.

4. The prover uses **Protocol (2)** (see below), by substituting $(x, x', b)$ with $(d_i, R, q-1)$, respectively, to execute:
$ZKPK\{d_i, R : C = G^{d_i} H^R \pmod n \ \wedge \ d_i \in [0, q-1]\}$

**Protocol (1).** $\boxed{ZKPK\{x, x'_1, x'_2 : w_1 = g_1^x h_1^{x'_1} \pmod{n_1} \ \wedge \ w_2 = g_2^x h_2^{x'_2} \pmod{n_2}\}}$

*Assumption*: $x, x'_2 \in [0, b]$ and $x'_1 \in [0, b']$.

This protocol is from [CM99a], [Bou00], and is perfectly complete, honest verifier statistical zero-knowledge and sound under the strong RSA assumption [FO97] with the soundness error $2^{-u+1}$, given than $(g_1, h_1, n_1)$ is an instance of the Damgard-Fujisaki-Okamoto commitment scheme [FO97, DF02].

1. The prover picks random $r \in [1, \ldots, 2^{u+v}b - 1]$ , $\eta_1 \in [1, \ldots, 2^{u+v}b' - 1]$, $\eta_2 \in [1, \ldots, 2^{u+v}b - 1]$ and computes $W_1 = g_1^r h_1^{\eta_1} \pmod{n_1}$ and $W_2 = g_2^r h^{\eta_2} \pmod{n_2}$. It then sends $W_1$ and $W_2$ to the verifier $V$.

2. The verifier selects a random $c \in [0, \ldots, 2^u - 1]$ and sends it back to the prover.

3. The prover responds with $s = r + cx$ (in $\mathbb{Z}$), $s_1 = \eta_1 + cx'_1$ (in $\mathbb{Z}$) and $s_2 = \eta_2 + cx'_2$ (in $\mathbb{Z}$)

4. The verifier verifies as $g_1^s h_1^{s_1} = W_1 {w_1}^c \pmod{n_1}$ and $g_2^s h_2^{s_2} = W_2 {w_2}^c \pmod{n_2}$.

**Protocol (2).** $\boxed{ZKPK\{x, x' : C = G^x H^{x'} \pmod n \ \wedge \ x \in [0, b]\}}$

*Assumption*: $x \in [0, b]$ and $x' \in [0, 2^v b]$.

This protocol (from [Bou00]) is an exact range proof, honest verifier statistical zero-knowledge, complete with a probability greater than $1 - 2^{-v}$, and sound under the strong RSA assumption given that $(G, H, n)$ is an instance of the Damgard-Fujisaki-Okamoto commitment scheme, similarly as in protocol (1).

1. The prover sets $T = 2(u + v + 1) + |b|, X = 2^T x, X' = 2^T x', \beta = 2^{u+v+1}\sqrt{b}$ and $C_T = G^X H^{X'} \pmod{n}$.

2. The prover uses **Protocol (3)** (see below), by substituting $(x, x', com, B, \gamma)$ with $(X, X', C_T, 2^T b, 2^{T/2}\beta)$, respectively, to execute the following (note that $X \in [0, 2^T b]$):
   $ZKPK\{X, X' : C_T = G^X H^{X'} \pmod{n} \ \wedge \ X \in [-2^{T/2}\beta, 2^T b + 2^{T/2}\beta]\}$

   Proving that $X \in [-2^{T/2}\beta, 2^T b + 2^{T/2}\beta]$ implies that $x \in [0, b]$, since $2^{T/2}\beta < 2^T$.

**Protocol (3).** $\boxed{ZKPK\{x, x' : com = G^x H^{x'} \pmod{n} \ \wedge \ x \in [-\gamma, B + \gamma]\}}$
Here $\gamma = 2^{u+v+1}\sqrt{B}$.

*Assumption*: $x \in [0, B]$ and $x' \in [0, 2^v B]$.

This proof was proposed in [Bou00] and is honest verifier statistical zero-knowledge, complete with a probability greater than $1 - 2^{-v}$, and sound under the strong RSA assumption just like protocol (2).

1. The prover executes $ZKPK\{x, x' : com = G^x H^{x'} \pmod{n}\}$

2. The prover sets $x_1 = \lfloor\sqrt{x}\rfloor, x_2 = x - x_1^2, \hat{x}_1 = \lfloor\sqrt{B - x}\rfloor, \hat{x}_2 = B - x - \hat{x}_1^2$, and chooses randomly $r_1, r_2, \hat{r}_1, \hat{r}_2$ in $[0, 2^v B]$, such that $r_1 + r_2 = x'$ and $\hat{r}_1 + \hat{r}_2 = -x'$.

