160

Proactive Secret Sharing and Public Key Cryptosystems

by

Stanislaw Jarecki

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degrees of

Master of Science in Computer Science and Engineering

and

Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1995 [February 1996]

Author ....

Department of Electrical Engineering and Computer Science

September, 1995

Certified by ....

Ronald L. Rivest

Professor

Thesis Supervisor

Certified by ....

Hugo Krawczyk

IBM Research

Thesis Supervisor

Certified by ....

Moti Yung

IBM Research

Thesis Supervisor

Accepted by ....

F. R. Morgenthaler

Chairman, Departmental Committee on Graduate Students

Eng.

Proactive Secret Sharing and Public Key Cryptosystems

by

Stanislaw Jarecki

Submitted to the Department of Electrical Engineering and Computer Science
on September, 1995, in partial fulfillment of the
requirements for the degrees of
Master of Science in Computer Science and Engineering

and

Bachelor of Science in Computer Science and Engineering

Abstract

Secret sharing schemes protect secrecy and integrity of information by dividing it into shares and distributing these shares among different locations. In $k+1$ out of n threshold schemes, security is assured if *throughout the entire life-time of the secret* the adversary compromises no more than k of the n locations. For long-lived and sensitive secrets this protection may be insufficient. We propose a new type of secret sharing scheme, called *proactive*, in which the share holders periodically (e.g. once a day) rerandomize the distribution of the secret into shares in such a way that if the adversary learns no more than k shares before the rerandomization, this information is useless for attacking the secret afterwards. In other words, the adversary willing to learn or destroy the secret has to break to at least $k+1$ locations *during the same time period*, i.e. between consecutive executions of the rerandomization protocol.

We extend proactive secret sharing schemes to function sharing, which allows for various proactive public key cryptosystems. As example, we construct with $n = 2k+1$ servers a proactive Certification Authority, such that the adversary who wants to learn or destroy its secret signature key has to break to more than k servers during a single time period. We propose two efficient proactive secret sharing protocols. We define the security notions of proactive secret sharing and we provide the proofs of security of the two protocols we propose. Our solutions assume broadcast channel between servers and the computational hardness of some cryptographic primitives.

Thesis Supervisor: Ronald L. Rivest
Title: Professor

Thesis Supervisor: Hugo Krawczyk
Title: IBM Research

Thesis Supervisor: Moti Yung
Title: IBM Research

Acknowledgments

This thesis has been written while I was on an internship with the Network Security Group of I.B.M. T.J.Watson Research Laboratories in Hawthorne, New York. The main results of this thesis are published in the *Advances of Cryptography - CRYPTO'95* conference, as a paper written by Amir Herzberg, Hugo Krawczyk, Moti Yung and me. This thesis provides proofs of security of the protocols we propose in that paper, and it exposes the issues presented in that paper in greater detail and, hopefully, with greater clarity.

I would like to thank Moti Yung and Hugo Krawczyk, two researchers at IBM who worked with me most closely and who introduced me to the world of cryptography: Without their guidance I would not be able to write this work. I am also indebted to Amir Herzberg, who has been my benevolent manager at IBM. He took me to his network security group when I knew almost nothing about it, and gave me a wonderful project that has led to this thesis. I would like to thank Ron Rivest for his supervision of my work, and for his cryptography seminars at MIT. I am also indebted to him for watching over me as my undergraduate academic advisor at MIT.

I would like to dedicate this work to the memory of my father.

Contents

1 Introduction **5**

1.1 Motivation for Proactivity In Secret Sharing 5

1.2 Proactive Function Sharing 7

1.3 Previous Work on Secret Sharing and Mobile Adversaries 8

1.4 Organization of the Thesis 8

2 Preliminaries **9**

2.1 Model and Assumptions 9

2.2 Cryptographic Tools 11

2.3 Sketch of Security Definitions 13

3 Security Results of our Proactive Secret Sharing Schemes **15**

4 Periodic Share Renewal Protocol **17**

4.1 Initial Setting: Black-box Public Key 17

4.2 Initialization of Secret Sharing 18

4.3 Share Renewal 18

4.4 Share Renewal Protocol in the Presence of Active Attackers 19

4.5 Resolving Accusations 20

5 Share Recovery Protocol **23**

5.1 Detection of Corrupted Shares 23

5.2 Basic Share Recovery Protocol 25

5.3 Full Share Recovery Protocol 26

6 The Combined Protocol **29**

6.1 Reconstruction of the Secret 29
6.2 Dynamically Secure Private Keys 29
6.3 The Combined Protocol 30

7 Proactive Secret Sharing with Pedersen's VSS 31

7.1 Initialization 31
7.2 Share Renewal 32
7.3 Share Recovery 33

8 Applications: Proactive Public Key Cryptosystems 35

8.1 Proactively Secure Certification Authority 36
8.1.1 Background on ElGamal signatures 36
8.1.2 Distributed Signature Operation 37
8.2 Proactively Secure Database 39

9 Security Analysis 41

9.1 Basic Proactive Share Renewal 42
9.2 Protection of Integrity 44
9.3 Private Communication Links vs. Broadcast in VSS 46
9.4 Protection of Secrecy in the version using Pedersen's VSS 49
9.4.1 Share Renewal with Black-Box Encryption 49
9.4.2 Share Renewal with Semantically Secure Encryption 56
9.4.3 Secrecy Protection in the Share Recovery Protocol 71
9.4.4 Secrecy Protection during the Lifetime of the Proactive Algorithm 73
9.5 Protection of Secrecy in the version with Feldman's VSS 74

Chapter 1

Introduction

1.1 Motivation for Proactivity In Secret Sharing

Secret sharing schemes protect the secrecy and integrity of information by distributing it over different locations. For sensitive data these schemes constitute a fundamental protection tool, forcing the adversary to attack multiple locations in order to learn or destroy the information. In particular, in a $(k+1, n)$-threshold scheme, an adversary needs to compromise more than k locations in order to learn the secret, and corrupt at least $n-k$ of its shares in order to render it unreconstructible. However, the adversary has the *entire life-time of the secret* to mount these attacks. Gradual and instantaneous break-ins into a subset of locations over a long period of time may be feasible for the adversary. Therefore for long-lived and sensitive secrets the protection provided by traditional secret sharing may be insufficient.

A natural defense is to periodically refresh the secrets; however, this is not always possible. That is the case of inherently long-lived information, such as cryptographic master keys (e.g., signature/certification keys), data files (e.g., medical records), legal documents (e.g., a will or a contract), proprietary trade-secret information (e.g., Coca-Cola's formula), and more.

To realize how unsatisfactory such refreshment of the secret is, imagine that one wants to protect a legal document by encrypting it under some initial key and then periodically change that key, decrypting the document with the old key and encrypting it with the new one every time the key changes. Such a solution does not protect the integrity of the document at all, and it also exposes the secrecy to the adversary that happens to attack the server at the moment when the key is changing and the document is being decrypted.

Thus, what is actually required to protect secrecy and integrity of long-lived information is to be able to *periodically renew the shares without changing the secret*, in such a way that any information learned by the adversary about individual shares becomes obsolete after the shares are renewed. Similarly, to avoid gradual destruction of the information by corruption of its shares, it is necessary to *periodically recover* *lost or corrupted shares* without compromising their secrecy.

The above mechanisms, periodic renewal of shares and periodic recovery of corrupted shares, constitute the core properties of *proactive secret sharing* as presented here. In the proactive approach, the lifetime of the secret is divided into *periods of time* (e.g., a day, one week, etc.). At the beginning of each time period the share holders engage in an interactive *update protocol*, after which they hold completely *new* shares of the *same* secret. Previous shares become obsolete and should be safely erased. As a consequence, in the case of a $(k+1, n)$ proactive threshold scheme, the adversary trying to learn the secret is required to compromise $k+1$ locations during a *single* time period, as opposed to incrementally compromising $k+1$ locations over the *entire* life-time of the secret. As an example, consider a secret that lives for five years: A weekly refreshment of shares will reduce the time available for the adversary to break the $k+1$ necessary locations from five years to one week. Similarly, the destruction of the secret requires the adversary to corrupt $n-k$ shares in a *single* time period. Note that proactivity is a characteristics of the *storage phase* of secret sharing protocols. The traditional focus in secret sharing research has been on designing protocols to distribute the shares of a secret and then to reconstruct the secret from the shares. Our concern is what happens in between these two protocols: We replace the passive storage of shares in traditional secret sharing schemes with proactive secret sharing.

The proactive algorithm protects information from a *mobile* adversary, who does not control a server forever, but can succeed in breaking-in for limited periods of time and move on to attack other servers, possibly breaking into each secret sharing server multiple times. This paradigm of a mobile yet detectable adversary is motivated by two facts: On one hand secret sharing servers can be easily accessible through the network for an adversary to attack (especially if they have to perform some function with this secret, as in the case of the certification authority and other function sharing applications). Furthermore, nothing in the world can be forever secure: If we design an algorithm that can tolerate adversaries who break into servers and make them behave in arbitrarily bad way, then such an algorithm is in particular secure against hardware failures, electricity breakdowns, operating system crashes and all other disasters that do happen in real life even without malicious adversaries. On the other hand, intrusions in networks (such as software modification, viruses, etc.) and any other malfunctioning of a secret sharing server can be eventually exposed by available detection mechanisms like virus scanners, network monitoring etc. The goal behind a proactive security solution is to neutralize the power given to the adversary by the mobility property: To allow servers to regain the secret information destroyed by the adversary during a break-in and to make the information gained by the adversary during a break-in useless during future break-ins.

In this thesis we propose two proactive secret sharing schemes, which can both support up to $k = n/2 - 1$ corrupted parties at any time period. Both these schemes assume the existence of secure encryption and signature functions, as well as the security of the verifiable secret sharing schemes (VSS) based on homomorphic functions [Fel87, Ped91]. At the system level, we assume a broadcast channel, locally unpredictable sources of perfect randomness and synchrony (as in VSS).

1.2 Proactive Function Sharing

Proactive secret sharing protocol offers a way of maintaining sensitive information that provides a novel degree of protecting its secrecy and integrity. It can be used for secure storage of any long-lived data whose secrecy and integrity are very important. Moreover, proactive secret sharing provides secure storage of data without keeping this data in isolation, and therefore, it is well suited to function sharing applications, which usually require secrecy and integrity but also high availability of the information on which the secret-shared function depends. A good example is a decryption key of a secure database or archive, or a signature key of any network certification authority (key certification, time stamping, ticket granting etc):

- Such a key must be kept with as high as possible secrecy, proportional to the cost of damage that could be done by a dishonest person who learns this key. If in the future, banks will be accessible through computer networks and will verify identify of their customers with public keys signed by a network Certification Authority (CA), then the compromise of secrecy of that key could potentially cost all the assets of the bank.

- The integrity of that key is very valuable, because if it is lost, one has to rewrite all past signatures (certificates, time stamps, tickets etc) and reinstall the new public keys in the network, which is not only costly (it could involve issuing new smart cards to every user), but it could also compromise the security of the whole system.

- On the other hand, this key must be highly available in a network environment. The certification authority has to process its jobs, and therefore it must be connected to other computers with a communication network. In particular, this precludes the possibility of storing the sensitive information in isolation.

We refer to the above scenarios, in which the secret is distributed with a proactive secret sharing, and instead of being simply stored to be eventually reconstructed, it is constantly *used* to perform some function, as a *function sharing.* In other words, function sharing distributed the power of computing a function, instead of distributing a static information. In section 8, we will show how proactive secret sharing can be used as a building block in proactive function sharing.

1.3 Previous Work on Secret Sharing and Mobile Adversaries

Our work focuses on improving the *storage* of shares in secret sharing, which so far has been a neglected part of secret sharing protocols. The reader can find an extensive survey of secret sharing work in [Sim92] and a quick overview of the basic ideas in [Riv90]. Also, among the papers we cite in the further parts of the thesis, the introductions to [Rab88] and [Ped91] present a short history of important results in the secret sharing research. Particularly relevant to our work are the basic secret sharing scheme of Shamir [Sha79] and the verifiable secret sharing schemes of [Fel87] and [Ped91].

The mobile adversary setting, was originally presented in the context of secure function evaluation by Ostrovsky and Yung [OY91]. That solution allowed large (polynomial) redundancy in the system (redundancy is the ratio of total servers n to the threshold k of simultaneously faulty servers), and used the availability of huge majority of honest servers to achieve the very general task of secure computation in the information theoretic sense. The same mobile adversary model was then used in a more practical setting by Canetti and Herzberg [CH94] who proactively maintained a local pseudorandom number generators of n servers.

1.4 Organization of the Thesis

In **chapter 2** we describe our computational model: the requirements we impose on secret sharing servers and the network that connects them, the bounds on the adversary against which our scheme is secure and the computational assumptions we make. We present the cryptographic tools we use in our protocols and we sketch the notions of security with which we characterize the protocols we propose. In **chapter 3** we state the theorems about the strength of secrecy and integrity protection offered by the two versions of a proactive secret sharing protocol we propose in this paper. In **chapters 4, 5 and 6** we present our basic scheme: the proactive secret sharing protocol which uses Feldman's verifiable secret sharing scheme. In **chapter 7** we present the alternative scheme, replacing Feldman's with Pedersen's VSS. In **chapter 8** we present the function-sharing applications, namely, the proactively secure Certification Authority, and the proactively secure database. In **chapter 9** we present the proofs of the claims about the security of our algorithms, which are stated in chapter 3.

Chapter 2

Preliminaries

2.1 Model and Assumptions

We describe the model and the environment where our proactive secret sharing algorithm can be executed.

We assume a system of n servers $\mathcal{A} = \{P_1, P_2, \ldots, P_n\}$ that will (proactively) share a secret value x through a $(k+1, n)$-threshold scheme (i.e., k shares provide no information on the secret, while $k+1$ suffice for the reconstruction of the secret). We assume that the system is securely and properly initialized. The goal of the scheme is to prevent the adversary from learning the secret x, or from destroying it (In particular, any group of $k+1$ non-faulty servers should be able to reconstruct the secret whenever it is necessary).

SERVERS AND COMMUNICATION MODEL: Each server in $\mathcal{A}$ is connected to a common broadcast medium, called communication channel, with the property that every message sent on this channel can be instantly read by all other parties connected to this channel. We assume that the system is synchronized, i.e., the servers can access a common global clock. We also assume that each server in $\mathcal{A}$ has a local source of randomness.

TIME PERIODS AND UPDATE PHASES: Time is divided into *time periods* which are determined by the common global clock (e.g., a day, a week, etc.). At the beginning of each time period the servers engage in an interactive *update protocol* (also called *update phase*). At the end of an update phase the servers hold new shares of the secret x.

THE MOBILE ADVERSARY MODEL: The adversary can corrupt servers at any moment during a time period. If a server is corrupted during an update phase, we consider the server as corrupted during both periods adjacent to that update phase. We assume that the adversary corrupts *no more* than k out of n servers in each time period, where k must be smaller than $n/2$ (this guarantees the existence of $k+1$ honest servers at each time).

The reason behind this way of counting corrupted servers is that it is impossible or at least very hard to analyze what happens if we differentiated between adversary who moves from one server to another during the update phase and the adversary who just stays in both servers throughout. It is also not a realistic concern in our setting, where the update phase is negligibly short when compared to the length of a time period: If the adversary could move so fast that she could jump between servers during an update, if it continued to jump with the same speed during the time period, it could visit all the servers and destroy the secret sharing system. Furthermore, notice that even during a regular time period, we treat the adversary who jumps from one server to another in the same way as if both of them are corrupted throughout this time period.

Corrupting a server means any combination of learning the secret information in the server, modifying its data, changing the intended behavior of the server, disconnecting it, and so on. For the sake of simplicity, we do not differentiate between malicious faults and "normal" server failures (e.g., crashes, power failures etc.).

We also assume that the adversary is always connected to the broadcast channel, which means she can hear *all* the messages and inject her own. She cannot, however, modify messages broadcasted by a server that she does not control, nor can she prevent a non-corrupted server from receiving a broadcasted message. We also preclude the possibility that the adversary will flood this communication channel with messages and thus prevent servers $\mathcal{A}$ from communicating. Although this is an attack that can happen in real life, there seem to be no cryptographic ways of preventing it. Additionally, the adversary always knows the non-secret data and the algorithm that each machine performs.

COMPUTATIONAL BOUNDS ON THE ADVERSARY: We assume the adversary to be computationally bounded in the sense of being adequately modeled by a bounded polynomial time probabilistic Turing machine ($\mathcal{BPP}$). In particular, we assume that such an adversary cannot break the following cryptographic primitives: First, it cannot compute (with non-negligible probability) logarithms in a large prime field. Second, it cannot break semantically secure encryptions. Third, it cannot break existentially unforgeable signatures. The last two assumptions are equivalent to a requirement that there exists a semantically secure encryption scheme and an existentially unforgeable signature scheme (For formal descriptions of these notions and the discussions of cryptographic assumptions they rely on, see [GM84] and [GMR88]).

A NOTE ABOUT THE REMOVAL OF AN ADVERSARY FROM A SERVER: We assume that the adversary intruding the servers $\mathcal{A}$ is "removable", through a reboot procedure, when it is detected. The responsibility for triggering the reboot operation (or other measures to guarantee the normal operation of a server) relies on the system management which gets input from the servers in the network. To reboot a server, the operator has to reinstall the code with the proactive secret sharing algorithm on the server and trigger its execution (in section 6.2 we will add reinstallation of private/public keys of a server). For most applications, standard booting procedure in server's ROM can play a role of a trusted kernel which would guarantee the security of the reboot operation as long as the operators are honest.

In addition to regular detection mechanisms (e.g., anti-virus scanners) available to the system management, our protocols provide explicit mechanisms by which all uncorrupted servers (which always constitute majority) detect and alert about a server that misbehaves in the sense of sending out incorrect messages. We assume for simplicity that the reboot operation is performed immediately when attacks or deviations from the protocol are detected and that it takes a negligible amount of time compared with the duration of a time period.

We remark that the initialization of servers and reboot operations require a minimal level of *trust* in the system management, restricted to installation of correct programs and of public keys used for server-to-server communication. Specifically, no secret information is exposed to the system management. This level of trust regarding integrity of installed information is unavoidable for the initialization of any cryptographic system. It is also worth noticing that if the system management fails to install the communication keys properly, it can threaten the integrity of the secret, but not its secrecy.

ERASURE OF PAST INFORMATION: In our protocols we sometimes specify that the servers *erase* some information. This operation (performed by honest servers) is essential for proactive security. Not doing so would provide an adversary that attacks a server at a given period with information from a previous period, and the later could enable the adversary to break the system. In many computer systems, what seems like a memory update to the programmer, can in fact result only in an update of a cache, while the main memory or a copy swapped on a disk remains unchanged. It is a formal requirement of our proactive solution that the secret sharing servers be able to reliably erase their local data.

2.2 Cryptographic Tools

SHAMIR'S SECRET SHARING: Our secret sharing scheme is based on Shamir's scheme [Sha79]. Let q be a prime number, $x \in Z_q$[1] be the secret to be shared, n the number of participants (or share holding servers), and $k+1$ the reconstructibility threshold. The dealer D of the secret chooses a random polynomial f of degree k over Z_q subject to the condition $f(0) = x$. Each share x_i is computed by D as $f(i)$ and then transmitted *secretly* to participant P_i (The evaluation point i could be any publicly known value v_i which uniquely corresponds to P_i; we assume $v_i = i$ as the default value). The reconstruction of the secret can be done by having $k+1$ participants providing their shares and using polynomial interpolation to compute x.