3. The prover computes new commitments $e_1 = G^{x_1^2} H^{r_1} \pmod{n}, \hat{e}_1 = G^{\hat{x}_1^2} H^{\hat{r}_1} \pmod{n}$, $e_2 = G^{x_2} H^{r_2} \pmod{n}, \hat{e}_2 = G^{\hat{x}_2} H^{\hat{r}_2} \pmod{n}$, and sends $e_1$ and $\hat{e}_1$ to the verifier.

4. The verifier computes $e_2 = com/e_1 \pmod{n}$ and $\hat{e}_2 = G^B/(com * \hat{e}_1) \pmod{n}$.

5. The prover uses **Protocol (4)** (see below), by substituting $(x, x', com_{sq})$ with $(x_1, r_1, e_1)$ and then with $(\hat{x}_1, \hat{r}_1, \hat{e}_1)$, to execute the following:
   $ZKPK\{x_1 : e_1 = G^{x_1^2} H^{r_1} \pmod{n}\}$
   $ZKPK\{\hat{x}_1 : \hat{e}_1 = G^{\hat{x}_1^2} H^{\hat{r}_1} \pmod{n}\}$
   This proves that $e_1$ and $\hat{e}_1$ hide a square.

6. The prover uses **Protocol (5)** (see below), by substituting $(x, x', com_2, B_1)$ with $(x_2, r_2, e_2, 2\sqrt{B})$, respectively and then with $(\hat{x}_2, \hat{r}_2, \hat{e}_2, 2\sqrt{B})$, respectively, to execute the following (note that $x_2$ and $\hat{x}_2 \in [0, 2\sqrt{B}]$):
   $ZKPK\{x_2 : e_2 = G^{x_2} H^{r_2} \pmod{n} \ \wedge \ x_2 \in [-\gamma, \gamma]\}$
   $ZKPK\{\hat{x}_2 : \hat{e}_2 = G^{\hat{x}_2} H^{\hat{r}_2} \pmod{n} \ \wedge \ \hat{x}_2 \in [-\gamma, \gamma]\}$
   This proves that $e_2$ and $\hat{e}_2$ hide numbers belonging to $[-\gamma, \gamma]$.

   Steps 2, 5 and 6 above, imply that $x \in [-\gamma, B + \gamma]$.

**Protocol (4).** $\boxed{ZKPK\{x, x' : com_{sq} = G^{x^2} H^{x'} \pmod{n}\}}$

This protocol first appeared in [FO97], generalized (and corrected) in [DF02] and proves that a committed number is a square. The protocol is honest verifier statistical zero-knowledge, perfectly complete, and sound under the strong RSA assumption just like protocol (2).

**Protocol (5).** $\boxed{ZKPK\{x, x' : com_2 = G^x H^{x'} \pmod{n} \ \wedge \ x \in [-2^{u+v} B_1, 2^{u+v} B_1]\}}$

*Assumption*: $x \in [0, B_1]$, and $x' \in [0, 2^v B_1]$.

This proof was proposed in [CFT98], allows a prover to prove the possession of a discrete logarithm $x$ lying in the range $[-2^{u+v} B_1, 2^{u+v} B_1]$ given $x$ which belongs to a smaller interval $[0, B_1]$. Using the commitment scheme of [FO97, DF02], this proof is honest verifier statistical zero-knowledge, complete with a probability greater than $1 - 2^{-v}$, and sound under the strong RSA assumption with soundness error $2^{-u+1}$.

# B Difficulty in Extending the General Compilation Theorem of [PVar]

We give some intuition for the claim we make in the introduction, namely that the general composition theorem given by Passini and Vaudenay [PVar], for transforming KA protocols to SAS-AKA protocols given any SAS-MCA scheme, appears difficult to extend to KA schemes which share state between sessions. Consider a 2-round (non-authenticated) KA protocol. Note that both Diffie-Hellman and encryption-based KA protocols fall into this category. To save round complexity in the compiled SAS-AKA protocol, we would like to make the two messages generated by the KA protocol, $m_i$ of the initiator $P_i$ and $m_j$ of the responder $P_j$, inputs to the SAS-MCA scheme, where $P_i$ goes first, and $m_j$ is possibly based on $m_i$. (The known 3-round SAS-MCA protocols allow the responder's message $m_j$ to be picked in the second round.)

Note that at the time $P_j$ computes his response $m_j$, following the algorithm of the KA protocol on the received message $m_i$, the message $m_i$ is not yet authenticated by $P_j$. If the KA protocol does not share state between sessions, if $P_j$ computes $m_j$ computed on adversarially-chosen $\hat{m}_i$, this can possibly endanger only the current session, and since the SAS-MCA subprotocol will eventually let $P_j$ know that $\hat{m}_i$ was not sent by $P_i$, $P_j$ will reject in this session anyway. (And so will $P_i$, if we assume that $m_j$ always contains imitators own message, or its hash.)

However, if $P_j$ keeps a shared state between sessions then the information $P_j$ reveals in $m_j$, computed on *unauthenticated* message $\hat{m}_i$, could potentially reveal some secret information that endangers all other sessions of player $P_j$, or at least all other sessions between $P_j$ and $P_i$. One can easily create a perverse example of a KA protocol which is secure over authenticated channels and becomes insecure in this case. For example, take any Key Agreement protocol KA secure over authenticated links and let each player $P_j$ keep an additional long-term secret $s_j$ and compute a per-partner secret $k_i = F_{s_j}(< P_i >)$ where $F$ is a pseudorandom function. If the initiator's message $m_i$ contains a special symbol $\perp$, $P_j$ sends $m_j = k_i$ to $P_i$. Otherwise, $P_j$ follows the KA protocol but also at the end encrypts the resulting session key under this key $k_i$. In the authenticated link model, considering a static adversary model, an honest player never sends the $\perp$ symbol. If the encryption is semantically secure, encrypting the session key does not endanger its security. Also, if $F$ is a PRF, if the adversary learns other values of the $F$ function, under indices corresponding to

the corrupt players, again does not reveal any information about the values of $F$ on indices corresponding to honest players. On the other hand, this protocol is an insecure SAS-AKE protocol, because an adversary can inject message $\hat{m}_i = \perp$ on the insecure channel on behalf of any player $P_i$, and learn all the session keys between $P_j$ and $P_i$ in this way.

# C Sketch of an Improved Security Analysis of the Enc-MCA Protocol

As mentioned in footnote 5 on page 63, the security bound we show for the Enc-MCA protocol in Theorem 10 is not optimal, as it wastes one bit of the SAS bandwidth. However, this security claim can be improved to show that a probability of an attack against the Enc-MCA protocol can be bounded by $2^{-k} + O(\epsilon_C, \epsilon_E)$, under the same assumptions.

Here is a sketch of the improved security analysis. Recall the proof of theorem 10 which constructs two reductions, $\mathcal{B}_C$ and $\mathcal{B}_E$, attacking the OW-ExA commitment game and OW-R-CCA encryption game, respectively. In the improved analysis, we let $\mathcal{B}_C$ and $\mathcal{B}_E$ attack the "multi-challenge" versions of both the OW-ExA commitment game and OW-R-CCA encryption game. In the "multi-challenge" versions of either the OW-ExA or the OW-R-CCA security notions the adversary gets an array of encryption/commitment challenges from which it chooses the one which it wants to attack in the sense of the original notion. (Moreover, the adversary gets an access to the decryption/extraction oracle for each of the challenges.) The modified reductions $\mathcal{B}_C$ and $\mathcal{B}_E$ depend on the probabilities $\alpha_C$ and $\alpha_E$, $\alpha_E + \alpha_C \leq 1$, that adversary $\mathcal{A}$ chooses the attack pattern useful for either reduction (corresponding to our Case1 and Case2 events). The improved reduction algorithms $\mathcal{B}_C$ and $\mathcal{B}_E$ re-run the Enc-MCA-protocol adversary $\mathcal{A}$, and in each attempt they use another key from these "multi-challenge" versions of the encryption/commitment challenge. If the adversary successfully attacks in any of the runs, the reduction translates this attack successfully as well. The *conditional* probability that either one succeeds provided the adversary chooses the appropriate attack pattern, is bounded, respectively, by $2^{-k}+O(\epsilon_{mC})$ and $2^{-k}+O(\epsilon_{mE})$, where $\epsilon_{mC}$ and $\epsilon_{mE}$ are the security bounds on the multi-challenge versions of the commitment/encryption games. However, since the reduction needs $O(1/\alpha)$ of these challenges, by hybrid arguments we get $\epsilon_{mC} \approx \epsilon_C/\alpha_C$ and $\epsilon_{mE} \approx \epsilon_E/\alpha_E$, and hence the overall probability that $\mathcal{A}$ succeeds is bounded by $2^{-k} + O(\epsilon_C, \epsilon_E)$.