VERIFIABLE SECRET SHARING – VSS: In Shamir's scheme a misbehaving dealer can deal inconsistent shares to the participants, from which they will not be able to

[1]In fact this can be done over any finite field.

reconstruct a secret. To prevent such malicious behavior of the dealer one needs to implement a procedure or protocol through which a consistent dealing can be verified by the recipients of shares. Such a scheme is called *verifiable secret sharing* (VSS) [CGMA85]. Our work uses these schemes in an essential way. We implement our solution using specific schemes due to Feldman [Fel87] and Pedersen [Ped91]. These schemes are based on hard to invert homomorphic functions and, in particular, on the hardness of computing discrete logarithms over Z_p, for prime p.

In this paper we present two versions of a proactive secret sharing algorithm: one working with Feldman's and the other with Pedersen's VSS mechanism. In sections 4, 5 and 6, we present the version with Feldman's scheme since it is somewhat simpler and allows for better presentation of proactivity. In section 7 we show the whole proactive secret sharing protocol modified to use Pedersen's VSS.

FELDMAN'S VSS: We briefly describe Feldman's scheme. Let p and q be two prime numbers such that $p = mq + 1$, where m is a small integer (possibly 2 or 4). Let g be an element of Z_p of order q, i.e. $g^q = 1 \ (mod\, p)$. The basic idea behind this mechanism is that for each share x_i there is a *public* value $y_i = g^{x_i} \ (mod\, p)$ which by the homomorphic properties of the exponentiation function (i.e., $g^a g^b = g^{ab}$) allows every share-holder to verify that its own share is consistent with the public information.

The dealer chooses the polynomial f over Z_q with coefficients f_0, f_1, ..., f_k and broadcasts the corresponding values $g^{f_0}, g^{f_1}, \ldots, g^{f_k}$. Then it secretly transmits the value $x_i = f(i) \ (mod\, q)$ to P_i. Each server P_i verifies its own share by checking the following equation:

$$g^{x_i} \overset{?}{=} (g^{f_0})(g^{f_1})^i(g^{f_2})^{i^2} \ldots (g^{f_k})^{i^k} \ (mod\, p) \tag{2.1}$$

If this equation holds, P_i broadcasts a message saying that it accepts its share as proper. If all servers find their shares correct then the dealing phase is completed successfully. Indeed, by the homomorphic properties of the exponentiation function the above equation holds for all $i \in \{1 \ldots n\}$ if and only if the shares were dealt correctly.

Notice that besides allowing the verification of correct dealing of shares, the public values g^{x_i} can be used at time of secret reconstruction to verify that the participating shares are correct (see Section 6.1).

PEDERSEN'S VSS: Now we recount Pedersen's VSS scheme. Numbers p, q, g, x are the same as above. Additionally, we assume that there is a publicly known number $h \in Z_p^*$, such that nobody knows $d \in Z_q$, where $g^d = h \ (mod\, p)$. To deal the secret x, the dealer picks two random k-degree polynomials $\delta(\cdot), \gamma(\cdot)$ in Z_q with coefficients $\{\delta_m\}_{m\in\{0\ldots k\}}$ and $\{\gamma_m\}_{m\in\{0\ldots k\}}$ respectively, broadcasts $\{\epsilon_m\}_{m\in\{0\ldots k\}}$ where $\epsilon_m = g^{\delta_m} h^{\gamma_m} \ (mod\, p)$ for all $m \in \{0 \ldots k\}$. Then for every $i \in \{1 \ldots n\}$, it secretly sends to P_i its share $\{u_i, w_i\}$, where $u_i = \delta(i)$ and $w_i = \gamma(i)$. The new local verification equation through which every server P_i can check its share becomes:

$$g^{u_i}h^{w_i} \stackrel{?}{=} (\epsilon_0)(\epsilon_1)^i(\epsilon_2)^{i^2}\dots(\epsilon_k)^{i^k}\ (mod\ p) \tag{2.2}$$

The reason why this equation works is that

$$(g^{\delta_0}h^{\gamma_0})\ (g^{\delta_1}h^{\gamma_1})^i\dots(g^{\delta_k}h^{\gamma_k})^{i^k} = g^{\delta_0+\delta_1 i+\dots+\delta_k i^k}\ h^{\gamma_0+\gamma_1 i+\dots+\gamma_k i^k} = g^{\delta(i)}\ h^{\gamma(i)}\ (mod\ p)$$

The basic intuition behind this scheme is that even the computationally unbound adversary seeing $g^x h^z\ (mod\ p)$ can compute $x + dz\ (mod\ q)$, but this still gives out no information about x. In this way, Pedersen's VSS protocol does not allow the adversary to compute g^x. We describe this mechanism in more detail in section 9.2.

MODIFICATION OF ORIGINAL VSS SCHEMES: In both versions of our protocol, we substitute private sending of shares in the above VSS mechanisms with broadcasting them encrypted under public key of the share recipient. This allows us to implement *accusation* protocols thanks to which the server who were dealt wrong shares can prove it to the others. We discuss the reasons and consequences of this modification in section 9.3.

PUBLIC-KEY ENCRYPTION AND SIGNATURES: As mentioned in the paragraph on computational bounds on the adversary in section 2.1, our solution requires an existentially unforgeable signatures [GMR88], and an encryption scheme which is semantically secure in the sense of non-uniform definition of semantic security [GM84]. We do not specify or assume any particular implementation of these functions. For a pair of *sender* S and *receiver* R, we denote by $ENC_R(data)$ the probabilistic encryption of *data* under R's public key; and by $SIG_S(data)$ the signature of *data* under S's private key.

2.3 Sketch of Security Definitions

We give the definition of robustness in proactive secret sharing and we sketch the notions which we will use to describe the secrecy protections of our protocols.

Definition 2.3.1 *We call a proactive secret sharing scheme* **robust** *if at every time, the honest servers can reconstruct the correct secret x.*

The definitions of the secrecy protection offered by our solutions follow the notion of (non-uniform) *semantic security* introduced in [GM84]. We give here only a sketch of these definitions; The formal versions (definitions 9.4.4 and 9.4.5) are presented in section 9.4.2.

We model the adversary as a non-interactive $\mathcal{BPP}$ Turing machine which is fed with all the information learned by the adversary during the lifetime of the proactive protocol. This information consists of the information specified as public (like g, q, p

etc), the communication between servers, and the secret information of the servers that the adversary corrupted in each time period period. Modeling the adversary with an interactive $\mathcal{BPP}$ machine would capture the fact that the adversary can dynamically adapt its future strategy based on her current (and past) view of the computation. However, even if the security analysis in the case of adaptive adversary was possible, it should be first carried out using the restricted model of mobile yet non-adaptive adversary. The security analysis for the full case of adaptive adversaries, should be one of the future directions for this work.

Definition 2.3.2 (sketch) *Let κ be a function applicable to the space of secrets x. Let $p_0^{(\kappa)}$ be the probability that the adversary correctly computes the value $\kappa(x)$ when fed with the public information $\theta(x) = q$ which defines the range of the secret. Let $p_1^{(\kappa)}$ be the analogous probability but after the adversary is also fed with the information gathered during the lifetime of the protocol. Function $\kappa(x)$ models some knowledge about x, while the difference $p_1^{(\kappa)} - p_0^{(\kappa)}$ quantifies the incremental amount of that knowledge "learned" by the adversary by watching the execution of the protocol and actively intruding into the servers. We call a proactive secret sharing scheme* **semantically secure** *if for any function $\kappa(\cdot)$, the difference $\mathcal{K} = p_0^{(\kappa)} - p_1^{(\kappa)}$ is negligible, which means that $\mathcal{K}$ as a function of q decreases faster than any inverse polynomial of $|q|$ ($|q|$ is the bit length of the space in which the secret is randomly chosen).*

The notion of **semantic security relative to the knowledge of g^x** is defined just like above, except that now the public information $\theta(x)$ includes not only the space Z_q of the secret, but also value g^x $(mod\, p)$, i.e. the image of x under the one-way function of exponentiation in a finite field.

Chapter 3

Security Results of our Proactive Secret Sharing Schemes

We state the security properties of our proactive secret sharing algorithm, relative to the adversary described in paragraph "*The Mobile Adversary Model*" of section 2.1, namely, an adversary that corrupts at most k servers in each time period. The computational bounds on the adversary are mentioned in the theorems explicitly. The proofs of the theorems we state here are in section 9.

We present two different versions of the proactive secret sharing scheme because neither of them is strictly better than the other. In comparison with the proactive secret sharing algorithm using Feldman's VSS, the use of Pedersen's VSS scheme weakens the protection of integrity of secret x: It makes the robustness of a proactive secret sharing system subject to computational limits of the adversary, namely a subject to the assumption that the adversary cannot compute logarithms in a large prime field. However, in a trade-off for integrity, Pedersen's scheme strengthens the protection of secrecy of x: It allows us to achieve a true semantic security, while using Feldman's VSS gives only a semantic security relative to the knowledge of g^x.

Theorem 3.1.0 *The proactive secret sharing protocol with Feldman's VSS (as described in sections 4, 5 and 6) has the following properties:*

Robustness: The correct reconstructibility of the secret x is guaranteed as long as the adversary cannot break the signature scheme $SIG(\cdot)$.

Secrecy: If encryption $ENC(\cdot)$ is semantically secure, then this protocol is a proactive secret sharing scheme semantically secure relatively to the knowledge of $g^x \pmod p$.

Theorem 3.2.0 *The proactive secret sharing protocol with Pedersen's VSS (as described in section 7) has the following properties:*

Robustness: The correct reconstructibility of the secret x is guaranteed as long as the adversary cannot find logarithm $d = log_g h$ (where g and h are two random numbers in Z_p^*) and cannot break the signature scheme $SIG(\cdot)$.

Secrecy: If encryption $ENC(\cdot)$ is semantically secure, then this protocol is a semantically secure proactive secret sharing scheme.

A NOTE ABOUT EXPOSING g^x IN FELDMAN'S VSS: Feldman's VSS scheme makes the value $y = g^x \ (mod\ p)$ public, where $x = f(0)$ is the secret being shared. This is the reason why the semantic security of the secrecy protection of the proactive secret sharing protocol based using Feldman's VSS can be only stated relative to the knowledge of $g^x \ (mod\ p)$ (as in theorem 3.1.0). Assuming the hardness of the discrete logarithm operation, the entire value of x cannot be derived from y. However, there is partial information on x that can be efficiently derived, and that, consequently, is not protected by the scheme. Such unprotected information includes the value of $g^x \ (mod\ p)$ itself, the least significant bit of x, etc.

However, to stress the usefulness of Feldman's VSS scheme (and the version of our protocol that uses it) we outline here a methodology to apply it to a secret without leaking the partial information: The real secret to be protected, say s, is first encoded into a longer string x (an "envelope" for s) with the property that given x it is easy to recover s, but given $g^x \ (mod\ p)$ it is hard to derive any information on s (i.e., s represents the *hard core* information of x). In this way, the version of our protocol using Feldman's VSS would give semantically secure secrecy of s, under the assumption that the exponentiation function is hard to invert on random input.

For example, it is known ([LW88], see also [BM84]) that computing $log(|x|)$ upper bits of x from $g^x \ (mod\ p)$ is hard. Therefore, for a secret s of length $\log |q|$, one could construct the envelope x (of size $|q|$) as a concatenation of s (as upper most bits) and a random string r. A more practical scheme can be designed following the techniques of [BR94], which allow for construction of practical hard core envelopes for *any* one-way permutation (applicable, in particular, to the exponentiation function).

Throughout the paper we refer to x as the secret. Applications in which the exposure of the secret's exponent is unacceptable should use the above envelope method, or use the proactive secret sharing solution which uses Pedersen's VSS instead of Feldman's. However, if we apply this secret sharing scheme to implement proactive function sharing of public key cryptosystems, as in section 8, then, since in these schemes exponent g^x plays a role of a public key, it does not matter that the underlying proactive secret sharing exposes it to the adversary.

Chapter 4

Periodic Share Renewal Protocol

Here we present the fundamental component of our solution, namely, the protocol for periodic renewal of shares which preserves the secret and at the same time makes past knowledge obsolete for the adversary.

Beyond guaranteeing the secrecy of the shared secret, our scheme is robust in the sense of guaranteeing integrity and availability of the secret in the presence of up to k misbehaving servers.

4.1 Initial Setting: Black-box Public Key

Cryptographic solutions in a distributed environment typically require the ability to maintain private and authenticated communication between the servers. This is achieved by the servers having pairs of private and public keys corresponding to public-key cryptosystems with encryption and signature capabilities (e.g., RSA, El-Gamal, etc). However, an adversary that breaks into a server and learns its private key can then imp ersonate that server for the whole life of that private k
ing a break-in, the adversary could also modify the private or public keys stored on that server, thus disabling it from communicating with others. Also, if the adversary breaks into P_i and replaces P_j's public key (in P_i's storage of other servers' public keys) with her own, she can later spook P_j to P_i. Therefore, to ensure proactive security, it is necessary to maintain the system of private and public communication keys proactively, namely, to renew them periodically.

We will show in section 6.2 how this can be done in our context (a more general treatment of proactive authentication can be found in [CH95]). However, for clarity of presentation, we start by making the strong assumption that servers are equipped with a pair of private and public keys with a property that the private key cannot be learned or modified by the adversary, even if this adversary manages to break into the server (Similarly, we have to require that during a break-in, the attacker cannot modify the server's view of other servers' public keys). While such an intruder will be able to generate legal signatures and decrypt messages using the private key (as a "black-box"), it will not be able to learn the private key or modify any of the keys. We will remove this assumption and deal with the proactive maintenance of the private/public communication key pairs in section 6.2.

The security of this public key system is essential for our protocols, because all our communication is implemented as an authenticated broadcast, i.e. every message m sent by P_i will have a signature $SIG_i(m)$ attached to it. Whenever server P_i will need to send m "privately" to P_j, it will broadcast $m' = (i, j, ENC_j(m))$ (accompanied, by a signature $SIG_i(m')$).

4.2 Initialization of Secret Sharing

We assume an initial stage where a secret $x \in Z_q$ (for prime q) is encoded into n pieces $x_1, \ldots, x_n \in Z_q$ using a k-threshold Shamir's secret sharing: Each $P_i, i \in \{1 \ldots n\}$ holds its share x_i, where $x_i = f(i)$ for some k-degree polynomial $f(\cdot)$ over Z_q s.t. $x = f(0)$. We assume that this initialization has been carried out securely, i.e. we assume that the initial shares have the security and recoverability properties of traditional secret sharing. Both versions of our proactive secret sharing can be securely initialized without the trusted center: for the description of these solutions we refer the reader to [Fel87] and [Ped91]).

After the initialization, at the beginning of every time period, all honest servers trigger an *update phase* in which the servers perform a *share renewal protocol*. The shares computed in period t are denoted by using the superscript (t), i.e., $\{x_i^{(t)}\}_{t=0,1,\ldots}$. The polynomial corresponding to these shares is denoted $f^{(t)}(\cdot)$.

NOTATION: We say that shares $S = \{x_{i_m}\}_{m \in \{1,\ldots,k+1\}} \subset \{x_i\}_{i \in \mathcal{A}}$ *reconstruct* (or *interpolate* or *correspond*) to secret x if $\sum_{m \in \{1,\ldots,k+1\}} a_{i_m} x_{i_m} = x$, where $\{a_{i_m}\}_{m \in \{1,\ldots,k+1\}}$ are the Lagrange interpolation coefficients for set S.

For any t and any $S' \subset \mathcal{A}$, where $|S'| \geq k+1$, we will call shares $\{x_i^{(t)}\}_{i \in S'}$ *correct* if for every $k+1$ element subset of shares $S = \{x_{i_m}^{(t)}\}_{m \in \{1,\ldots,k+1\}} \subset \{x_i^{(t)}\}_{i \in S'}$, shares in S interpolate to secret x.

4.3 Share Renewal

To renew the shares at period $t = 1, 2, \ldots$, we adapt a simplified version of the update protocol presented by Ostrovsky and Yung in [OY91]. When the secret x is (distributively) stored as a value $f^{(t-1)}(0) = x$ of a k degree polynomial $f^{(t-1)}(\cdot)$ in Z_q, we can update this polynomial by adding it to a k degree random polynomial $\delta(\cdot)$, where $\delta(0) = 0$, so that $f^{(t)}(0) = f^{(t-1)}(0) + \delta(0) = x + 0 = x$. We can renew the shares $x_i^{(t)} = f^{(t)}(i)$ thanks to the linearity of the polynomial evaluation operation:

$$f^{(t)}(\cdot) \leftarrow f^{(t-1)}(\cdot) + \delta(\cdot)\ (mod\ q) \quad \Longleftrightarrow \quad \forall_i\ f^{(t)}(i) = f^{(t-1)}(i) + \delta(i)\ (mod\ q)$$

In our system we will have $\delta(\cdot) = \sum_{i\in\{1\dots n\}} \delta_i(\cdot)\ (mod\ q)$, where each polynomial $\delta_i(\cdot)$ for $i \in \{1 \dots n\}$ is of degree k and is picked independently and at random by the ith server subject to the condition $\delta_i(0) = 0$. The share renewal protocol for each server P_i, $i \in \{1 \dots n\}$, at the beginning of the time period t is as follows:

1. P_i picks k random numbers $\{\delta_{im}\}_{m\in\{1\dots k\}}$ from Z_q. These numbers define a polynomial $\delta_i(z) = \delta_{i1}z^1 + \delta_{i2}z^2 + \dots + \delta_{ik}z^k$ in Z_q, whose free coefficient is zero and hence, $\delta_i(0) = 0$.
2. For all other servers P_j, P_i *secretly* sends $u_{ij} = \delta_i(j)\ (mod\ q)$ to P_j.
3. After decrypting u_{ji}, $\forall j \in \{1 \dots n\}$, P_i computes its new share $x_i^{(t)} \leftarrow x_i^{(t-1)} + (\,u_{1i} + u_{2i} + \dots + u_{ni}\,)\ (mod\ q)$ and *erases* all the variables it used except of its current secret key $x_i^{(t)}$.

This protocol solves the share renewal problem against a ("passive") adversary that may learn the secret information available to corrupted servers but where all servers follow the predetermined protocol. This is formulated in the theorem below. (A solution against "active cheaters", i.e. in the presence of *byzantine* faults, is presented in section 4.4). Notice that we assume in step 2 that the shares are transmitted to the corresponding holders with perfect secrecy. Equivalently, we could specify that every share u_{ji} is broadcasted to P_i on C in encrypted form (i.e. as $ENC_i(u_{ji})$), where encryption operation $ENC_i(\cdot)$ is a black-box encryption, giving perfect secrecy to those that don't know the secret key of P_i. This allows us to prove the information-theoretic secrecy of this scheme. In the next sections we use public key encryption for the transmission of these shares and then the secrecy of the scheme becomes a subject of the strength of the public key encryption scheme.