# D Proof of Theorem 11 (multi-player, multi-session SAS-MCA)

*Proof.* As usual, the proof goes by contradiction. That is, we prove that if there exists an $(n, R, \bar{R})$-adversary $\mathcal{A}$ which can attack the proposed protocol in time $T \geq min(T_C, T_E) - \mu$, with a probability better than $nR\bar{R}2^{-k} + max(nR\bar{R}\epsilon_C, nR\bar{R}\epsilon_E)$, then there exists *either* a $T$-time adversary $\mathcal{B}_C$ which wins the hiding-on-extraction-attack game OW-ExA with a probability better than $2^{-k}+\epsilon_C$, *or* there exists an adversary $\mathcal{B}_E$ which wins the OW-R-CCA game for the encryption scheme with probability better than $2^{-k} + \epsilon_E$.

$\mathcal{A}$ succeeds if it can find a player $P_i$ and a session $s$ with a party $P_j$, such that $P_i$ accepts a message $\hat{m_j}^{(s)}$ on this session, without launching an instance of $P_j$ on message $\hat{m_j}^{(s)}$ on $s$. $\mathcal{A}$ can achieve this by routing the SAS message $SAS_j^{(s')}$ corresponding to some other session $s'$, on the session $s$. In other words, $\mathcal{A}$ wins if $R_i^{(s)} \oplus \hat{R_j}^{(s)}$ turns out to be the same as $\hat{R_i}^{(s')} \oplus R_j^{(s')}$.

Similar to the proof for the $(2, 1, 1)$-attacker case ( see the proof of Theorem 10), we proceed as follows.

For each of these three message interleaving patterns we'll have to consider two subcases, depending on whether the pair $(\hat{m}_i^{(s')}, \hat{PK}_i)$ that the adversary delivers to $P_j$ in message #5 (see Figure 4.3) is equal to $(m_i^{(s)}, PK_i)$ that player $P_i$ sends in message #1.

Let's denote the event that adversary succeeds in an attack as AdvSc, the event that $(\hat{m}_i^{(s')}, \hat{PK}_i) = (m_i^{(s)}, PK_i)$ *and* that the attack succeeds as SM, the event that $(\hat{m}_i^{(s')}, \hat{PK}_i) \neq (m_i^{(s)}, PK_i)$ *and* that the attack succeeds as NSM, and we'll use Int[1], Int[2], Int[3] to denote events when the adversary follows, respectively, the 1st, 2nd, or 3rd message interleaving pattern. We divide the six possible patterns which the successful attack must follow into the following two cases:

$$\mathsf{Case1} = (\mathsf{NSM} \vee (\mathsf{SM} \wedge \mathsf{Int}[2])) \quad \text{and} \quad \mathsf{Case2} = (\mathsf{SM} \wedge \mathsf{Int}[1]) \vee (\mathsf{SM} \wedge \mathsf{Int}[3]))$$

We show that if adversary succeeds in any of the above cases with probability $p$, then this implies an attack against one of the above security assumptions. We construct two reduction algorithms, $\mathcal{B}_C$ and $\mathcal{B}_E$, which attack respectively the commitment and the encryption scheme used in the Enc-MCA protocol. More specifically, $\mathcal{B}_C$ attacks the OW-ExA property of the commitment scheme, and $\mathcal{B}_E$ attacks the OW-R-CCA property of encryption. Both algorithms $\mathcal{B}_C$ and $\mathcal{B}_E$ will use the Enc-MCA attacker $\mathcal{A}$. The reductions $\mathcal{B}_C$ and $\mathcal{B}_E$

are successful depending on which of the above cases $\mathcal{A}$ takes. Namely, in event Case1 reduction $\mathcal{B}_C$ wins the OW-ExA commitment game, and in event Case2 reduction $\mathcal{B}_E$ wins the OW-R-CCA encryption game.

Note that $\mathcal{B}_C$ and $\mathcal{B}_E$ need to guess the values $(P_i, s, P_j)$ that $\mathcal{A}$ attacks, out of the instances launched by $\mathcal{A}$ (with a probability $1/(nR)$) and the intermediary session $s'$ (with a probability $1/\bar{R}$).

Since, AdvSc = Case1$\vee$Case2, and therefore, if $\Pr[\mathsf{AdvSc}] = p$ then either $\Pr[\mathsf{Case1}] \geq p/(2nR\bar{R})$ or $\Pr[\mathsf{Case2}] \geq p/(2nR\bar{R})$. Note that by assumption on $p$, we have that $p/(2nR\bar{R}) > 2^{-k} + \epsilon_C$ *and* $p/(2nR\bar{R}) > 2^{-k} + \epsilon_E$, and hence *either* $\mathcal{B}_C$ wins the OW-ExA commitment game with probability greater than $2^{-k} + \epsilon_C$ *or* $\mathcal{B}_E$ wins the OW-ExA commitment game with probability greater than $2^{-k} + \epsilon_E$. Moreover, both $\mathcal{B}_C$ and $\mathcal{B}_E$ make only a few cryptographic operations in addition to running $\mathcal{A}$, so their running time will be $T + \mu$ for a small constant $\mu$, and hence, by assumption on $T$, their running times are below both $T_C$ and $T_E$. Since this contradicts either the $(T_C, \epsilon_C)$ OW-ExA-security of the commitment scheme or the $(T_E, \epsilon_E)$ OW-R-CCA-security of the encryption, the theorem will follow.

It remains for us to construct algorithms $\mathcal{B}_C$ and $\mathcal{B}_E$ with the properties claimed above. Algorithm $\mathcal{B}_C$, depending on the behavior of $\mathcal{A}$, executes one of the following sub-algorithms:

> If $(\hat{m}_i, \hat{PK}_i) \neq (m_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern I, II, or III, then $\mathcal{B}_C$ executes sub-algorithms, respectively, $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$, and $\mathcal{B}_C[3]$.
>
> If $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern II, $\mathcal{B}_C$ executes $\mathcal{B}_C[4]$.
>
> Otherwise, i.e. if $\mathcal{A}$ sends $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ and $\mathcal{A}$ follows patterns I or III, $\mathcal{B}_C$ fails.

Similarly, based on the behavior of $\mathcal{A}$, algorithm $\mathcal{B}_E$ proceeds in one of the following ways:

> If $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern I, $\mathcal{B}_E$ executes $\mathcal{B}_E[1]$.
>
> If $(\hat{m}_i, \hat{PK}_i) = (m_i, PK_i)$ and $\mathcal{A}$ chooses interleaving pattern III, $\mathcal{B}_E$ executes $\mathcal{B}_E[2]$.
>
> Otherwise, i.e. if $\mathcal{A}$ sends $(\hat{m}_i, \hat{PK}_i) \neq (m_i, PK_i)$ or $\mathcal{A}$ follows interleaving pattern II, $\mathcal{B}_E$ fails.

**Constructions of $\mathcal{B}_C$[1], $\mathcal{B}_C$[2] and $\mathcal{B}_C$[3].** Whenever $\mathcal{A}$ starts a protocol for any player $P_I$ with role as "$ROLE$" on any session $S$, such that $(I, ROLE, S) \neq (i, init, s')$ or $(I, ROLE, S) \neq (j, resp, s)$, $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$ and $\mathcal{B}_C[3]$ simulate a perfect view to $\mathcal{A}$ by simply following the protocol. The

interaction of $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$ and $\mathcal{B}_C[3]$ with $\mathcal{A}$ and with an instance each of player $P_i$ and $P_j$, when $\mathcal{A}$ launches an instance of $P_i$ as initiator on session $s'$ and an instance of $P_j$ as responder on session $s$, follow in exactly the same manner as depicted in Figures E.1, E.2 and E.3, respectively, of the proof of Theorem 10.

**Construction of $\mathcal{B}_C$[4].** $\mathcal{B}_C[4]$ simply follows the protocol for any player $P_I$ with role as "ROLE" on session $S$ such that $(I, ROLE, S) \neq (i, init, s)$ or $(I, ROLE, S) \neq (j, resp, s')$ $\mathcal{B}_C[4]$'s communication with $\mathcal{A}$ and with an instance each of player $P_i$ and $P_j$, when $\mathcal{A}$ launches an instance of $P_i$ as an initiator on session $s$ and an instance of $P_i$ as a responder on session $s'$ and an instance of $P_j$ on $s'$, follows in exactly the same manner as we showed in Figure E.4 of the proof of Theorem 10.