Theorem 4.3.1 *If all servers follow the above share renewal protocol then:*

Robustness: The new shares interpolate to the secret x.

Secrecy: An adversary who in any time period eavesdrops on no more than k servers learns nothing about the secret.

4.4 Share Renewal Protocol in the Presence of Active Attackers

In the above basic share renewal protocol an *active* adversary controlling a server can cause the destruction of the secret by dealing inconsistent share updates or just by choosing a polynomial δ_i with $\delta_i(0) \neq 0$. In order to assure the detection of wrongly dealt shares we add to the above basic protocol a *verifiability* feature. Namely, we adapt to our scenario Feldman's verifiable secret sharing scheme as described in section 2.2. In traditional applications of verifiable secret sharing, the fact that all the share-holders find their shares to be consistent is used as a proof for a correct dealing of the secret. In our case, this is used as a proof for correct dealing of update shares by the servers.

The verifiable share renewal protocol for each server P_i at period t is as follows:

1. P_i picks k random numbers $\{\delta_{im}\}_{m\in\{1\ldots k\}}$ from Z_q to define the polynomial $\delta_i(z) = \delta_{i1}z^1 + \delta_{i2}z^2 + \ldots + \delta_{ik}z^k$. It also computes values $\epsilon_{im} = g^{\delta_{im}}\ (mod\ p)$, $m \in \{1\ldots k\}$.

2. P_i computes $u_{ij} = \delta_i(j)\ (mod\ q)$, $j \in \{1\ldots n\}$, and $e_{ij} = ENC_j(u_{ij})$, $\forall j \neq i$.

3. P_i broadcast the message $VSS_i^{(t)} = (i, t, \{\epsilon_{im}\}_{m\in\{1\ldots k\}}, \{e_{ij}\}_{j\in\{1\ldots n\}\setminus\{i\}})$, and the signature $SIG_i(VSS_i^{(t)})$.

4. For all messages broadcasted in the previous step by other servers, P_i decrypts the shares intended for P_i (i.e., computes u_{ji} out of e_{ji}, $\forall j \neq i$), and verifies the correctness of shares using the equivalent of the verifiability equation 2.1 from section 2.2, namely, for all $j \neq i$ it verifies:

$$g^{u_{ji}} \overset{?}{=} (\epsilon_{j1})^{i}(\epsilon_{j2})^{i^2}\ldots(\epsilon_{jk})^{i^k}\ (mod\ p). \tag{4.1}$$

(Notice that this equation accounts for the condition $\delta_j(0) = 0$.)

5. If P_i finds all the messages sent in the previous step by other servers to be correct (e.g., all have correct signatures, time period numbers, etc.), and all the above equations to hold, then it broadcasts a signed acceptance message announcing that all checks were successful.

6. If all servers sent acceptance messages then P_i proceeds to update its own share by performing: $x_i^{(t)} \leftarrow x_i^{(t-1)} + (\,u_{1i} + u_{2i} + \ldots + u_{ni}\,)\ (mod\ q)$ and *erases* all the variables it used except for its current share $x_i^{(t)}$.

7. If in the above step 5, P_i finds any irregularities in the behavior of other servers during step 4 then it broadcasts a signed *accusation* against the misbehaving server(s). When to send accusations, and how to resolve them is discussed in the next subsection.

4.5 Resolving Accusations

In step 5 of the above protocol, each server checks the correct behavior and dealing of other servers. If a misbehaving server is found then there are two kinds of actions to take. One is *not* to use the polynomial $\delta_i(\cdot)$ dealt by this server in the renewal of shares in step 6. The second is to alert the system management so that it could take measures to rectify the misbehaving server (e.g., it may be required to reboot the server in order to "expel" the adversary). However, an accusation against a server by another server requires verification, since a misbehaving server could falsely accuse others. For a consistent update of shares, the honest servers need to agree on who the "bad" servers are. We explain below how each server P_i decides on its list $\mathcal{B}_i$ of bad processors.

We say that a message from server P_i at period t is *correct* if it complies with the specifications of the above protocol, including all the specified fields and information (e.g., the correct time period number) as well as a correct signature.

We distinguish between three classes of irregularities in the protocol:

1. Formally incorrect messages: wrong time periods, numbers out of bounds etc.

2. Two or more correct yet different messages from the same server (i.e. containing a valid signature), or no message at all from some server

3. A mismatch in equation 4.1.

Notice that irregularities of the first two types are discovered using public information only, and therefore, all (honest) servers can always detect them and mark the corresponding servers as "bad". The faults of the third kind cause a problem, since they are discovered only locally by a server that receives a share causing a mismatch in equation 4.1.

When server P_i finds that equation 4.1 corresponding to the information sent by P_j does not hold, it has to broadcast an accusation against P_j. The servers must then decide whether it is P_i or P_j who is cheating. A way to do this is by having P_j publicly "defend" itself: If P_j sent a correct u_{ji}, namely, one that passes equation 4.1, then it can expose this value and prove that it corresponds to the publicly available encryption value e_{ji} which was broadcasted by P_j in step 3. To prove this P_j may need to reveal additional information used to compute the encryption (like the random vector used in probabilistic encryption). However, P_j does not need to reveal any private information of itself. Then everybody can check whether the u_{ji} and the additional information published by P_j encrypts under P_i's public key to e_{ji} as broadcasted by P_j in step 3. Second, everybody can check whether this u_{ji} matches equation 4.1. If P_j defends itself correctly then all servers mark P_i as bad, otherwise P_j is marked as bad. Notice that in some encryption schemes (like RSA), the information published by the accuser is sufficient for public verification, which simplifies the above general protocol.

Once all accusations are resolved, every honest server P_i holds the same list of bad servers $\mathcal{B}_i$ (i.e., for each pair (P_i, P_j) of non-faulty servers, $\mathcal{B}_i = \mathcal{B}_j$). Now the computation of the new shares is done by replacing step 6 of the share renewal protocol by:

$$x_i^{(t)} \leftarrow x_i^{(t-1)} + \sum_{j \notin \mathcal{B}_i} u_{ji} \ (mod\ q)$$

Chapter 5

Share Recovery Protocol

In a proactive secret sharing system, participating servers must be able to make sure whether shares of other participating servers have not been corrupted (or lost), and restore the correct share if necessary. Otherwise, an adversary could cause the loss of the secret by gradually destroying $n - k$ shares. In this section we present the necessary mechanisms for detection and recovery of corrupted shares.

A server can have an incorrect share at the beginning of some time period, because it was controlled by the adversary during the previous share renewal protocol (and hence it was prevented to update its share correctly), or because the adversary attacked the server after the update phase and modified the server's secret share. A secret share can also be lost because a server was rebooted or replaced by a new server.

Without share recovery, the proactive scheme would not be secure even against adversaries who can change the local state of the servers they attacked, or in general, in any way disable the attacked server from performing the right protocol (by, for example, disconnecting it from the network). In particular, without this mechanism the scheme is insecure in the presence of hardware failures, crashes of the operating system etc. Also, in a practical system, every server whose misbehavior is detected by others will be rebooted, and its share will be lost because of the reboot procedure.

5.1 Detection of Corrupted Shares

How are corrupted shares detected? In some cases it is easy to detect that a server requires to recover its correct share. This is the case of servers that do not participate in an update phase (e.g., due to a crash), or servers that misbehave during that phase. However, if the share of some server is ("silently") modified by the adversary (e.g., after an update phase) then this modification may go undetected. Hence, in the spirit of proactiveness, the system must periodically test the correctness of the local states of the participating servers, detecting in this way lost or modified shares.

For clarification we note that in section 4.2 we define the notion of *correct* shares. The maximal set of correct shares defines the current correct secret sharing polynomial $f^{(t)}(\cdot)$. The correct secret sharing polynomial is also inductively defined in the proactive update algorithm: If $f^{(t)}(\cdot)$ is a correct secret sharing polynomial in time period $t-1$ then $f^{(t)}(\cdot) = f^{(t-1)}(\cdot) + \sum_{i \notin \mathcal{B}^{(t)}} \delta_i^{(t)}(\cdot)$ is a correct secret sharing polynomial in time period t, where $\mathcal{B}^{(t)}$ is the set of servers that during the update phase at the beginning of time period t correctly created and distributed their partial update polynomials $\delta_i^{(t)}(\cdot)$.

To implement the distributed verifiability of shares, we add an invariant that in each time period t, each server P_i stores a set $\{y_j^{(t)}\}_{j \in \{1\ldots n\}}$ of exponents $y_j^{(t)} = g^{x_j^{(t)}}$ $(mod\ p)$ of current shares of all servers in $\mathcal{A}$. This invariant will also provide robustness in the secret reconstruction protocol (see section 6.1).

Now it is clear why we use Feldman's VSS scheme (and Pedersen's too) as our building block. Not only because it is non-interactive and simple (our protocols would be too costly with interactive VSS), but also because it provides a natural way to change the public exponents $y_i = g^{x_i}$ consistently with changes to the secret values x_i. It is the homomorphism of one-way function of exponentiation in a prime field that allows for verifiability of a local computation on secret shares with a public computation on the images of these shares. This mechanism gives us the share recovery described here and the robustness of the secret reconstruction described in section 5.

The above invariant is achieved as follows:

- First, we augment section 4.2 with the requisite that each server stores the values $y_j^{(0)}$ corresponding to the initial shares $x_j^{(0)}, j \in \{1\ldots n\}$ (this can be achieved by performing Feldman's VSS at initialization).

- Second, using the homomorphism of the exponentiation function, we supplement step 6 of the update protocol in section 4.4 so that each server P_i updates its set $\{y_j\}_{j \in \{1\ldots n\}}$ by computing for every j:

  $$y_j^{(t)} \leftarrow y_j^{(t-1)} * (g^{u_{1j}} * g^{u_{2j}} * \cdots * g^{u_{nj}})\ (mod\ p)$$

  In the general case, the above product is computed using only update shares corresponding to servers that did not misbehave in the update phase, i.e:

  $$y_j^{(t)} \leftarrow y_j^{(t-1)} * \prod_{a \notin \mathcal{B}_i} g^{u_{aj}}\ (mod\ p)$$

  Also, notice that although the servers in protocol 4.4 do not know update shares u_{aj} of other servers, they can compute their exponents by equation 4.1 from the publicly broadcasted $\{\epsilon_{am}\}_{m \in \{1\ldots k\}}$

  $$g^{u_{aj}} \leftarrow \prod_{m \in \{1\ldots k\}} (\epsilon_{am})^{j^m}\ (mod\ p)$$

LOST SHARE DETECTION PROTOCOL: We extend the *update phase* between time periods to include a *share recovery* protocol executed before the *share renewal* protocol. Its first part is the *lost share detection protocol* which works as follows: Every server checks whether its share $x_i^{(t)}$ corresponds to the $y_i^{(t)}$ it stores, i.e. whether $g^{x_i} \overset{?}{=} y_i \ (mod\ p)$. If not, it broadcasts a singed request saying that it needs a share recovery. Otherwise, if its x_i and y_i are consistent, P_i broadcasts the values $\{y_j^{(t)}\}_{j\in\{1\ldots n\}}$ it stores, together with a proper signature. After collecting these messages from all servers and checking their signatures, each server decides by majority on the current proper set $\{y_j^{(t)}\}_{j\in\{1\ldots n\}}$ (correcting its own set if necessary). Now each server P_i can decide on a set $\mathcal{B}_i$ of servers which presented an incorrect (i.e., different from majority) exponent of their own share. These are the servers that P_i believes to need a share recovery, in addition to servers that broadcasted an explicit request to have their shares recovered (In particular, it can be the case that for some i, $P_i \in \mathcal{B}_i$, which means that server P_i decided that its own share is not correct). It is clear that every pair of non-faulty servers (P_i, P_j) has the same view about who has an incorrect share, i.e., $\mathcal{B}_i = \mathcal{B}_j = \mathcal{B}$. From our assumptions about the adversary, there are no more than k servers holding a wrong share at the end of each time period, i.e. $|\mathcal{B}| \leq k$.

5.2 Basic Share Recovery Protocol

The share recovery algorithm is based on the fact that in Shamir's $(k+1,n)$-threshold scheme, any group $\mathcal{D} \subset \mathcal{A}$ of d shares $(k+1 \leq d \leq n-1)$ can be thought of as a $(k+1,d)$-threshold secret sharing of any of the remaining shares $x_r, r \notin \mathcal{D}$.

A straightforward way to reconstruct the shares $x_r = f^{(t)}(r)$ for $r \in \mathcal{B}$, is to let each server in $\mathcal{D} = \mathcal{A}\backslash\mathcal{B}$ send its own share to P_r, which would allow P_r to recover the whole polynomial $f^{(t)}(\cdot)$ and $f^{(t)}(r)$ in particular. However, this would also expose the secret x to P_r. Instead, for each $r \in \mathcal{B}$, the servers in $\mathcal{D}$ will collectively generate a random secret sharing of x_r in a way analogous to that used to re-randomize the secret sharing of the main secret x in the share renewal protocol: Every server P_i in $\mathcal{D}$ deals a random k-degree polynomial $\delta_i(\cdot)$, such that $\delta_i(r) = 0 \ (mod\ q)$. By adding $\delta_i(\cdot)$'s to $f^{(t)}(\cdot)$, a new, random secret sharing $\{x'_i\}_{i\in\mathcal{D}}$ of x_r is obtained. The servers can now send these new shares to P_r, to allow it to compute x_r without letting P_r learn anything about the original shares $\{x_i\}_{i\in\mathcal{D}}$. Also, any coalition of k or less servers, not including P_r, will learn nothing about the value of x_r.

We first present the share recovery protocol stripped of verifiability. It is secure only against an adversary that eavesdrops into k or less servers, but can not change the behavior of the servers. For each P_r that requires share recovery, the following protocol is performed:

1. Each $P_i, i \in \mathcal{D}$, picks a random k-degree polynomial $\delta_i(\cdot)$ over Z_q such that $\delta_i(r) = 0$, i.e. it picks random coefficients $\{\delta_{ij}\}_{j\in\{1\ldots k\}} \subset Z_q$ and then computes $\delta_{i0} = -\sum_{j\in\{1\ldots k\}} \delta_{ij} r^j \ (mod\ q)$.

2. Each $P_i, i \in \mathcal{D}$, broadcasts $\{ENC_j(\delta_i(j))\}_{j \in \mathcal{D}}$.

3. Each $P_i, i \in \mathcal{D}$, creates its new share of x_r, $x'_i = x_i + \sum_{j \in \mathcal{D}} \delta_j(i)$ and sends it to P_r by broadcasting $ENC_r(x'_i)$.

4. P_r decrypts these shares and interpolates them to recover x_r.

5.3 Full Share Recovery Protocol

In the general case, the adversary not only can eavesdrop into the servers but also cause the corrupted servers to deviate from their intended protocol. To cope with these cases, we add to the above protocol (section 5.2) the necessary "verifiability" properties for the dealing of polynomials $\delta_i(\cdot)$ in step 2 and for reconstruction of x_r from x_i's in step 3 and 4:

1. Each $P_i, i \in \mathcal{D}$ picks a random k-degree polynomial $\delta_i(\cdot)$ in Z_q such that $\delta_i(r) = 0$. It does it by randomly picking $\{\delta_{ij}\}_{j \in \{1 \ldots k\}} \subset Z_q$ and computing $\delta_{i0} = -\sum_{j \in \{1 \ldots k\}} \delta_{ij} r^j \ (mod\ q)$.

2. Each P_i verifiably secret-shares its polynomial $\delta_i(\cdot)$ among the set $\mathcal{D}$ using the same mechanism as in the share renewal protocol, i.e., by broadcasting

$$VSS_i = (i, \{g^{\delta_{im}} \ (mod\ p)\}_{m \in \{0 \ldots k\}}, \{ENC_j(\delta_i(j))\}_{j \in \mathcal{D}}) \ , \ SIG_i(VSS_i)$$

3. For all servers P_i, P_j in $\mathcal{D}$, P_j checks P_i by locally verifying whether $\delta_i(r) = 0 \ (mod\ q)$:

$$\prod_{m \in \{0 \ldots k\}} (g^{\delta_{im}})^{r^m} \stackrel{?}{=} 1 \ (mod\ p) \tag{5.1}$$

and whether $\delta_i(j)$ is consistent with exponents of the coefficients of $\delta_i(\cdot)$:

$$g^{\delta_i(j)} \stackrel{?}{=} \prod_{m \in \{0 \ldots k\}} (g^{\delta_{im}})^{j^m} \ (mod\ p) \tag{5.2}$$

4. Depending on the above verification the servers broadcast acknowledgments (if both equations agree) or start accusation protocols (if equation 5.2 does not hold). By public resolution of the accusations, each server in $\mathcal{D}$ and the server P_r decide on set $\mathcal{D}' \subset \mathcal{D}$ of servers that properly constructed and distributed their re-randomization polynomials $\delta_i(\cdot)$. As in the share-renewal protocol, all honest servers (including the recovering server P_r) will arrive at the same set $\mathcal{D}'$.

5. Each server $P_i, i \in \mathcal{D}'$ creates its new share of x_r, $x'_i = x_i + \sum_{j \in \mathcal{D}'} \delta_j(i)$ and sends it encrypted and signed to P_r by broadcasting $REC_i = (i, ENC_r(x'_i))$ and $SIG_i(REC_i)$ on the communication channel.

6. P_r decrypts all the x_i's and takes the exponents $\{g^{\delta_{jm}} \ (mod\ p)\}_{j \in \mathcal{D}', m \in \{0\ldots k\}}$ that were broadcasted in step (2). Then, for all $i \in \mathcal{D}'$, P_r takes the current valid exponent y_i of P_i's share (P_r knows it from the lost share detection protocol) and verifies whether:
$$g^{x_i'} \overset{?}{=} y_i * \prod_{j \in \mathcal{D}'} g^{\delta_j(i)} = g^{x_i + \sum_{j \in \mathcal{D}'} \delta_j(i)} \ (mod\ p) \tag{5.3}$$
where for all $j \in \mathcal{D}'$:
$$g^{\delta_j(i)} \leftarrow \prod_{m \in \{0\ldots k\}} (g^{\delta_{jm}})^{(i^m)} \ (mod\ p)$$
7. In this way P_r arrives at a set $\mathcal{D}'' \subseteq \mathcal{D}'$ of servers that during step (5) broadcasted correct new shares x_i'. Now P_r can interpolate these shares to recover x_r, because from our assumptions on the adversary, $|\mathcal{D}''| \geq k+1$

MULTI-SECRET SHARING: Theoretically, instead of recovering each lost share separately by treating the set $\{x_i\}_{i \in \mathcal{D}}$ as a secret sharing of a single x_r for each $r \in \mathcal{B}$, we can treat it as a multi-secret sharing (introduced in [FY92]) of all $\{x_r\}_{r \in \mathcal{B}}$. The servers $\mathcal{D}$ can recover shares $\{x_r\}_{r \in \mathcal{B}}$ simultaneously, by adding random k-degree polynomials $\delta_i(\cdot)$ such that $\delta_i(r) = 0$ for all $r \in \mathcal{B}$ to their shares and then sending the new shares to servers $\mathcal{B}$ to let them reconstruct their original shares. Even though this "simultaneous" solution allows servers $\mathcal{B}$ to learn each other's shares $\{x_r\}_{r \in \mathcal{B}}$, this scheme will be secure against the adversary we specified in section 2.1: Since we do not distinguish between an adversary that destroys the share and the adversary that learns it, if the servers $\mathcal{B}$ need a share recovery, we can assume their shares are known to the adversary already.

However, in practice, such a solution obviously weakens the security of the system: Even though we do not specify it formally, our proposed solution is secure if in every time period an adversary manages to destroy the shares of k servers without learning them (e.g. by crashing the servers) *and* simultaneously manages to learn k other shares (e.g. by injecting memory-scanning viruses into the servers that store them).

Chapter 6

The Combined Protocol

In this section we present the remaining parts of our protocol. Then we show how the protocols described here and in sections 4 and 5 combine to the proactive secret sharing protocol using Feldman's VSS.

6.1 Reconstruction of the Secret

In sections 4 and 5 we have shown how to renew and recover shares so they stay consistent with the secret, and in particular, so that at any time there are at least $k+1$ honest parties that could reconstruct the secret if desired. However, the secret reconstruction protocol itself is possible only if the participants are able to detect servers that provide incorrect shares to the reconstruction. This detection is easily accomplished as follows: To reconstruct the secret in time period t, every server broadcasts (signed) its share $x_i^{(t)}$ together with the set of public images $\{y_i^{(t)}\}_{i\in\{1\ldots n\}}$ it holds. From lemma 9.2.1, majority of servers stores the same correct set $Y = msety_i^{(t)} i \in \{1 \ldots n\}$, so every server can locally verify every submitted share x_i against the y_i that is included in the majority of the submitted sets Y. Then each server can interpolate the correct shares to the secret x and broadcast it on the communication channel. The outside observer recognizes the secret x as the value broadcasted by the majority of the servers.

6.2 Dynamically Secure Private Keys

In the above presentation we have assumed for simplicity that the servers are equipped with ideally protected private / public key pairs used for authentication and encryption of server-to-server communication (see section 4.1). We now show how to remove this protected key assumption. We extend the update phase between time periods to include a third component, the *private key renewal* protocol, which will be triggered before share recovery and share renewal. As a result of the private key renewal, an adversary that breaks into a server in period t, but which does not control the server at period $t+1$, cannot learn this server's new key.

The private key renewal protocol at the beginning of each update phase works as follows: Each server P_i chooses a new pair of private and public keys $a_i^{(t)}, b_i^{(t)}$ and broadcasts the new public key $b_i^{(t)}$ authenticated by its signature using its *previous* private key $a_i^{(t-1)}$. The other servers can verify this signature, using $b_i^{(t-1)}$ from the previous time period. Clearly, an adversary that controlled the server at time period $t-1$, or before, but not during the update phase between periods $t-1$ and t, cannot learn the new private key chosen by the server. However, if the adversary knows $a_i^{(t-1)}$ then, even if she is not controlling P_i during the private key renewal protocol of period t, she can choose her own private key and inject its public counterpart into the broadcast channel, authenticated as if it originated from P_i. But since P_i is not actively controlled by the adversary anymore, it will send its own authenticated public key to the communication channel as well. This will result in two different messages legally authenticated as coming from P_i, which will constitute a public proof of P_i's compromise and must trigger a reboot procedure.

When a server is rebooted, it internally chooses its new private key, publishing only the corresponding public key, which must be then installed on all servers in A. At the same time, public keys $\{a_i^{(t)}\}$ of other servers must be installed on the rebooted server. Notice that installing these public authentication keys requires the same degree of trust in the system management as during the initialization of the system.

6.3 The Combined Protocol

Combining all the above pieces we get our full protocol for proactive secret sharing: At the beginning of every time period, secret sharing servers trigger an *Update Phase*, which consists of the following stages:

1. Private Key Renewal
2. Share Recovery (including Lost Share Detection)
3. Share Renewal

Notice that the above protocol is in fact a protocol for proactive storage of the secret. The secret-sharing initialization and the secret reconstruction protocols are discussed in sections 4.2 and 6.1.

Chapter 7

Proactive Secret Sharing with Pedersen's VSS

In sections 4, 5 and 6 we presented the proactive secret sharing protocol using Feldman's VSS scheme since Feldman's VSS is simpler than the one presented by Pedersen in [Ped91], and hence makes the ideas behind our proactive protocols easier to understand. However, as we claim in section 3, Feldman's scheme reduces the secrecy protection of our protocol to semantic security *relative* to the prior knowledge of $g^x \ (mod\ p)$. In this section, we present the initialization, share renewal and share recovery parts of the alternative proactive secret sharing protocol which uses Pedersen's instead of Feldman's VSS. The secret reconstruction protocol described in section 6.1 stays the same for the version with Pedersen's VSS (except that the meaning of the public versions of the shares is slightly different). The private key renewal protocol also stays the same as in section 6.2, because maintainance of communication keys is orthogonal to the issue of whether we use Feldman's or Pedersen's VSS mechanism for renewal and recovery of shares.

7.1 Initialization

We extend the initial requirements described in section 4.2 so that all servers in $\mathcal{A}$ know a pair of numbers g, h in Z_p^* but even the (computationally bounded) adversary controlling up to the minority of the servers does not know $d = log_g h$. We can make the servers agree on such a pair (g, h) in a following way: Every server broadcasts a random pair (g_i, h_i) in Z_p^*, collects other servers' broadcasts and computes $g = \prod_{i\in\{1...n\}} g_i \ (mod\ p)$ and $h = \prod_{i\in\{1...n\}} h_i \ (mod\ p)$.

In addition to knowing g and h, every server P_i holds a share $\{x_i^{(0)}, z_i^{(0)}\}$ consisting of two numbers in Z_q such that there exist two k degree polynomials $f(\cdot)$ and $b(\cdot)$ in Z_q where $f(0) = x$ and for all $i \in \{1 \ldots n\}$, $f(i) = x_i$, $b(i) = z_i$ (Pedersen describes in [Ped91] how to initialize this setting securely). Also, every server P_i stores the set of public versions $\{y_i^{(0)}\}_{i\in\{1\ldots n\}}$ of the secret shares, but they have a different meaning now:

$$y_i = g^{x_i} h^{z_i} \ (mod\ p)$$

7.2 Share Renewal

We present the new version of the share renewal protocol for each P_i at period t

1. P_i picks $2k$ random numbers $\{\delta_{im}, \gamma_{im}\}_{m\in\{1\ldots k\}}$ from Z_q to define polynomials $\delta_i(z) = \delta_{i1}z^1 + \delta_{i2}z^2 + \ldots + \delta_{ik}z^k$ and $\gamma_i(z) = \gamma_{i1}z^1 + \gamma_{i2}z^2 + \ldots + \gamma_{ik}z^k$ in Z_q.

2. P_i computes values $\epsilon_{im} = (g)^{\delta_{im}}(h)^{\gamma_{im}}$ $(mod\ p)$ for each $m \in \{1\ldots k\}$ and values $u_{ij} = \delta_i(j)$ $(mod\ q)$ and $w_{ij} = \gamma_i(j)$ $(mod\ q)$ for each $j \in \{1\ldots n\}$

3. P_i broadcasts message $VSS_i^{(t)} = (i, t, \{\epsilon_{im}\}_{m\in\{1\ldots k\}}, \{ENC_j(u_{ij}, w_{ij})\}_{j\in\{1\ldots n\}\setminus\{i\}})$ and the signature $SIG_i(VSS_i^{(t)})$.

4. For all messages broadcasted in the previous step by other servers, P_i decrypts its shares, i.e. $\{u_{ji}, w_{ji}\}_{j\in\{1\ldots n\}\setminus\{i\}}$, and verifies their correctness by checking for all $j \neq i$
$$g^{u_{ji}} h^{w_{ji}} \overset{?}{=} \prod_{m\in\{1\ldots k\}} (\epsilon_{jm})^{i^m} \ (mod\ p) \tag{7.1}$$

5. If P_i finds all the messages $VSS_j^{(t)}, j \neq i$ to be correct, then it broadcasts a signed acceptance message. Otherwise, just like in the protocol 4.4, if there is something wrong with message $VSS_j^{(t)}$, P_i either automatically marks P_j as bad and/or starts an accusation protocol against it as described in section 4.5, with the only difference that the subject of the accusation is a pair (u_{ji}, w_{ji}) instead of a single value u_{ji}.

6. Let $\mathcal{B}_i$ be the set of servers that P_i marked as "bad" in the above process (If all servers broadcasted acceptance messages then $\mathcal{B}_i$ is empty). Then P_i performs the following updates: First it computes its own new share:
$$x_i^{(t)} \leftarrow x_i^{(t-1)} + \sum_{j\notin\mathcal{B}_i} u_{ji} \ (mod\ q) \ , \ z_i^{(t)} \leftarrow z_i^{(t-1)} + \sum_{j\notin\mathcal{B}_i} w_{ji} \ (mod\ q)$$
and then it updates its set $\{y_j\}_{j\in\{1\ldots n\}}$ by assigning for all j
$$y_j^{(t)} \leftarrow y_j^{(t-1)} * \prod_{a\notin\mathcal{B}_i} \left(\prod_{m\in\{1\ldots k\}} (\epsilon_{am})^{j^m} \right) \ (mod\ p)$$
$$\text{(which is equal to } \ y_j^{(t-1)} * \prod_{a\notin\mathcal{B}_i} g^{u_{aj}} h^{w_{aj}} \ (mod\ p) \ )$$

Finally, P_i erases all the variables it used in this protocol, except for its current share $(x_i^{(t)}, z_i^{(t)})$ and its new set $\{y_j^{(t)}\}_{j\in\{1\ldots n\}}$.

7.3 Share Recovery

The lost share detection protocol from section 5.1 changes only minimally: Every server P_i verifies whether its share is consistent with the value y_i it stores, by checking whether $g^{x_i}h^{z_i} \overset{?}{=} y_i \ (mod\ p)$. Depending on this outcome, P_i either broadcasts a request for share recovery or broadcasts the set $\{y_j\}_{j\in\{1\ldots n\}}$.

In the lost share detection protocol, every honest server will arrive at the same set $\mathcal{B}$ of servers that need share recovery. Let $\mathcal{D} = \mathcal{A} \setminus \mathcal{B}$. The new protocol for recovery of the share of each server P_r, $r \in \mathcal{B}$ is as follows:

1. Each $P_i, i \in \mathcal{D}$ picks two random k-degree polynomials $\delta_i(\cdot), \gamma_i(\cdot) \in Z_q[z]$ such that $\delta_i(r) = \gamma_i(r) = 0$ by picking random coefficients $\{\delta_{ij}, \gamma_{ij}\}_{j\in\{1\ldots k\}}$ in Z_q and computing $\delta_{i0} = -\sum_{j\in\{1\ldots k\}} \delta_{ij} r^j \ (mod\ q)$ and $\gamma_{i0} = -\sum_{j\in\{1\ldots k\}} \gamma_{ij} r^j \ (mod\ q)$.

2. Each P_i verifiably secret-shares $\delta_i(\cdot), \gamma_i(\cdot)$ among servers in $\mathcal{D}$ by broadcasting $VSS_i = (i, \{g^{\delta_{im}} h^{\gamma_{im}} \ (mod\ p)\}_{m\in\{0\ldots k\}}, \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j\in\mathcal{D}})$, together with $SIG_i(VSS_i)$.

3. Every server receives the above broadcasts and decrypts the parts that are encrypted under its public key. Then for all pairs P_i, P_j of servers in $\mathcal{D}$, P_j checks whether $\delta_i(r) = \gamma_i(r) = 0 \ (mod\ q)$ by verifying

$$\prod_{m\in\{0\ldots k\}} (g^{\delta_{im}} h^{\gamma_{im}})^{(r^m)} \overset{?}{=} 1 \ (mod\ p) \tag{7.2}$$

   and whether $(\delta_i(j), \gamma_i(j))$ is consistent with commitments broadcasted in VSS_i

$$g^{\delta_i(j)} h^{\gamma_i(j)} \overset{?}{=} \prod_{m\in\{0\ldots k\}} (g^{\delta_{im}} h^{\gamma_{im}})^{(j^m)} \ (mod\ p) \tag{7.3}$$

4. Depending on the above verification the servers broadcast acknowledgments or start accusation protocols just like in protocol 5.3. By public resolution of the accusations, each server in $\mathcal{D}$ and the server P_r decide on set $\mathcal{D}' \subset \mathcal{D}$ of servers that properly constructed and distributed their re-randomization polynomials $\delta_i(\cdot)$ and $\gamma_i(\cdot)$.

5. Each server $P_i, i \in \mathcal{D}'$ creates its new share of (x_r, z_r), $x_i' = x_i + \sum_{j\in\mathcal{D}'} \delta_j(i)$, $z_i' = z_i + \sum_{j\in\mathcal{D}'} \gamma_j(i)$ and sends it encrypted and signed to P_r by broadcasting $REC_i = (i, ENC_r(x_i', z_i'))$ and $SIG_i(REC_i)$ on the communication channel.

6. P_r decrypts these broadcasts, takes exponents $\{g^{\delta_{jm}} h^{\gamma_{jm}} \ (mod\ p)\}_{j \in \mathcal{D}', m \in \{0 \dots k\}}$ that were broadcasted in step 2 and verifies whether:

$$g^{x'_i} h^{z'_i} \overset{?}{=} y_i * \prod_{j \in \mathcal{D}'} g^{\delta_j(i)} h^{\gamma_j(i)} \ (mod\ p) \tag{7.4}$$

(which should be equal to $g^{x_i + \sum_{j \in \mathcal{D}'} \delta_j(i)} \ h^{z_i + \sum_{j \in \mathcal{D}'} \gamma_j(i)} \ (mod\ p)$)

where for all $j \in \mathcal{D}'$:

$$g^{\delta_j(i)} h^{\gamma_j(i)} \leftarrow \prod_{m \in \{0 \dots k\}} (g^{\delta_{jm}} h^{\gamma_{jm}})^{(i^m)} \ (mod\ p)$$

7. In this way P_r arrives at a set $\mathcal{D}'' \subseteq \mathcal{D}'$ of servers that in step 5 broadcasted correct new shares (x'_i, z'_i). Now P_r can interpolate $\{x'_i\}_{i \in \mathcal{D}''}$ to recover x_r and then interpolate $\{z'_i\}_{i \in \mathcal{D}''}$ to recover z_r.

Chapter 8

Applications: Proactive Public Key Cryptosystems

Proactive secret sharing can be used to store any sensitive information. However, as mentioned in the introduction (section 1.2), proactive secret sharing is most useful for sharing of cryptographic functions. The functions that we know how to proactively secret share are all ElGamal-like public key cryptosystems in which $x \in Z_q$ is a secret key, and $y = g^x \ (mod\ p)$ is its public counterpart. These functions were described by ElGamal in [ElG85], while their distributed versions were first presented in [DF90]. It is a hard problem to proactivize RSA-based public key cryptosystems. So far we have designed only an exponential-cost proactive RSA signature scheme [JJKY95]. Even though this solution has a communication and computation cost exponential in the number of secret sharing servers, since the number of servers in function sharing is usually small (like 3 to 5), this solution is not inefficient in practice.

PROACTIVE FUNCTION SHARING: The notion of function sharing is well formalized in [DDFY94] (also, see [DF90]). We will sketch it here as follows. Let $f(\cdot) : D \rightarrow I$ be the function we want to share. A $(k+1, n)$ threshold function sharing scheme of $\mathcal{F}(\cdot)$ consists of algorithms *Gen* and *Com*. *Gen* takes $\mathcal{F}(\cdot)$ and creates n partial functions $\mathcal{F}_i(\cdot) : D \rightarrow I_i$ for $i \in \{1 \ldots n\}$. *Com* is a public algorithm such that for any argument $m \in D$ and for any sequence of $(k+1)$ elements $\{i_1, \ldots, i_{k+1}\} \subset \{1, \ldots, n\}$, *Com* combines the results of the partial functions applied to m into the result of $\mathcal{F}$ on the same input: $Com[\mathcal{F}_{i_1}(m), \mathcal{F}_{i_2}(m), \ldots \mathcal{F}_{i_{k+1}}(m)] = \mathcal{F}(m)$.

For example, we can share the exponentiation function $\mathcal{F} : Z_q \rightarrow Z_p$, $\mathcal{F}(m) = m^x \ (mod\ p)$ as follows: *Gen* takes x and creates n shares $\{x_i\}_{i \in \{1 \ldots n\}}$ of x with $(k+1, n)$ Shamir's secret sharing with a k degree polynomial $f(\cdot)$ in Z_q. Each function $\mathcal{F}_i : Z_q \rightarrow Z_p$ is defined as $\mathcal{F}_i(m) = m^{x_i} \ (mod\ p)$. Then *Com* takes the results of $(k+1)$ functions (let $\{i_1, \ldots, i_{k+1}\}$ be their indexes) and computes

$$\begin{aligned} Com[\mathcal{F}_{i_1}(m), \mathcal{F}_{i_2}(m), \ldots \mathcal{F}_{i_{k+1}}(m)] &= (\mathcal{F}_{i_1}(m))^{a_{i_1}} (\mathcal{F}_{i_2}(m))^{a_{i_2}} \ldots (\mathcal{F}_{i_{k+1}}(m))^{a_{i_{k+1}}} \\ &= m^{a_{i_1} x_{i_1}} * m^{a_{i_2} x_{i_2}} * \ldots * m^{a_{i_{k+1}} x_{i_{k+1}}} \end{aligned}$$

$$\begin{aligned} &= m^{a_{i_1}x_{i_1}+a_{i_2}x_{i_2}+\ldots a_{i_{k+1}}x_{i_{k+1}}} \\ &= m^x \ (mod\ p) \\ &= \mathcal{F}(m) \end{aligned}$$

where $\{a_{i_1}, a_{i_2}, \ldots, a_{i_{k+1}}\}$ are Lagrange interpolation coefficients, i.e. $a_{i_1}x_{i_1} + a_{i_2}x_{i_2} + \ldots a_{i_{k+1}}x_{i_{k+1}} = x \ (mod\ q)$.

Notice that in the example above, function $\mathcal{F}$ is defined by the secret element x, and similarly the partial functions $\{\mathcal{F}_i\}_{i\in\{1\ldots n\}}$ are defined by shares $\{x_i\}_{i\in\{1\ldots n\}}$. In this sense, function sharing is an extension of secret sharing. Now, if we maintained shares $\{x_i\}_{i\in\{1\ldots n\}}$ of x proactively (renewing them every every time period), algorithm Com would still produce $\mathcal{F}(m) = x^m \ (mod\ p)$ as long as majority of the shares $\{x_i^{(t)}\}_{i\in\{1\ldots n\}}$ correspond to x for each time period t, and as long as there is a way of telling apart a partial $\mathcal{F}_i(m)$ computed with the correct share x_i from the result of applying some incorrect partial function to m. We call such a combination of function sharing and proactive maintainance of shares, a *proactive function sharing.*

We give two examples of public key cryptosystems in which the above mechanism of sharing the exponentiation function is combined with proactive maintainance of shares of the secret exponent. The first example is a proactively secure Key Certification Authority implemented with proactive sharing of the ElGamal signature function. In the forthcoming paper [HJJ+95] we give more examples of proactive signature sharing. The second example is a proactively secure database implemented with proactive sharing of the ElGamal decryption function.