**Construction of $\mathcal{B}_E$[1] ($\mathcal{B}_E$[2] follows similarly).** $\mathcal{B}_E[1]$ chooses the secret key, private key pair for every player except $P_i$. The public key of $P_i$ is set to be the public key $PK$ that $\mathcal{B}_E[1]$ obtains from the challenger. Whenever $\mathcal{A}$ starts a protocol for any player $P_I$ with role "$ROLE$" on a session $S$, such that $I \neq i$ or ($I = i, ROLE \neq$ "init") or $(I, ROLE, S) \neq (j, resp, s')$, $\mathcal{B}_E[1]$ simply follows the protocol. When $\mathcal{A}$ launches a protocol for $P_i$ as an initiator, but on a session $S$ different than $s$, $\mathcal{C}_E$ follows the protocol with public key of $P_i$ being set to $PK$ and makes use of the decryption oracle to obtain $R_j^{(S)}$ by querying the ciphertext $\mathsf{Enc}_{PK}(S, \hat{m_j}^{(S)}, R_j^{(S)})$ that $\mathcal{A}$ provides, to the decryption oracle. The interaction of $\mathcal{B}_E[1]$ with $\mathcal{A}$ and with the challenger, when $\mathcal{A}$ launches an instance of $P_i$ as an initiator on session $s$ and an instance of $P_j$ as responder on session $s'$ follows in exactly the same manner we showed in Figures 4.4 of the proof of Theorem 10.

If $\mathcal{A}$ wins in time $T$ with probability $p$, then our algorithm $\mathcal{B}$ wins in time $T+\mu$ with probability $p/(2nR\bar{R})$, where $\mu$ is a small constant denoting the additional time complexity taken by $\mathcal{B}$. This contradicts either $(T_C, \epsilon_C)$-security of the commitment scheme or $(T_E, \epsilon_E)$-

security of OW-R-CCA encryption scheme, and therefore,

$$T \geq min(T_C, T_E) - \mu \text{ or } p \leq nR\bar{R}2^{-k+1} + max(2nR\bar{R}\epsilon_C, 2nR\bar{R}\epsilon_E) \tag{D.2}$$

□

# E Reductions $\mathcal{B}_C[1]$, $\mathcal{B}_C[2]$, $\mathcal{B}_C[3]$ and $\mathcal{B}_C[4]$ in the Proof of Theorem 10 (two-player, single-session SAS-MCA)

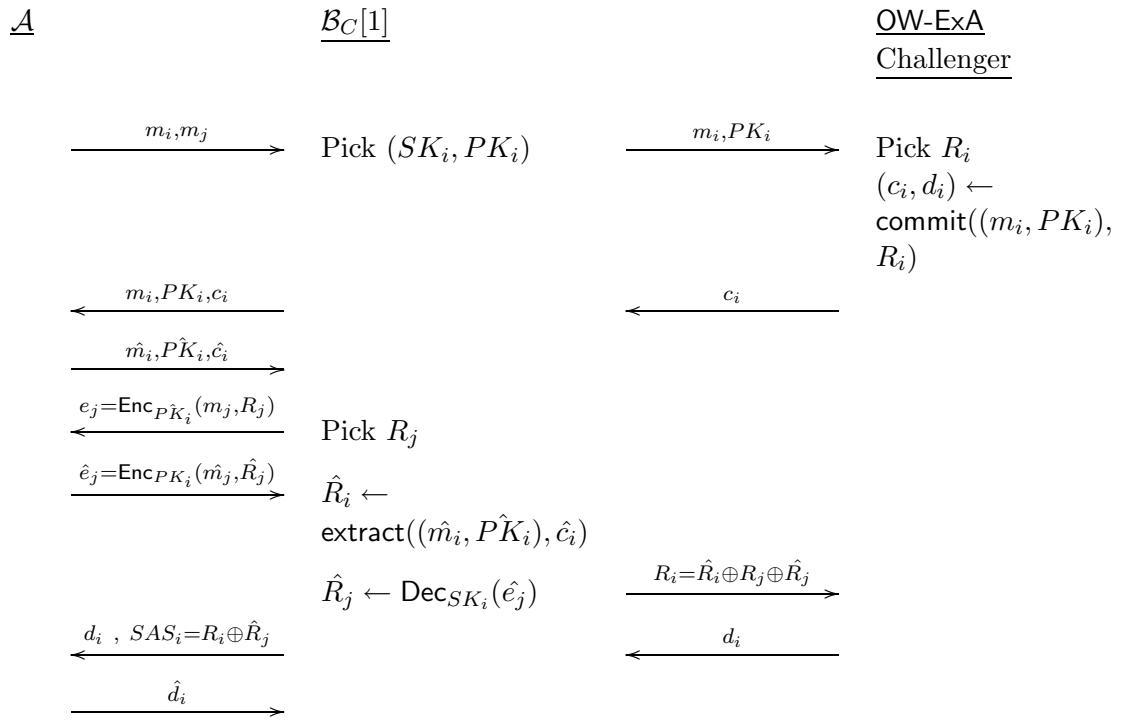



Figure E.1: Construction of $\mathcal{B}_C[1]$ ($(m_i, PK_i) \neq (\hat{m}_i, \hat{PK}_i)$, interleaving case I)