8.1 Proactively Secure Certification Authority

A particularly attractive application (presented in [HJJ+95]) of proactive function sharing provides *proactive digital signatures* which achieve the benefits of threshold signatures, with the additional property that the scheme is broken only if the adversary corrupts more than a threshold of the servers in a *single* time-period. For signature keys that live for long time and require very strong security, like keys of network Certification Authority, this solution is particularly suitable.

8.1.1 Background on ElGamal signatures

There are many ElGamal-like signature schemes that can be performed in this setting [NR94, HPM94], but we present the one used by Harn [Har94] and based on [AMV90], because it allows for efficient distribution of the signature function. The party who knows x can sign a message $m \in Z_q$ by picking a random number $k \in Z_q$ and computing:

$$r = g^k \ (mod\ p)$$

$$s = x * m - k * r \ (mod\ q)$$

The signature on m is the pair (r, s) which can be verified by equation

$$y^m \overset{?}{=} r^r * g^s \ (mod\ p)$$

because if the pair (s, r) is computed correctly from x and some k, then $y^m = g^{x*m} \ (mod\ p)$ and $r^r * g^s = g^{k*r} * g^{x*m-k*r} = g^{x*m} \ (mod\ p)$ too.

8.1.2 Distributed Signature Operation

During every time period (t), the system of n servers that proactively secret share x can process outside requests to issue the above signatures. When an outside party submits a request to sign message m by broadcasting it on the communication channel, this triggers each server $P_i \in A$ to verify with the local copy of the policy of this Certification Authority whether m should be signed or not. If it should then every server P_i performs the following protocol (which is only a slight revision of Harn's protocol given in [Har94]):

1. If $i \in \mathcal{D}_i$, (i.e. if P_i thinks that it belongs to the set of $k+1$ servers that should be issuing partial signatures in this round), P_i issues the first part of its partial signature by picking a random number $k_i \in Z_q$, computing:

$$r_i = g^{k_i} \ (mod\ p) \tag{8.1}$$

and broadcasting $(i, r_i, SIG_i(i, r_i))$.

2. Every server P_i in $\mathcal{A}$ collects the broadcasted values r_j and checks whether the set $\mathcal{E}$ of servers that properly broadcasted their r_j's is equal to $\mathcal{D}_i$. If not, P_i goes back to the first step of the protocol, but with a new set $\mathcal{D}_i \leftarrow \mathcal{E} \cup NEW(\mathcal{A} \setminus \mathcal{D}_i, k+1-|\mathcal{E}|)$, where $NEW(A, x)$ is a public function which returns x elements of set A.

3. Otherwise, if P_i sees that every server in $\mathcal{D}_i$ properly broadcasted their r_j, P_i computes the first part of the final signature:

$$r = \prod_{j \in \mathcal{D}_i} r_j \ (mod\ p) \tag{8.2}$$

4. Every server P_i such that $i \in \mathcal{D}_i$ computes its partial signature key a_i:

$$a_i = x_i * \prod_{j \in (\mathcal{D}_i \setminus \{i\})} \frac{j}{j-i} \ (mod\ q) \tag{8.3}$$

And using that key, issues the second part of its partial signature by computing:

$$s_i = a_i * m - k_i * r \ (mod\ q) \tag{8.4}$$

and broadcasting $(i, s_i, SIG_i(i, s_i))$.

5. Every server P_i in $\mathcal{A}$ collects these broadcasts and creates a new set $\mathcal{E}$ of servers in $\mathcal{D}_i$ that properly broadcasted their s_j's and for which the following verification equation holds:

$$(y_j^m)^{\prod_{u \in (\mathcal{D}_i \setminus \{j\})} \frac{u}{u-j} \ (mod\ q)} = g^{s_j} * r_j^r \ (mod\ p) \tag{8.5}$$

   Notice that if $\mathcal{D}_i = \mathcal{D}_j$ then $(y_j^m)^{\prod_{u \in (\mathcal{D}_i \setminus \{j\})} \frac{u}{u-j} \ (mod\ q)} = (g^m)^{a_j} \ (mod\ p)$.

   If $\mathcal{E} \neq \mathcal{D}_i$ then P_i recomputes its set $\mathcal{D}_i$: $\mathcal{D}_i \leftarrow \mathcal{E} \cup NEW(\mathcal{A} \setminus \mathcal{D}_i, k+1-|\mathcal{E}|)$ and goes back to the first step of the protocol.

6. Otherwise, i.e. if all partial signatures are correct, each P_i (for which $i \in \mathcal{D}_i$) assembles the second part of the final signature:

$$s = \sum_{j \in \mathcal{D}_i} s_j \ (mod\ q) \tag{8.6}$$

   and broadcasts (m, s, r).

The party that originated the request to sign m can take these signatures and verify them with y.

Verification of the signature: The (m, s, r) triple is a standard ElGamal signature and can be verified with the public key (p, g, y). Notice that if a set $\mathcal{D}$ of $k+1$ servers in $\mathcal{A}$ broadcasted the same (s, r), then $\forall_{i \in \mathcal{D}} \mathcal{D}_i = \mathcal{D}$. Therefore (all equations are modulo p):

$$\begin{aligned}
r^r * g^s &= (\prod_{j \in \mathcal{D}} r_j)^r * g^{(\sum_{j \in \mathcal{D}} s_j)} && \text{(from eq. (2) and (6))} \\
&= (\prod_{j \in \mathcal{D}} g^{k_j})^r * g^{\sum_{j \in \mathcal{D}} (a_j * m - k_j * r)} && \text{(from eqs. (1) and (4))} \\
&= g^{(\sum_{j \in \mathcal{D}} k_j) * r} * g^{m * (\sum_{j \in \mathcal{D}} a_j) - r * (\sum_{j \in \mathcal{D}} k_j)} && \text{(just algebra...)} \\
&= g^{m * (\sum_{j \in \mathcal{D}} a_j)} * g^{r * (\sum_{j \in \mathcal{D}} k_j)} * g^{-r * (\sum_{j \in \mathcal{D}} k_j)} && \text{(rearranging terms...)} \\
&= g^{m * (\sum_{j \in \mathcal{D}} a_j)} \\
&= g^{m * (\sum_{j \in \mathcal{D}} (x_j * \prod_{u \in (\mathcal{D} \setminus \{j\})} \frac{u}{u-j}))} && \text{(from eqs. (3))} \\
&= g^{m * x} && \text{(from Lagrange interpolation formula)} \\
&= y^m && \text{(because } y = g^x\text{)}
\end{aligned}$$

8.2 Proactively Secure Database

Instead of direct proactive secret sharing, we can protect the array of sensitive data by storing it encrypted with ElGamal-like public-key encryption on multiple servers, and proactively secret sharing the ElGamal decryption key. This application prove useful for storing not data files themselves, but an array of DES keys used for symmetric encryption of the data files that one wants to store securely.

The ElGamal encryption works as follows: The encryption of $m \in Z_q$ is a pair $(m * y^k, g^k)$ (both modulo p), where k is an element of Z_q picked at random and $y = g^x \ (mod\, p)$ is the public key. To decrypt, the owner of the secret key x can compute $m = (my^k) * (g^k)^(- x) = m * g^{xk-xk} \ (mod\, p)$. This decryption function $\mathcal{F}'(\cdot) : Z_p \times Z_p \to Z_p$, $\mathcal{F}'(c_1, c_2) = c_1 * x_2^{-x}$ can be shared in the same way as the exponentiation function $\mathcal{F}(\cdot)$ we used as an example above. Shares of x_i can be also maintained with proactive secret sharing using Feldman's VSS, and the two protocols can be combined into proactive sharing of ElGamal decryption function.

To ensure integrity of the data, we have to add a *data corruption detection protocol* to the update phase: The servers compare the hashes of their ciphertexts and decide by majority on the correct version. Since the ciphertext itself is not secret, the servers that have a correct version can broadcast it, so the servers that lost it can regain a correct ciphertext. If this proactively secure archive is used to store DES keys for encryption and decryption of data files, then to ensure the same level of integrity protection of the encrypted files themselves, these files should also be replicated and hashed, and their hashes should be periodically compared to provide detection of data corruption.

This scheme, just like proactive signature sharing from the above section, achieves the benefits of threshold function sharing with the additional property that the adversary can read the secret data, or destroy the data and/or the decryption key, only if she breaks to more than k servers during the same time period.

Chapter 9

Security Analysis

Organization

This section contains the proofs of theorems 3.1.0 and 3.2.0 which state the security properties of the two proactive secret sharing schemes we propose. These proofs are organized as follows:

We begin in **section 9.1** by proving theorem 4.3.1 about the correctness and the secrecy protection of the basic proactive share renewal algorithm presented in section 4.3. This algorithm contains the core of the full share renewal scheme, stripped of the verifiability mechanisms, and consequently secure only against a minority of gossip faults (i.e. against the adversaries that only eavesdrop on the secret information stored by the server, as opposed to *byzantine* faults that mean that the adversary who compromises the server can cause it to follow an algorithm of the adversary's choice). We present this theorem and its proof because they provide the intuition about the correctness and the security characteristics of the full proactive protocol.

In **section 9.2**, we prove the robustness of both proactive secret sharing schemes, i.e. the robustness parts of theorems 3.1.0 and 3.2.0.

In **section 9.3** we justify why we use encrypted broadcast for server to server communication during our protocols, in place of the private links assumption used in the original VSS schemes by Feldman and Pedersen. We will refer to this section in the **section 9.4**, in which we analyze the secrecy protection of the proactive protocol with Pedersen's VSS, building to the proof of the secrecy part of theorem 3.2.0.

Although in the presentations of our algorithms (chapters 4 to 7) we presented the version with Feldman's VSS first, in the proofs of secrecy, we will start with the version with Pedersen's VSS, because it allows for better organization of the proofs: In **section 9.5**, where we prove the secrecy part of theorem 3.1.0, i.e. the secrecy characteristics of our scheme with Feldman's VSS, we will use the theorems proven for the version with Pedersen's VSS in section 9.4.

Section 9.4 is itself divided into four major subsections: In **subsection 9.4.1** we analyze a slightly modified version of the share renewal protocol (in the version using Pedersen's VSS): To achieve information-theoretic secrecy in our protocol, we introduce an additional requirement on the encryption scheme we use, namely a *black-box* assumption. This section is designed to show that our scheme is essentially as secure as the original Pedersen's VSS scheme it uses. Furthermore, we will need these results in **subsection 9.4.2**, where we analyze the secrecy of the share renewal protocol when this special assumption about black-box encryption scheme is removed. In that section we will also formally define the notions of *envelope*, *semantically secure envelope*, *envelope semantically secure relative to knowledge of* g^x and *semantically secure secret sharing scheme.* We will end the section by proving that the adversary's view of the sequence of executions of the share renewal protocol is a semantically secure envelope of the secret x. In **subsection 9.4.3** we show that analogous proofs can be carried out with regards to repetitive executions of the share recovery protocol. Finally, in **subsection 9.4.4**, we show that the analogous proof goes through if the adversary's view includes the execution of the whole proactive algorithm, consisting of runs of proactive share renewal interleaved with runs of the proactive share recovery protocol.

9.1 Basic Proactive Share Renewal

In this section we will prove theorem 4.3.1 about the correctness and the security characteristics of the basic proactive share renewal algorithm as described in section 4.3. In that scheme, we assumed that the adversary can only *eavesdrop* on the servers that she compromised. We also assumed that the shares are transferred between servers with perfect secrecy on the links. We claimed that the adversary that can compromise no more then k servers during each time period, cannot learn anything about the secret, and furthermore, that the servers always hold shares of the correct secret.

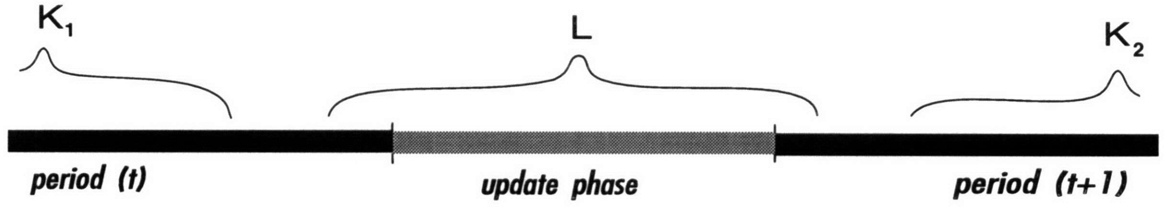
Proof of theorem 4.3.1: We proceed by an inductive argument where it is assumed that at initialization the shares correspond to the secret x according to Shamir's scheme. Furthermore, we assume that at each time period $r = 1, 2, .., t-1$ the theorem holds. In particular it means that after the update phase of time period $t-1$, the shares interpolate to the secret and that the adversary has learned nothing about the secret. We prove that this condition is preserved during time period t.

CORRECTNESS: Let S be a set of $k+1$ shares resulting from the t-th update phase. For simplicity of notation, assume $S = \{x_1^{(t)}, x_2^{(t)}, \ldots, x_{k+1}^{(t)}\}$. Let $a_1, a_2, \ldots, a_{k+1}$ be the Lagrange interpolation coefficients for set S, i.e. $\sum_{i\in\{1,\ldots,k+1\}} a_i f(i) = f(0)$ for any k-degree polynomial $f(\cdot)$. We have:

$$\sum_{i=1}^{k+1} a_i x_i^{(t)} = \sum_{i=1}^{k+1} a_i \left(x_i^{(t-1)} + \sum_{j=1}^{n} \delta_j(i) \right) \qquad \text{(by step 3 of the protocol)}$$

$$
\begin{aligned}
&= \sum_{i=1}^{k+1} a_i x_i^{(t-1)} + \sum_{j=1}^{n}\sum_{i=1}^{k+1} a_i \delta_j(i) \\
&= x + \sum_{j=1}^{n} \delta_j(0) && \text{(by interpolation)} \\
&= x && \text{(because } \delta_j(0) = 0 \text{ for all } j\text{)}
\end{aligned}
$$

SECRECY: Let A be an eavesdropping adversary. Let K_1 be the set of k_1 servers that A eavesdropped into in period $t-1$ but not in period t; let L be the set of l servers that A eavesdropped into both in period $t-1$ and in period t (we may assume that A eavesdropped into these servers during the update phase); and, let K_2 be the set of k_2 servers that A that eavesdropped into in period t but not in period $t-1$. By our assumption on the adversary, we have $k_1 + l \leq k$ and $l + k_2 \leq k$. We will assume a clear worst case when $k_1 = k_2 = k - l$. Here is a picture that shows which faults are counted as l, k_1 and k_2:



We now show that the availability of all this information about shares and updates does not provide information about x.

Notice that since we assume that k shares from period $t-1$ are known, fixing the secret x determines the interpolation polynomial $f^{(t-1)}$ corresponding to period $t-1$. Similarly, from the k known shares of period t, fixing x determines the polynomial $f^{(t)}$. By construction these polynomials are consistent with the available information from the set of shares available to the adversary (i.e. $\{x_i^{(t-1)}\}_{i \in K_1 \cup L}$ and $\{x_i^{(t)}\}_{i \in K_2 \cup L}$). On the other hand, the difference between these polynomials represent a k-degree polynomial with free coefficient zero and its evaluation on the points corresponding to the servers in L is consistent with the new shares available to the adversary. (They are consistent also with the value of the partial shares corresponding to the polynomials $\delta_i(z)$ which are randomly chosen conditioned only to $\delta_i(0) = 0$).

Since the above argument holds for *any* value of x, then all possible values of x are consistent with the available information. On the other hand, there is no degree of freedom beyond x (i.e., x and the known shares determine all the additional shares), and hence the distribution on x conditioned on the information available to A is uniform. In other words, no information on x is revealed. ■

9.2 Protection of Integrity

In this section we prove the robustness part of theorem 3.1.0 and 3.2.0, i.e. we prove the robustness properties of the two proactive secret sharing schemes we propose in this thesis. For the sake of the proof of robustness of the scheme with Pedersen's VSS, we state theorem 9.2.2 about the integrity protection of the original Pedersen's VSS scheme. In both proofs we will use the following lemma:

Lemma 9.2.1 *The servers that are not compromised by the adversary in time period t, hold the correct sets $Y_{correct} = \{y_i^{(t)}\}_{i \in \mathcal{A}}$ of public versions of correct secret shares $\{x_i^{(t)}, z_i^{(t)}\}_{i \in \mathcal{A}}$.*

The same is true of $Y_{correct}$ in the version with Feldman's VSS.

Proof: We prove it by induction. It is true at the initialization of the system, and we will show that if this is true at the beginning of one update phase, it will be true at the beginning of the next one:

Let $\mathcal{C} \subset \mathcal{A}$ and $\mathcal{C}' \subset \mathcal{A}$ be the servers that are not compromised during the time period $t-1$ and t respectively. In the share recovery protocol, each server in $\mathcal{C}$ broadcasts $Y_{correct}$ and every server in $\mathcal{C}'$ receives it and since by assumption the majority of broadcasted Y's are equal to $Y_{correct}$, each server in $\mathcal{C}'$ adopts it as a correct set Y. Later, in the share renewal protocol, each server in $\mathcal{C} \cup \mathcal{C}'$ will arrive at the same set $Y'_{correct}$ of new public versions of correct secret shares, because each honest server in the share renewal protocol has the same view of set $\mathcal{A} \setminus \mathcal{B}$ of servers whose update polynomials are correct (and constitute a correct update function for both the secret shares and their public versions in Y). Therefore, servers in $\mathcal{C}'$ will hold the correct new set $Y'_{correct}$ until the end of the next update phase. ■

Proof of the robustness part of theorem 3.1.0: We omit this proof: Throughout the description of the protocol (sections 4, 5 and 6) and in the proof of theorem 4.3.1 above, we already gave the argument why the mobile adversary cannot destroy more than k shares of x in every time period: In spite of the adversary's actions, every server that was not compromised in time period t, holds correct secret share $x_i^{(t)}$ of secret x. Furthermore, from lemma 9.2.1, every such server holds a correct set $Y_{correct}$ of public images of all correct shares $\{x_i^{(t)}, \ldots, i \in \{1 \ldots n\}\}$. This means that honest servers can reconstruct the secret following the protocol described in section 6.1. ■

INTEGRITY PROTECTION IN PEDERSEN'S VSS: Before we give the proof of the robustness part of theorem 3.2.0, we recount the original theorem that Pedersen proves about the integrity protection of his VSS scheme.

Theorem 9.2.2 *In Pedersen's VSS scheme as presented in 2.2, the dealer can distribute inconsistent shares if and only if he can find $log_g h$.*[1]

[1]However, see section 9.3

We call shares $\{\{u_i, w_i\}\}_{i\in\{1...n\}}$ *inconsistent* if there are no k-degree polynomials $\delta(\cdot), \gamma(\cdot)$ in Z_q s.t. $\delta(i) = u_i$ and $\gamma(i) = w_i$ for all $i \in \{1 \ldots n\}$.

In other words, Pedersen's VSS scheme gives only computational protection of integrity, i.e. the ability of the servers to verify whether the shares they receive are consistent, is a subject to computational limits imposed on the dealer.

The following important fact is used to prove the above theorem:

Lemma 9.2.3 *Being able to find two pairs $(a, b), (a', b')$ in Z_q such that $g^a h^b = g^{a'} h^{b'} \pmod p$ is equivalent to finding $log_g h$ over Z_p, because $log_g h = \frac{a-a'}{b'-b} \pmod q$.*

Proof of the robustness part of theorem 3.2.0: If we assume that the signature scheme $SIG(\cdot)$ is existentially unforgeable, then the adversary can render x "unreconstructible" if the following happens:

1. The new shares $\{(x_i, z_i)\}_{i\in\{1...n\}}$ computed at the end of the share renewal protocol are not consistent, or they are consistent but x_i's interpolate to some $x' \neq x$.

2. When the lost share detection protocol starts, an honest server P_i has a share (x_i, z_i) which is inconsistent with other shares, but the honest servers decide that P_i does not need share recovery.

3. In the share recovery protocol, the recovered share (x_r, z_r) of the honest P_r (who had lost its share in the previous time period) is not consistent with other shares.

How can any of this happen?

Ad(1): The new shares can be inconsistent only if some of the partial update polynomials are inconsistent and the honest servers fail to realize that. However, if some server P_i manages to distribute inconsistent shares $\{(u_{ij}, w_{ij})\}_{j\in\{1...n\}}$ for which the verification equation 7.1 holds for every j, then from theorem 9.2.2 it follows that the adversary controlling P_i can find $log_g h$.

Ad(2): In the lost share detection protocol, the only way a server cannot convince others that it needs a share recovery is when it thinks that its share is correct, i.e. if $g^{x_i} h^{z_i} = y_i \pmod p$ where y_i is held by the majority of the servers. From lemma 9.2.1, the majority of the broadcasted sets $\{y_j\}_{j\in A}$ at the beginning of the lost share detection protocol is correct, and hence P_i arrives by voting at a correct public version y_i of its share (x_i, z_i). If the adversary had invaded P_i during the time period and had changed its share to (x'_i, z'_i) s.t. $g^{x'_i} h^{z'_i} = y_i \pmod p$ but (x'_i, z'_i) is inconsistent with other shares, then $(x'_i, z'_i) \neq (x_i, z_i)$ while $g^{x_i} h^{z_i} = g^{x'_i} h^{z'_i} = y_i \pmod p$ which (from lemma 9.2.3) means that the adversary can find $log_g h$.

From the discussion of the consistency of the public versions y_i of secret shares in the separate paragraph below, it follows that the adversary cannot fool P_i to think that its modified share (x'_i, z'_i) is right by first fooling it to accept an incorrect y'_i.

Ad(3): In the share recovery protocol, P_r can reconstruct an inconsistent share (x_r, y_r) in the following three cases:

- First, if the re-randomization shares $\{\delta_i(j), \gamma_i(j)\}_{i,j\in\mathcal{D}'}$ received by servers in $\mathcal{D}'$ are consistent with some polynomials $\{\delta_i(\cdot), \gamma_i(\cdot)\}_{i\in\{1\ldots n\}\mathcal{D}'}$ for which $\delta_i(r) = \gamma_i(r) = 0$, but some server $P_i, i \in \mathcal{D}'$ sends to P_r a pair $(x_i'', z_i'') \neq (x_i + \sum_{j\in\mathcal{D}'} \delta_j(i), z_i + \sum_{j\in\mathcal{D}'} \gamma_j(i))$ for which equation 7.4 holds. Again, from lemma 9.2.3 this means that P_i can find $log_g h$.

- Second, if some server P_i in $\mathcal{D}'$ distributed inconsistent shares that passed the verification equation 7.3. It follows from theorem 9.2.2 directly that P_i can compute $log_g h$.

- Third, if the re-randomization shares $\{\delta_i(j), \gamma_i(j)\}_{i,j\in\mathcal{D}'}$ received by servers in $\mathcal{D}'$ are consistent with some polynomials $\{\delta_i(\cdot), \gamma_i(\cdot)\}_{i\in\mathcal{D}'}$ (i.e. if equation 7.3 holds for each $i \in \mathcal{D}'$), but for some $i \in \mathcal{D}'$, $(\delta_i(r), \gamma_i(r)) \neq (0, 0)$ even though the equation 7.2 still holds for that i. But this would mean that $g^{\delta_i(r)}h^{\gamma_i(r)} = g^0h^0 = 1 \ (mod\ p)$ and so, from lemma 9.2.3 it follows that P_i can find $log_g h = -\frac{\delta_i(r)}{\gamma_i(r)} \ (mod\ q)$.

■

9.3 Private Communication Links vs. Broadcast in VSS

Our protocols makes explicit the silent feature of VSS protocols which is not discussed by either Pedersen or Feldman, but which considerably complicates the issue of secrecy protection. It is the issue of private communication channels with which the shares are transfered from the dealer to the shareholders in both Feldman's and Pedersen's VSS schemes. In contrast, as we mentioned in section 2.2, in our solution we assume that broadcast channel is the only medium of communication between the servers in $\mathcal{A}$. To achieve privacy and authentication given by secret server-to-server communication links, we have to resort to public key encryption and signature system (as explained at the end of section 4.1).

Modification of Pedersen's and Feldman's VSS

Notice that in our proactive protocols we use a slightly modified version of VSS schemes as presented in section 2.2: We assume that the servers have a securely initialized public key encryption system. Then, instead of sending $Share_i$ privately to P_i, the dealer publicly broadcasts $ENC_i(Share_i)$. If server P_i checks its share with the verification equation and it does not agree with the publicly broadcasted commitments on the secret sharing polynomial (or a pair of polynomials in the case of Pedersen's VSS), P_i starts an accusation protocol, as described in section 4.5.

The reason why we modify the VSS schemes so that all communication is via broadcast channel is that we claim that these schemes will not otherwise work (in the sense that the claims of theorem 9.2.2 cannot be true) for the number of byzantine faults $k \geq \frac{n}{3}$. If we want our proactive secret sharing system to work for $n = 2k + 1$, we have to modify the underlying VSS schemes as described above, and only then the above theorems will apply. To make the VSS scheme more similar to our proactive secret sharing setting, let's agree that the dealer belongs to the n share holding servers $\mathcal{A}$ (and always leaves one share of the secret for itself). Assume that k of the servers are dishonest, including the dealer. To see why sending shares on private communication channels cannot work when $\frac{n}{3} \leq k < \frac{n}{2}$, consider the following scenario:

The dealer sends no message at all to $n - 2k$ of the honest servers (let's call these servers $\mathcal{N}$), and consistent shares (consistent with the publicly broadcasted commitments) to both the other k honest servers (let's call them $\mathcal{H}$) and to the remaining $k - 1$ dishonest servers (let's call them $\mathcal{D}$). Just before the protocol ends, servers $\mathcal{D}$ can always erase their shares. Consequently, $\mathcal{H}$ cannot accept this dealing as correct, because at the end there could be only k consistent shares of x, and hence x will be lost (if k was the reconstructibility threshold, the adversary could learn the secret by compromising k servers). Since servers $\mathcal{H}$ do not see what, if anything, was sent to servers $\mathcal{N}$, the protocol must have the following two clauses:

- If a server does not receive a consistent share from the dealer, it broadcasts a *protest* message. In particular, when the dealer does not send to this server anything at all, this server does not have anything more than a protest message to convince others that it was cheated.

- If a server receives at least $n - 2k$ protest messages, it decides that the dealer cheated.

However, such a protocol is insecure in the face of the following behavior of the dishonest servers: Let $\mathcal{D}'$ be the set of k dishonest servers and let the dealer be honest this time. Then after the distribution of shares, servers $\mathcal{D}'$ will broadcast their protest messages and, since there are $k \geq n - 2k$ of them, the honest servers will decide that the dealer is dishonest. In this way, the adversary can always prevent successful sharing of the secret.

If $k < \frac{n}{3}$ then the above protocol will work with reconstructibility threshold $k+1$: If $k < \frac{n}{3}$ then $n - 2k > k$, and hence, if a server receives $n - 2k$ protest messages, some of them must be sent by honest servers, which means that the dealer cheated. On the other hand, when a bad dealer sends $n - 2k - 1$ bad shares to the honest servers $\mathcal{N}$ and both the dealer and the servers $\mathcal{D}$ erase their shares, servers $\mathcal{H}$ will have enough shares to reconstruct x, because $|\mathcal{H}| = (n - k) - (n - 2k - 1) = k + 1$.

For surveys of the bounds on the adversary and the computational assumptions they necessitate for VSS algorithms, we send the reader to [Rab88], [RB89] and [BGW88].

Consequences for Secrecy Protection

Because we want our scheme to work for $n = 2k + 1$, we substitute sending shares over private server-to-server links by sending them encrypted and authenticated (with public key system) on a common broadcast channel. In this way, the recipient P of a secret share s who is cheated by the dealer, can prove this to all the honest servers connected to the communication channel, by following the accusation protocol we describe in section 4.5. This protocol uses the dealer's broadcast of s encrypted under the public key of P as its commitment to sending s privately to P.

This change, however, reduces the secrecy protection of both schemes, but most notably Pedersen's, since in Feldman's scheme, g^x is revealed to the adversary anyway. To prove that his VSS scheme gives information theoretic secrecy of x, Pedersen uses the fact that the shares $\{u_i, w_i\}$ are sent on private channels in an essential way: Let $\mathcal{B} \subset \mathcal{A}$ be the set of servers compromised by the adversary during the dealing of the secret ($|\mathcal{B}| \leq k$). Since the shares are sent with perfect privacy, the view $View$ of the adversary in theorem 9.4.3 contains only the set of shares $\{\delta(i), \gamma(i)\}_{i \in \mathcal{B}}$ and the set of broadcasted commitments $\{g^{\delta_m} h^{\gamma_m}\}_{m \in \{0 \ldots k\}}$ [2]. If we extend this view by the set of publicly visible encryptions of the remaining shares $\{ENC_i(\delta(i), \gamma(i))\}_{i \in \mathcal{A} \setminus \mathcal{B}}$, then the claim of theorem 9.4.3 that for all $x_0 \in Z_q$

$$Pr(x = x_0 \mid View) = Pr(x = x_0)$$

is no longer true: Since every encrypted message decrypts to a specific share, only one set of shares $\{\delta(i), \gamma(i)\}_{i \in \{1 \ldots n\}}$, and consequently, only one value x_0 is consistent with $View$.

Instead of information theoretic secrecy, the protection of the secrecy of x in our modified VSS protocols, becomes a subject to the strength of the encryption scheme $ENC(\cdot)$ we use (in addition to the information-hiding strength of the publicly broadcasted commitment on the shares, which in Pedersen's VSS does not give out any information on x, but in Feldman's VSS gives out g^x).

We should note that to achieve $n = 2k + 1$ bound, we could in principle also use the VSS mechanism proposed by Tal Rabin in [Rab88]. We choose not to, because this solution assumes broadcast *and* secret links between the servers and its communication costs are much bigger (as opposed to Feldman's and Pedersen's VSS, it is an interactive scheme). However, since [Rab88] does not base her results on cryptographic assumptions, it is possible that one would prefer it over VSS mechanisms we use, and so one possible extension of this work would be to check whether her VSS

[2]To avoid redundancy, we will omit that these exponentiations are always computed $(mod\ p)$

algorithm can also be adopted to the proactive setting.

9.4 Protection of Secrecy in the version using Pedersen's VSS

Organization

The proof of the semantic security of the proactive protocol with Pedersen's VSS (the secrecy part of theorem 3.2.0) has to be broken down into smaller steps:

In section 9.3 we justify why our protocol achieves different secrecy protection (i.e. semantic security) then Pedersen's VSS on which we built our scheme (i.e. information theoretic secrecy). In section 9.4.1 we show that under special assumptions (which were overlooked by Pedersen), our share renewal protocol also offers information theoretic secrecy. In section 9.4.2 we remove the special assumptions introduced in the section before, we define all the notions of *semantic security* which we will use, and we give the formal proof of the semantic security of the share renewal protocol.

In sections 9.4.1 and 9.4.2, we discuss only the secrecy properties of the share renewal protocol. Later, in section 9.4.3, we prove that this level of secrecy protection is preserved during the share recovery. Finally, in section 9.4.4 we show that the whole proactive algorithm, i.e. the repeated execution of both share renewal and share recovery protocols, has the same security properties.

9.4.1 Share Renewal with Black-Box Encryption

From the discussion in the previous section, we conclude that our scheme differs from the original VSS scheme of Pedersen by having:

- *a different communication model*: broadcast with explicit encryption instead of private links
- *a different model of the adversary*: our scheme can support up to $\frac{n}{2}$ byzantine faults as opposed to up to $\frac{n}{3}$ byzantine faults or up to $\frac{n}{2}$ gossip faults.

In this section we argue that if we modified our share renewal protocol to conform to the original setting of Pedersen's VSS protocol, it would achieve the same level of secrecy, i.e. information theoretical secrecy.

For the purpose of this section only, we conform our scheme to Pedersen's setting by making a following abstract assumption: We assume that the encryption scheme $ENC(\cdot)$ we use for broadcasting shares is a *black box encryption* scheme, i.e. the ciphertext gives no (information theoretical) information about the cleartext to the party that does not know the decryption key. We do not claim that there is a possible efficient black-box encryption scheme that can be used in our proactive setting (in particular, it *can not* be a public key scheme): we use it as an abstract construction whose only purpose is to make the broadcasting of the encrypted shares in a single run of the share renewal protocol look exactly like sending them on secure private links:

- The adversary who does compromise one of the end parties can not learn anything from the ciphertext.
- The ciphertext cannot be used as a commitment of the sender on the cleartext.

In section 9.4.2 this abstraction is removed and the proofs we give here are extended for the more realistic case of $ENC(\cdot)$ being a semantically secure encryption scheme.

The share renewal protocol with black box encryption cannot use the accusation mechanism to fight against up to $\frac{n}{2}$ byzantine faults. We have to modify the share renewal algorithm in section 7.2 in step 5: Instead of accusations, the cheated servers can broadcast only protest messages (just like in section 9.3) and servers decide on set $\mathcal{B}$ of bad servers non-interactively: if P_1 receives at least $\frac{n}{3}$ protest messages against P_2, P_1 marks P_2 as bad.

Consequently, the share renewal and share recovery protocols (and the VSS protocol by itself too) using this encryption is secure either against $k < \frac{n}{2}$ faults that are only gossip faults (in other words, the adversary cannot disable the attacked servers from performing the correct protocol) or only $k < \frac{n}{3}$ byzantine faults. Because our aim is to prove the secrecy of our scheme, we pick the first model of the adversary: there are still $k < \frac{n}{2}$ of corrupted servers, but we agree that they cannot actively cheat. In particular, in step 6, the set of servers that P_i believes to be faulty, is always empty, i.e. $\mathcal{B}_i = \{\}$.

With these assumptions, we prove that the periodical share renewal protocol preserves information theoretic secrecy of the secret x. Notice that if we assume only gossip faults, the share recovery algorithm is spurious and the periodical share renewal protocol will constitute the whole proactive update protocol. We state it formally as follows:

Theorem 9.4.1 *If in the share renewal protocol we use Pedersen's commitment scheme and we assume that the encryption scheme used in transfer of shares is a black-box (i.e. information-theoretically secure) encryption, then an adversary who compromises up to $k < \frac{n}{2}$ servers in a time period does not learn anything about the secret x:*

$$Pr(x = x_0 \mid \{View^{(t)}\}_{t\in\{1,\ldots,t_o\}}) = Pr(x = x_0)$$

for any x_0 in Z_q and any integer t_0, where $View^{(t)}$ is adversary's view during the update between time period $t-1$ and t.

We will break the proof of the above into several steps. First we state a lemma 9.4.4 in which we reformulate Pedersen's claim of secrecy of his VSS scheme, to adapt it to the way in which every server uses the VSS mechanism to distribute its partial update polynomial. In lemma 9.4.5 we extend the claim of lemma 9.4.4 to the total update polynomial, which is the sum of all the partial update polynomials. In lemma 9.4.6 we show that the public information $\{y_i\}_{i\in\{1\dots n\}}$ about the shares of the secret is equivalent to the public information about the coefficients of the secret sharing polynomial as is available in the original Pedersen's scheme. We use these lemmas to prove theorem 9.4.2, which says that a single execution of the share renewal protocol (in a version using black box encryption) preserves information theoretic secrecy of the secret x. Lastly, we extend the proof of theorem 9.4.2 to cover the case of multiple executions of the share renewal protocol, in this way proving the full theorem 9.4.1.

Theorem 9.4.2 *The view of the adversary during a single run of the share renewal protocol between time period $t-1$ and t, does not give her any information about the secret x, i.e:*

$$Pr(x = x_0 \mid View^{(t)}) = Pr(x = x_0)$$

for any x_0 in Z_q.

In other words, $View^{(t)}$ does not bias the probability that $x = x_0$ to any value x_0 in Z_q.

We use the notation from section 9.1:

- K_1 is the set of k_1 servers that are corrupted in period $t-1$ but not in period t
- K_2 is the set of k_2 servers that are corrupted in in period t but not $t-1$
- L is the set of l servers that are corrupted during the update phase between period $t-1$ and t, and can be counted as corrupted during both these periods

As in the proof of theorem 4.3.1, we assume a clear worst case when $k_1 = k_2 = k - l \neq 0$. Without loss of generality, we assume that $L = \{1, \dots, l\}, K_1 = \{l+1, \dots, k\}, K_2 = \{k+1, \dots, 2k-l\}$. Since the encryption scheme $ENC_i(\cdot)$ is a black-box, we will ignore it: it carries no information to the adversary who does not know the secret decryption key of party P_i. Therefore the adversary's view during this update can be summed up as follows:

$$View^{(t)} = (Shares^{(t-1)}, Shares^{(t)}, Ims^{(t-1)}, Ims^{(t)}, \{Full_i\}_{i\in L}, \{Partial_i\}_{i\notin L}) \quad (9.1)$$

where:

$$\begin{aligned} Shares^{(t-1)} &= \{x_i^{(t-1)}, z_i^{(t-1)}\}_{i\in L\cup K_1} \\ Shares^{(t)} &= \{x_i^{(t)}, z_i^{(t)}\}_{i\in L\cup K_2} \end{aligned}$$

$$\begin{aligned}
Ims^{(t-1)} &= \{y_i^{(t-1)}\}_{i\in\mathcal{A}} \\
Ims^{(t)} &= \{y_i^{(t)}\}_{i\in\mathcal{A}} \\
Full_i &= \delta_i(\cdot),\gamma_i(\cdot) \\
Partial_i &= \{\delta_i(j),\gamma_i(j)\}_{j\in L},\{g^{\delta_{im}}h^{\gamma_{im}}\}_{m\in\{1\ldots k\}}
\end{aligned}$$

We picked the "$Partial_i, Full_i$" notation to make a distinction between the update polynomials distributed by servers $P_i \notin L$ about whom the adversary knows only a partial information (i.e. the shares received by servers in L and the commitments on the coefficients), and the update polynomials distributed by servers $P_i \in L$ about whom the adversary knows everything (since the adversary controls servers in L).