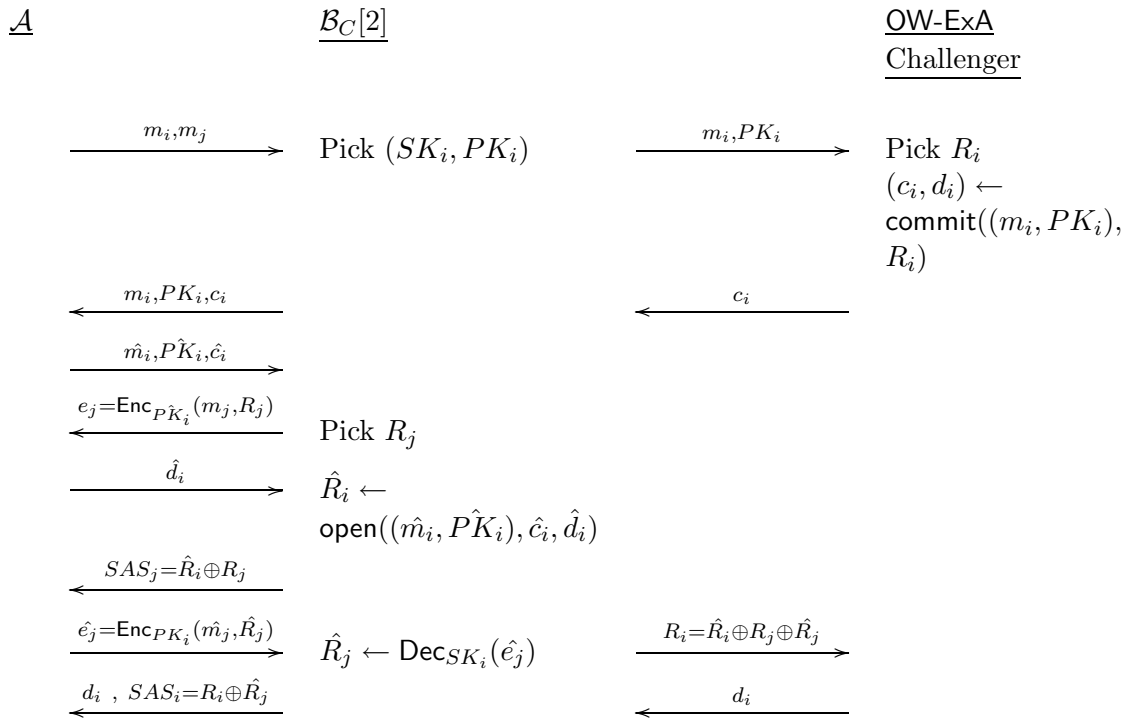



Figure E.2: Construction of $\mathcal{B}_C[2]$ ($(m_i, PK_i) \neq (\hat{m}_i, \hat{PK}_i)$, interleaving case II)