The original theorem that Pedersen proves about the secrecy protection of his VSS scheme in [Ped91] is as follows:

Theorem 9.4.3 *If $PedView$ is the view of the adversary who compromises no more than k servers during an execution of the original Pedersen's VSS scheme as presented in 2.2, then for any x_0 in Z_q*

$$Pr(x = x_0 \mid PedView) = Pr(x = x_0)$$

In other words, Pedersen's VSS scheme gives perfect (information theoretical) secrecy protection of x, because adversary's view $PedView$ (up to k shares $\{u_i, w_i\}$ and all exponents $\{\epsilon_i\}_{i\in\{0\ldots k\}}$) is consistent with x being equal to any value in Z_q.

Unfortunately, we cannot apply this theorem directly to our scheme because we use Pedersen's VSS scheme in a very peculiar way: Namely, every server P_i performs the VSS protocol to secret share a *publicly known value* of zero. We cannot apply theorem 9.4.3 directly to VSS schemes performed by servers $P_i \notin L$ because they pick only k coefficients of polynomial $\gamma_i(\cdot)$ at random (for all i, $\gamma_{i0} = 0$). Also, it does not make sense to prove anything about secrecy protection of a number which everybody knows to be equal to zero, since for all i, $\delta_i(0) = 0$. Instead, we will prove the following lemma, very similar to theorem 9.4.3, but useful for our scheme:

Lemma 9.4.4 *Let $V = \{v_j\}_{j\in K_1}$ be a set of any $k - l$ elements in Z_q. Then:*

$$Pr(\forall_{j\in K_1}\delta_i(j) = v_j \mid Partial_i) = Pr(\forall_{j\in K_1}\delta_i(j) = v_j)$$

In other words, we claim that $k - l$ values of every polynomial $\delta_i(\cdot)$, $i \notin L$ are hidden from the adversary with perfect, information theoretical secrecy.

Proof: This proof will be very similar to Pedersen's proof of theorem 9.4.3. All equations are modulo q, unless otherwise noted.

We will show that there exists a pair of k-degree polynomials $Poly_V = (\delta_i^*(\cdot), \gamma_i^*(\cdot))$ in Z_q where $\delta_i^*(0) = 0, \gamma_i^*(0) = 0$ and for every $j \in K_1$, $\delta_i^*(j) = v_j$, such that $Poly_V$ is consistent with $View^{(t)}$ in the following sense:

$$\begin{aligned}
\forall_{j\in L} \quad \delta_i^*(j) &= \delta_i(j) && (9.2)\\
\forall_{j\in L} \quad \gamma_i^*(j) &= \gamma_i(j) && (9.3)\\
\forall_{m\in\{1\ldots k\}} \quad g^{\delta_{im}^*}h^{\gamma_{im}^*} &= g^{\delta_{im}}h^{\gamma_{im}} \ (mod\ p) && (9.4)
\end{aligned}$$

where $\{\delta_{im}^*, \gamma_{im}^*\}_{m\in\{1\ldots k\}}$ are coefficients of polynomials $Poly_V$.

Notice that Pedersen's commitments $\{g^{\delta_{im}}h^{\gamma_{im}}\}_{m\in\{1\ldots k\}}$ uniquely define a k-degree polynomial $G_i(\cdot)$ in Z_q whose coefficients $\{G_{im}\}_{m\in\{1\ldots k\}}$ are such that $g^{G_{im}} = g^{\delta_{im}}h^{\gamma_{im}}$ $(mod\ p)$. Since P_i followed the protocol, we have that

$$G_i(\cdot) = \delta_i(\cdot) + d\gamma_i(\cdot) \qquad (9.5)$$

where $d = log_g h\ (mod\ p)$.

Since $\delta_i^*(\cdot)$ is a k-degree polynomial with free coefficient zero, setting $\delta_i^*(j) = \delta_i(j)$ for every $j \in L$ (to satisfy equation 9.2) and assigning $\delta(j) = v_j$ for every $j \in K_1$ defines $\delta_i^*(\cdot)$. Now we define $\gamma_i^*(\cdot) = \frac{G_i(\cdot)-\delta_i^*(\cdot)}{d}$. From equation 9.5 it follows that $\delta_i^*(\cdot) + d\gamma_i^*(\cdot) = \delta_i(\cdot) + d\gamma_i(\cdot)$, and hence, $\gamma_i^*(0) = 0$ and also, for every $m \in \{1\ldots k\}$, $\delta_{im}^* + d\gamma_{im}^* = \delta_{im} + d\gamma_{im}$, from which equation 9.4 follows. Equation 9.3 is satisfied too, because for every $j \in L$:

$$\begin{aligned}
\gamma_i^*(j) &= \frac{G_i(j) - \delta_i^*(j)}{d} && \text{(from the definition of } \gamma_i^*(\cdot))\\
&= \frac{(\delta_i(j) + d * \gamma_i(j)) - \delta_i^*(j)}{d} && \text{(from equation 9.5)}\\
&= \frac{(\delta_i(j) + d * \gamma_i(j)) - \delta_i(j)}{d} && \text{(from the definition of } \delta_i^*(\cdot))\\
&= \gamma_i(j)
\end{aligned}$$

This means that $Poly_V$ is consistent with $View^{(t)}$, and consequently it proves the lemma. ■

The next lemma follows immediately:

Lemma 9.4.5 *For all well-formed sets $\{Full_i\}_{i\in L}, \{Partial_i\}_{i\notin L}$, and any set $V = \{v_j\}_{j\in K_1}$ of elements in Z_q*

$$Pr(\forall_{j\in K_1}\delta(j) = v_j \mid \{Full_i\}_{i\in L}, \{Partial_i\}_{i\notin L}) = Pr(\forall_{j\in K_1}\delta(j) = v_j)$$

where $\delta(\cdot) = \sum_{i\in\mathcal{A}}\delta_i(\cdot)$

In other words, every set of $k - l$ values $\{\delta(j)\}_{j\in K_1}$ is equally possible given the information contained in $\{Full_i\}_{i\in L}, \{Partial_i\}_{i\notin L}$

Proof: Let i_0 be some element in $\mathcal{A} \setminus L$. From lemma 9.4.4, the values $\{\delta_{i_0}(j)\}_{j\in K_1}$ can be equal to any set $V = \{v_j\}_{j\in K_1}$ of elements in Z_q. Even if every polynomial $\delta_i(\cdot), i \neq i_0$ is fully determined by $\{Full_i\}_{i\in L}, \{Partial_i\}_{i\notin L}$ (say, for $i \neq i_0, j \in K_1$, $\delta_i(j) = w_{ij}$), the crucial values of the total update polynomial $\{\delta(j)\}_{j\in K_1}$ can be equal to any set $V' = \{v'_j\}_{j\in K_1}$. Otherwise, if say, $\delta(j_0)$ can't be equal to some v then it follows that $\delta_{i_0}(j_0)$ can't be equal to $v - \sum_{i\neq i_0} w_{ij_0}$, which is a contradiction. ■

In the proof of theorem 9.4.2 we will need one more lemma:

Lemma 9.4.6 *For any k-degree polynomials $f(\cdot), b(\cdot)$ in Z_q with free coefficient zero, sets $S_1 = \{g^{f_m}h^{b_m}\}_{m\in\{1...k\}}$ and $S_2 = \{g^{f(i)}h^{b(i)}\}_{i\in\mathcal{A}}$ contain the same knowledge.*

Proof: The claim of the lemma follows from standard interpolation properties of polynomials: For all i, we have that $g^{f(i)}h^{b(i)} = g^{\sum_{m\in\{1...k\}}(f_m)^{i^m}} h^{\sum_{m\in\{1...k\}}(g_m)^{i^m}} = \prod_{m\in\{1...k\}}(g^{f_m}h^{g_m})^{i^m}$ which shows that from S_1 one can compute S_2. To show that the reverse is also true, note that the above equation is based on the linear relation between values $\vec{v_f} = \{f(1), \ldots, f(n)\}$, $\vec{v_r} = \{b(1), \ldots, b(n)\}$ and coefficients $\vec{c_f} = \{f_1, \ldots, f_k\}$, $\vec{c_r} = \{b_1, \ldots, b_k\}$ by equations $\vec{v_f} = A* \vec{c_f}$ and $\vec{v_r} = A* \vec{c_r}$, where $A = [a_{im}]_{i\in\{1...n\}}^{m\in\{1...k\}}$ is a k-by-n matrix of numbers $a_{im} = i^m$.

For any k-element subset $I = \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$, there is a k-by-k matrix $A' = [a_{im}]_{i\in I}^{m\in\{1...k\}}$ which relates $\vec{v'_f} = \{f(i_j)\}_{j\in\{1...k\}}$ and $\vec{v'_r} = \{b(i_j)\}_{j\in\{1...k\}}$ to $\vec{c_f}$ and $\vec{c_r}$ by equations $\vec{v'_f} = A'* \vec{c_f}$ and $\vec{v'_r} = A'* \vec{c_r}$. Since A' is a Vandermonde matrix, it is reversible, and $\vec{c_f} = (A')^{-1}* \vec{v'_f}$ and $\vec{c_r} = (A')^{-1}* \vec{v'_r}$. If $\vec{a} = \{a_1, \ldots, a_k\}$ is the m-th row of $(A')^{-1}$ then $f_m = \vec{a} * \vec{v'_f}$, and since it is a linear relation, it can be carried over in the exponent, i.e. $g^{f_m} = \prod_{j\in\{1...k\}}(g^{f(i_j)})^{(a_j)}$ and similarly, $h^{b_m} = \prod_{j\in\{1...k\}}(h^{b(i_j)})^{(a_j)}$. It follows that

$$g^{f_m}h^{b_m} = \prod_{j\in\{1...k\}} (g^{f(i_j)}h^{b(i_j)})^{(a_j)}$$

which shows that S_1 can be computed from S_2 and completes the proof of the lemma.

■

Proof of theorem 9.4.2: We will show that for every value $x = x_0$ in Z_q, there exists a family of k-degree polynomials: $\mathcal{F}_{x_0} = f^{(t-1)}(\cdot), f^{(t)}(\cdot), r^{(t-1)}(\cdot), r^{(t)}(\cdot)$, where $f^{(t-1)}(0) = f^{(t)}(0) = x_0$, such that $\mathcal{F}_{x_0}$ is consistent with $View^{(t)}$ in the sense that

1. $f^{(t-1)}(i) = x_i^{(t-1)}$, $r^{(t-1)}(i) = z_i^{(t-1)}$ for all $i \in K_1 \cup L$
2. $f^{(t)}(i) = x_i^{(t)}$, $r^{(t)}(i) = z_i^{(t)}$ for all $i \in K_2 \cup L$
3. $g^{(f^{(t-1)}(i))}h^{(r^{(t-1)}(i))} = y_i^{(t-1)}$ for all $i \in \mathcal{A}$
4. $g^{(f^{(t)}(i))}h^{(r^{(t)}(i))} = y_i^{(t)}$ for all $i \in \mathcal{A}$

5. $\delta(\cdot) = f^{(t)}(\cdot) - f^{(t-1)}(\cdot)$ and $\gamma(\cdot) = r^{(t)}(\cdot) - r^{(t-1)}(\cdot)$ are consistent with information given in $\{Full_i\}_{i\in L}, \{Partial_i\}_{i\notin L}$.

From lemma 9.4.6, we get $Pr(x = x_0 \mid \{f^{(t)}(i), r^{(t)}(i)\}_{i\in K_2\cup L}, \{g^{(f_m^{(t)})}h^{(r_m^{(t)})}\}_{m\in\{1\ldots k\}}) = = Pr(x = x_0 \mid Shares^{(t)}, Ims^{(t)})$. Therefore, from theorem 9.4.3, we have that $Pr(x = x_0 \mid Shares^{(t)}, Ims^{(t)}) = Pr(x = x_0)$. This means that there exist polynomials $f^{(t)}(\cdot), r^{(t)}(\cdot)$, where $f(0) = x_0$, that satisfy constraints 1. and 3. Similarly, there exist polynomials $f^{(t-1)}(\cdot), r^{(t-1)}(\cdot)$, where $f^{(t-1)}(0) = x_0$ that are consistent with $Shares^{(t-1)}, Ims^{(t-1)}$ and x_0, and which satisfy constraints 2. and 4.

We will show that constraint 5. is satisfied too. First we show that polynomial $\delta^*(\cdot) = \sum_{i\in\mathcal{A}} \delta_i(\cdot)$ (as constrained by information given in $\{Full_i\}_{i\in L}, \{Partial_i\}_{i\notin L}$) can be equal to $\delta(\cdot) = f^{(t)}(\cdot) - f^{(t-1)}(\cdot)$: From correctness of the share renewal algorithm, for $j \in L$, $\delta^*(j) = \sum_{i\in\mathcal{A}} \delta_i(j) = x_j^{(t)} - x_j^{(t-1)}$, which, from definition of $f^{(t)}(\cdot)$ and $f^{(t-1)}(\cdot)$, is equal to $f^{(t)}(j) - f^{(t-1)}(j) = \delta(j)$. Lemma 9.4.5 means that values $\{\delta^*(j)\}_{j\in K_1}$ can be any set of $k - l$ elements in Z_q. Hence, in particular we can have $\delta^*(j) = \delta(j)$ for $j \in K_2$. Since $\delta^*(\cdot), \delta(\cdot)$ are k-degree polynomials, with free coefficient zero, if they have k common values, they must be equal.

Notice that $\delta^*(\cdot)$ and $\{Full_i\}_{i\in L}, \{Partial_i\}_{i\notin L}$ define the second update polynomial $\gamma^*(\cdot)$ because there is only one k-degree polynomial $\gamma^*(\cdot)$ in Z_q whose coefficients $\{\gamma_m^*\}_{m\in\{1\ldots k\}}$ satisfy equation $g^{\delta_m^*}h^{\gamma_m^*} = \prod_{j\in\mathcal{A}} g^{\delta_{jm}}h^{\gamma_{jm}} = g^{(\sum_{j\in\mathcal{A}}\delta_{jm})}h^{(\sum_{j\in\mathcal{A}}\gamma_{jm})}$ for $m \in \{1\ldots k\}$. We will show that this $\gamma^*(\cdot)$ is equal to $\gamma(\cdot) = r^{(t)}(\cdot) - r^{(t-1)}(\cdot)$:

For every $i \in \mathcal{A}$ we have:

$$\begin{aligned}
g^{\delta(i)}h^{\gamma(i)} &= g^{f^{(t)}(i)-f^{(t-1)}(i)}h^{r^{(t)}(i)-r^{(t-1)}(i)} && \text{(from our definition of } \delta(\cdot) \text{ and } \gamma(\cdot)) \\
&= (g^{f^{(t)}(i)}h^{r^{(t)}(i)}) * (g^{f^{(t-1)}(i)}h^{r^{(t-1)}(i)})^{(-1)} \\
&= (y_i^{(t)}) * (y^{(t-1)})^{(-1)} && \text{(from theorem 9.4.3)} \\
&= \prod_{j\in\mathcal{A}}(\prod_{m\in\{1\ldots k\}} (g^{\delta_{jm}}h^{\gamma_{jm}})^{i^m}) && \text{((*) from step 6 in section 7.2)} \\
&= \prod_{m\in\{1\ldots k\}} \left(\prod_{j\in\mathcal{A}} g^{\delta_{jm}}h^{\gamma_{jm}}\right)^{i^m} \\
&= \prod_{m\in\{1\ldots k\}} (g^{\delta_m^*}h^{\gamma_m^*})^{i^m} && \text{(from our definition of } \delta^*(\cdot)) \\
&= g^{(\sum_{m\in\{1\ldots k\}}\delta_m^* i^m)}h^{(\sum_{m\in\{1\ldots k\}}\gamma_m^* i^m)} \\
&= g^{\delta^*(i)}h^{\gamma^*(i)}
\end{aligned}$$

In step $(*)$ we use the fact that $View^{(t)}$ is a view of a correct execution of the share renewal protocol, which means that values $\{y_i^{(t-1)}, y_i^{(t)}\}_{i\in\mathcal{A}}$ in $Ims^{(t-1)}, Ims^{(t)}$ and values $\{g^{\delta_{jm}}h^{\gamma_{jm}}\}_{i\in\mathcal{A},m\in\{1\ldots k\}}$ in $\{Full_i\}_{i\in L}$ and $\{Partial_i\}_{i\notin L}$ must be related by equation in step 6 of the share renewal protocol (section 7.2).

From the above reasoning, and from the fact that function $F : Z_q \to Z_p$, $F(x) = g^x \ (mod\, p)$ is one-to-one, it follows that for all $i \in \mathcal{A}$, $\delta(i) + d\gamma(i) = \delta^*(i) + d\gamma^*(i)$. Since we proved that $\delta(\cdot) = \delta^*(\cdot)$, it follows that for all $i \in \mathcal{A}$, $\gamma(i) = \gamma^*(i)$. Since these are polynomials of degree k, it follows that $\gamma(\cdot) = \gamma^*(\cdot)$. This shows that polynomials $\mathcal{F}_{x_0}$ are consistent with $View^{(t)}$, and hence completes the proof of the lemma. ■

Proof of theorem 9.4.1: We can extend the same argument as in the proof of theorem 9.4.2 to the case of multiple runs of the share renewal protocol, i.e. when the adversary's view is a set $Views = \{View^{(t)}\}_{t \in \{1,\ldots,t_0\}}$ of single share renewal views, for arbitrarily big integer t_0. Namely, we show that $Views$ is consistent with any value $x_0 \in Z_q$ of secret x. We slightly modify the notation:

$$Views = (Shares^{(0)}, Ims^{(0)}, \{View^{(t)}\}_{t \in \{1,\ldots,t_0\}}) \tag{9.6}$$

where

$$\begin{aligned}
View^{(t)} &= (Shares^{(t)}, Ims^{(t)}, \{Partial_i^{(t)}\}_{i \notin L^{(t)}}, \{Full_i^{(t)}\}_{i \in L^{(t)}}) \\
Shares^{(t)} &= \{x_i^{(t)}, z_i^{(t)}\}_{i \in L^{(t)} \cup K^{(t)} \cup L^{(t+1)}} \\
Ims^{(t)} &= \{y_i^{(t)}\}_{i \in \mathcal{A}} \\
Partial_i^{(t)} &= \{\delta_i^{(t)}(j), \gamma_i^{(t)}(j)\}_{j \in L^{(t)}}, \{g^{\delta_{i,m}^{(t)}} h^{\gamma_{i,m}^{(t)}}\}_{m \in \{1\ldots k\}} \\
Full_i^{(t)} &= \delta_i^{(t)}(\cdot), \gamma_i^{(t)}(\cdot)
\end{aligned}$$

where:

- $L^{(t)}$ is the set of servers compromised by the adversary during the update between time period $t-1$ and t. For correctness of the above notation, $L^{(t_0)} = \{\}$.
- $K^{(t)}$ is the set of servers compromised by the adversary in the tth time period, but not during any of the adjoining two updates. In particular, $K^{(t-1)} \cap L^{(t)} = L^{(t)} \cap K^{(t)} = \{\}$.