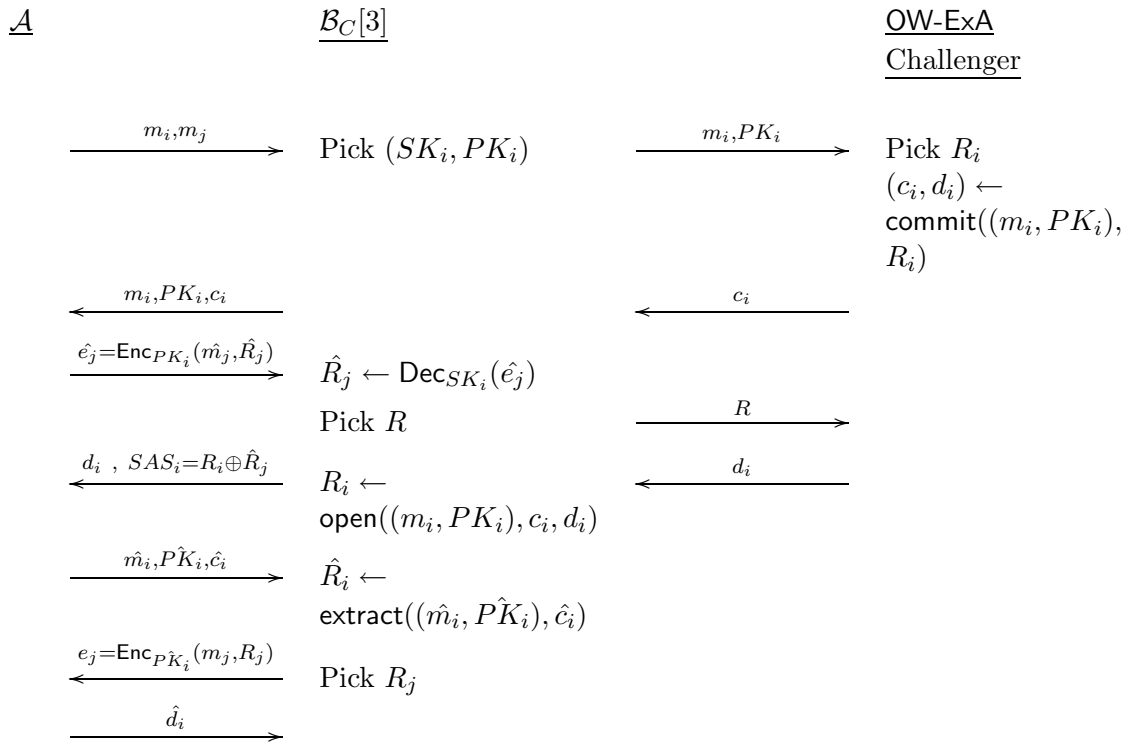



Figure E.3: Construction of $\mathcal{B}_C[3]$ ($(m_i, PK_i) \neq (\hat{m_i}, \hat{PK_i})$, interleaving case III)

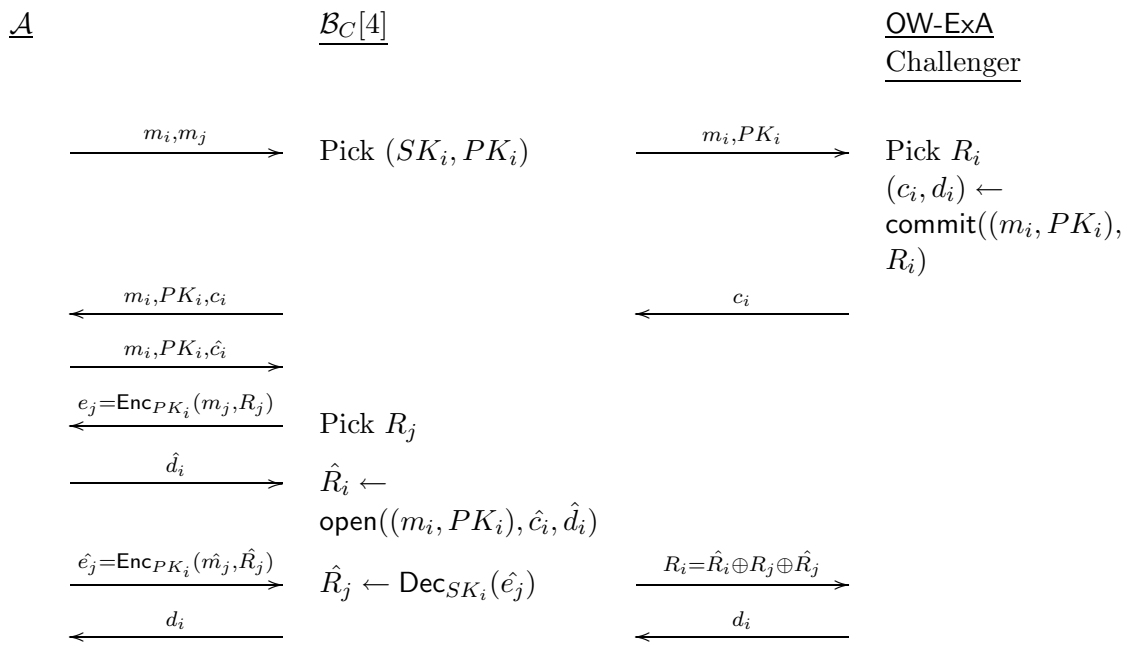



Figure E.4: Construction of $\mathcal{B}_C[4]$ ($(m_i, PK_i) = (\hat{m_i}, \hat{PK_i})$, interleaving case II)