Similarly to a single update, the worst cases are when for all t, $|L^{(t)}| + |K^{(t)}| + |L^{(t+1)}| = k$. If we set x to x_0, then each pair $Shares^{(t)}, Ims^{(t)}$ defines a pair of k-degree polynomials $\mathcal{F}_{x_0}^{(t)} = (f^{(t)}(\cdot), r^{(t)}(\cdot))$. The same proof as in the theorem 9.4.2 can be applied to show that for all t, a pair $\mathcal{F}_{x_0}^{(t-1)}, \mathcal{F}_{x_0}^{(t)}$ is consistent with the update information available to the adversary in $\{Partial_i^{(t)}\}_{i \notin L^{(t)}}, \{Full_i^{(t)}\}_{i \in L^{(t)}}$. This ends the proof of the theorem, because a family $\{\mathcal{F}_{x_0}^{(t)}\}_{t \in \{1,\ldots,t_0\}}$ which is consistent with $Views$ can be constructed for any x_0 in Z_q. ■

9.4.2 Share Renewal with Semantically Secure Encryption

As we discussed in section 9.4.1, the assumption of black-box encryption is unnatural: We introduced it to show a general result that the proactive share renewal algorithm protects secrecy and integrity of x just as well as the original Pedersen's VSS scheme. We will use these results in the proofs that follow.

Definition of Semantically Secure Envelopes

In the proofs in this section we want to show that the information $Info$ available to the adversary is a semantically secure envelope of the values V that we do not want the adversary to learn. We define this notion below, following the definition of semantic security for encryption schemes given in [Gol89]. Using this notion we can formalize the intuitive definition of what it means for a proactive secret sharing protocol to be semantically secure, that we sketched in section 2.3. We will concern ourselves only with a *non-uniform* notion of semantic security.

Both $Info$ and V are some (sets of) entities that appear in our protocol (for example the secret x, the encrypted message $ENC_3(\delta_2^{(t_0)}(3), \gamma_2^{(t_0)}(3))$, the sets of initial shares $ENC_{x_i^{(0)}, z_i^{(0)}}(i \in \{1 \ldots n\})$ etc). We treat $Info$ and V as random variables: their values range as specified in the protocol and their probability distributions are taken over random choices also as specified by the protocol (in all protocols here we assume the servers use random number generators).

Definition 9.4.1 *Let $Info$ and V be some (sets of) entities appearing in the protocol P. For every value v of V, we define $Info(v)$ to be a random variable s.t.*

$$Prob(Info(v) = x) = Prob(Info = x \mid V = v)$$

where probability is taken over random choices as specified by the protocol P.

We call $Info(v)$ an envelope of *v. We also refer to $Info$ as to an* envelope scheme on *V by denoting it as $Info(\cdot)$.*

The definition of semantic security below relates the hardness of learning any knowledge about V from $Info$ to the length of information $Info$ in bits. In all our proofs, the length of $Info$ is linearly related to the length of the prime number q. We denote the length of q as h, i.e. $2^{h-1} \leq q < 2^h$. We call h the *security parameter.* To make sure that the bit amount of data available to the adversary is a linear function of h, we have to relate parameters k, n, p and the number t_0 of time periods for which the proactive secret sharing algorithm will execute. Number p is defined as the smallest prime number s.t. $p = mq + 1$ fore some (even) integer m. Since q is picked so that such p exists for small $m = 2, 4$ or 6, the length of numbers in Z_p will be at most $log_2 6 * h$. Parameter n is odd and we assume that $n \leq l_1(h)$ where $l(\cdot)$ is some linear function with integer coefficients. k is an integer s.t. $2k + 1 = n$. The maximal lifetime of the algorithm (counted in number of time periods) $t_0 = l_2(h)$ where $l_2(\cdot)$ is also a linear function with integer coefficients.

Let Q be an infinite sequence of all valid values of q. Both $Info$ and V are in fact a family of random variables $\{Info_q\}_{q \in Q}$ and $\{V_q\}_{q \in Q}$. Similarly, one can think of our protocols as of a family of protocols: different protocol for every $q \in Q$. We can also think of the envelope scheme defined in 9.4.1 as of a family of envelope schemes $\{Info_q(\cdot)\}_{q \in Q}$.

Definition 9.4.2 *Let $Info$ and V be some (sets of) entities appearing in protocol P. Let Q be an infinite sequence of valid values of q. We call $Info$ a* semantically secure envelope *of V in protocol P, if for every function λ, every function θ and every $\mathcal{BPP}$ algorithm A, there exists a $\mathcal{BPP}$ algorithm A' such that*

$$\begin{aligned}\forall_c \exists_{h_0} \forall_{q \text{ s.t. } |q| \geq h_0} Prob[A(Info_q(V_q), \theta(V_q), 1^{|q|}) = \lambda(V_q)] &\leq \\ Prob[A'(\theta(V_q), 1^{|q|}) = \lambda(V_q)] &+ \frac{1}{|q|^c}\end{aligned}$$

where the probability is taken over random choices of algorithms A and A' and over distribution of V_q (determined by random choices of algorithm P) and distribution of $Info_q(V_q)$ (also determined from definition 9.4.1 by random choices of P).

For simplicity of notation, we will ignore the $1^{|q|}$ input, assuming it is built into the description of algorithms A and A' for every q. Even though $1^{|q|}$ factor does not appear in the input and function $\theta(V_q)$ can be of sublinear size in $|q|$, by saying A and A' are $\mathcal{BPP}$ we will always mean that they are turing machines making random coin tosses (random choices) and it halts in at most $poly(|q|)$ steps, where $poly(\cdot)$ is some polynomial function.

We will also give the definition of envelope scheme secure in the sense of indistinguishability:

Definition 9.4.3 *Let $Info, V, Q, P$ be like above. We call $Info$ an* envelope of V secure in the sense of indistinguishability *in protocol P, if for every two sequences $\{v_q\}_{q \in Q}, \{v'_q\}_{q \in Q}$ of values of V and every $\mathcal{BPP}$ algorithm A*

$$\begin{aligned}\forall_c \exists_{h_0} \forall_{q \text{ s.t. } |q| \geq h_0} |Prob[A(v_q, v'_q, 1^{|q|}, Info_q(v_q)) = 1] &- \\ Prob[A(v_q, v'_q, 1^{|q|}, Info_q(v'_q)) = 1]| &\leq \frac{1}{|q|^c}\end{aligned}$$

where the probability is taken over random choices of algorithm A and over distributions of $Info_q(v_q)$ and $Info_q(v'_q)$ (determined from definition 9.4.1 by random choices of algorithm P).

Since both these definitions of secrecy of envelopes (9.4.2 and 9.4.3) are analogous to definitions of secrecy of encryption schemes, we will use without proof the theorem proved in [Gol89] for secrecy of encryption schemes that semantic security and probabilistic polynomial time indistinguishability are equivalent in the non-uniform case.

Now we can give the formal definition of semantically secure secret sharing scheme:

Definition 9.4.4 *Let P (or actually a family $\{P_q\}_{q \in Q}$) be a proactive secret sharing scheme. Assume that the number of servers n and the number of time periods t_0 for which the protocol lasts are both linear functions of $|q|$. Let $Info$ be all the information about the computations of P observed by the mobile adversary (who is defined in section 2.1) during the lifetime of the protocol. We call P a* semantically secure secret sharing scheme *if $Info$ is a semantically secure envelope scheme of secret x.*

The notion of relative semantic security is defined analogously:

Definition 9.4.5 *Let P and $Info$ be as above. Let ω be some function on x. We call P a secret sharing scheme* semantically secure relatively to *ω if for every function λ and θ and every $\mathcal{BPP}$ algorithm A, there exists a $\mathcal{BPP}$ algorithm A' such that*

$$\forall_c \exists_{h_0} \forall_{q \text{ s.t. } |q| \geq h_0} Prob[A(Info_q(X_q), \theta(X_q), 1^{|q|}) = \lambda(X_q)] \leq$$
$$Prob[A'(\omega(X_q), \theta(X_q), 1^{|q|}) = \lambda(X_q)] + \frac{1}{|q|^c}$$

where the probability is taken over random choices of algorithms A and A', over distribution of X_q (we denote secret x as X_q because we treat it as a collection of random variables for every $q \in Q$. X_q (i.e. secret x) is always uniformly distributed in Z_q) and distribution of $Info_q(X_q)$ (determined from definition 9.4.1 by random choices of algorithm P).

Changes from the Black Box Encryption Model

In this section we return to a share renewal protocol that uses a semantically secure public key encryption scheme $ENC(\cdot)$ for secret server-to-server communication on a broadcast channel. We also return to the original adversary model described in section 2.1, i.e. we remove the constraints on the adversary that were imposed because of the black-box encryption assumption in section 9.4.1, namely, that the adversary causes only gossip and not byzantine faults. Consequently, it is no longer true that all servers will correctly distribute their update polynomials: some servers among those compromised by the adversary can try to cheat in the protocol. However, since we have the (semantically secure) public-key encryption used to transfer the shares, the accusation protocols will work, theorem 9.2.2 applies, and hence, the proof of robustness in section 9.2 also applies. In particular, we know that the cheating servers will be always identified by the honest ones, provided that the adversary does not find $d = log_g h$ (in addition to the constrains described in section 2.1).

We will use the notation from section 9.4.1: Servers L are compromised during the update, while K_1 and K_2 are compromised before and after the update but not during. But now we have to add a new variable $\mathcal{D}$ to denote the set of servers that correctly distribute well-formed update polynomials during an update phase. From the correctness of the share renewal algorithm, $\mathcal{A} \setminus \mathcal{D} \subset L$, i.e. only the servers compromised by the adversary could have distributed their update polynomials incorrectly. From section 4.5 we know that whatever the adversary does, every honest server agrees on the set $\mathcal{D}$. Hence, in the proofs that follow, we will treat $\mathcal{D}$ as a variable picked by the adversary (just like L and K), with the constraint that $\mathcal{A} \setminus L \subset \mathcal{D} \subset \mathcal{A}$: The proofs must work for any such $\mathcal{D}$.

A NOTE ABOUT NON-INTERACTIVE MODEL OF THE ADVERSARY: In our definitions of security we formalize the adversary as a non-interactive $\mathcal{BPP}$ Turing machine. As we mention in section 2.3, this is a restricted model of mobile yet non-adaptive adversaries, because in reality, the adversary can decide what to do next (which servers to compromise) as a function of her current view of the computation. Because we will deal with only the restricted, non-adaptive model, we will always assume that the strategy of the adversary (i.e. the choices of sets $L^{(t)}, K^{(t)}$ and $D^{(t)}$ for all time periods t) is set before the execution of the proactive secret sharing algorithm begins.

A NOTE ABOUT USING ENCRYPTION $ENC(\cdot)$: In the discussion of proactive protocol using Pedersen's VSS, we use encryption scheme $ENC(\cdot)$ to always encrypt a pair of numbers in Z_q. When we specify the strength of secrecy protection of our protocols relative to the strength of encryption scheme $ENC(\cdot)$, we mean its strength on input of size $2|q|$.

Since we cannot ignore the publicly broadcasted encrypted shares, we extend the adversary's view $Partial_i$ of the update polynomial distributed by P_i, $i \notin L$:

$$Partial_i = \{\delta_i(j), \gamma_i(j)\}_{j\in L}, \{g^{\delta_{im}} h^{\gamma_{im}}\}_{m\in\{1\ldots k\}}, \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j\notin L} \qquad (9.7)$$

Lemma 9.4.7 *$Partial_i$ is a semantically secure envelope of values $V = \{\delta_i(j)\}_{j\in K_1}$.*

Proof: We will prove the lemma by contradiction. We assume that $Partial_i$, treated as an envelope scheme of V, is not secure in the sense of indistinguishability. Then, using a hybrid method (introduced in [GM84]) we show that encryption scheme $ENC(\cdot)$ is not secure in the sense of indistinguishability, which would mean that it is not a semantically secure encryption scheme.

Assume that there is a constant c such that for every $h_o \in Z$, there exists a prime q of length $h \geq h_0$, a $\mathcal{BPP}$ machine A and two sets $V_1 = \{v_j^1\}_{j\in K_1}$ and $V_2 = \{v_j^2\}_{j\in K_1}$ of values in Z_q such that:

$$|Prob[A(V_1, V_2, Partial_i(V_1)) = 1] - Prob[A(V_1, V_2, Partial_i(V_2)) = 1]| > \frac{1}{h^c} \qquad (9.8)$$

where the probability is taken over the random choices made by $Partial_i(\cdot)$.

We define an envelope $Partial_i'(\cdot)$ of values $V = \{\delta_i(j)\}_{j\in K_1}$, which looks just like $Partial_i$ in $\mathcal{P}_V$, except that instead of encryptions of shares $\{ENC_j(\delta_i(j), \gamma_i(j))\}_{j\notin L}$, $Partial_i'(\cdot)$ has random encryptions of zeroes: $\{ENC_j(0,0)\}_{j\notin L}$. In other words

$$Partial_i'(V) = \{\delta_i(j), \gamma_i(j)\}_{j\in L}, \{g^{\delta_{im}} h^{\gamma_{im}}\}_{m\in\{1\ldots k\}}, \{ENC_j(0,0)\}_{j\notin L}$$

where $\delta_i(\cdot), \gamma_i(\cdot)$ are uniformly picked polynomials which agree with values in V. Notice that $Partial'_i(V)$ contains exactly the same amount of information about $\delta_i(\cdot), \gamma_i(\cdot)$ as set $Partial_i$ from lemma 9.4.5: $k-l$ random encryptions of two zeroes clearly does not carry any additional information. Consequently, machine A cannot distinguish between random envelopes $Partial'_i(V_1)$ and $Partial'_i(V_2)$, because from lemma 9.4.5, $Partial'_i(\cdot)$ is an informational theoretic secure envelope scheme, and therefore $Partial'_i(V_1)$ and $Partial'_i(V_2)$ have the same probability distribution. Let's denote the following probabilities:

$$\begin{aligned}
p_1 &= Prob[A(V_1, V_2, Partial_i(V_1)) = 1] \\
p'_1 &= Prob[A(V_1, V_2, Partial'_i(V_1)) = 1] \\
p'_2 &= Prob[A(V_1, V_2, Partial'_i(V_2)) = 1] \\
p_2 &= Prob[A(V_1, V_2, Partial_i(V_2)) = 1]
\end{aligned}$$

where probabilities are taken over the coin tosses made by $Partial_i(\cdot)$ and $Partial'_i(\cdot)$.

From the discussion above we know that $p'_1 = p'_2$. From equation 9.8, $|p_1 - p_2| > \frac{1}{h^c}$. From triangle inequality, $|p_1 - p'_1| + |p'_1 - p_2| = |p_1 - p'_1| + |p_2 - p'_2| \geq |p_1 - p_2|$, which means that either $|p_1 - p'_1|$ or $|p_2 - p'_2|$ is bigger than $\frac{1}{2h^c}$. We can assume that $|p_1 - p'_1| > \frac{1}{2h^c}$; if the second case is true, the proof will be analogous.

We use the hybrid technique again to show that A can be used as a distinguisher of encryption scheme $ENC(\cdot)$. We define a sequence of $n-l+1$ random variables $\{P^{(s)}\}_{s\in\{0,\dots,n-l\}}$ which will fill the gap between the probability distribution of $Partial_i(V_1)$ and $Partial'_i(V_1)$: Each $P^{(s)}$ is created by uniformly picking some $\delta_i(\cdot), \gamma_i(\cdot)$ which are consistent with V_1 and creating a valid view $Partial_i$ with these polynomials, except that instead of last s encryptions of valid shares $\{\delta_i(j), \gamma_i(j)\}_{j\in\{n-s+1,\dots,n\}}$, it contains s random encryptions of a pair of two zeroes.

$$\begin{aligned}
P^{(0)}(V_1) &= Common, \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j\in\{l+1,\dots,n\}}, \\
P^{(1)}(V_1) &= Common, \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j\in\{l+1,\dots,n-1\}}, ENC_n(0,0) \\
P^{(2)}(V_1) &= Common, \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j\in\{l+1,\dots,n-2\}}, \{ENC_j(0,0)\}_{j\in\{n-1,n\}} \\
\dots &= \dots \qquad , \dots \qquad\qquad\qquad , \dots \\
P^{(n-l-1)}(V_1) &= Common, ENC_{l+1}(\delta_i(l+1), \gamma_i(l+1)), \{ENC_j(0,0)\}_{j\in\{l+2,\dots,n\}} \\
P^{(n-l)}(V_1) &= Common, \{ENC_j(0,0)\}_{j\in\{l+1,\dots,n\}}
\end{aligned}$$

where

$$Common = \{\delta_i(j), \gamma_i(j)\}_{j\in L}, \{g^{\delta_{im}} h^{\gamma_{im}}\}_{m\in\{1\dots k\}}$$

The probability distribution of $P^{(s)}(V_1)$ for each s is given by the fact that $P^{(s)}(V_1)$ depends on $n-l$ random encryptions $ENC(\cdot)$ and on $k+l$ random numbers $\{\delta_i(j)\}_{j\in L}$ and $\{\gamma_i(j)\}_{j\in L\cup K_1}$ which are picked uniformly from Z_q. We can assign probabilities to these variables by the following equation for every $s \in \{0, \dots, n-l\}$:

$$p_1^{(s)} = Prob[A(V_1, V_2, P^{(s)}(V_1)) = 1]$$

Clearly, $p_1 = p_1^{(0)}$ and $p'_1 = p_1^{(n-l)}$. From the triangle inequality

$$\sum_{s\in\{0,\ldots,n-l-1\}} |p_1^{(s+1)} - p_1^{(s)}| \geq |p_1^{(n-l)} - p_1^{(0)}| = |p'_1 - p_1| > \frac{1}{2h^c}$$

which means that there is $s_0 \in \{0,\ldots,n-l-1\}$ s.t. $|p_1^{(s_0+1)} - p_1^{(s_0)}| > \frac{1}{2(n-l)h^c}$.

We want to show that there must exist a pair v^* such that if schemes $P^{(s_0)}(V_1)$ and $P^{(s_0+1)}(V_1)$ are restricted to picking polynomials for which $(\delta_i(n-s_0), \gamma_i(n-s_0)) = v^*$, then $|Prob[A(V_1, V_2, P^{(s_0+1)}(V_1)) = 1] - Prob[A(V_1, V_2, P^{(s_0)}(V_1)) = 1]| > \frac{1}{2(n-l)h^c}$. Consider random variables $E(v), E'(v)$, for every $v \in Z_q$, which are defined just like $P^{(s_0)}(V_1), P^{(s_0+1)}(V_1)$ respectively, except that they take only these values of $P^{(s_0)}(V_1)$ and $P^{(s_0+1)}(V_1)$ for which $(\delta_i(n-s_0), \gamma_i(n-s_0)) = v$. For every $v \in Z_q \times Z_q$, set $\mathcal{D}^{(v)}$ of possible values of variable $E(v)$ is a subset of the set $\mathcal{D}$ of possible values of $P^{(s_0)}(V_1)$. Probability distribution of $E(v)$ inside $\mathcal{D}^{(v)}$ is determined by $k+l-2$ random choices of coefficients of $\delta_i(\cdot)$ and $\gamma_i(\cdot)$ in Z_q and $n-l$ random encryptions $ENC(\cdot)$. Notice that

$$\mathcal{D} = \mathcal{D}^{(0,0)} \cup \mathcal{D}^{(0,1)} \cup \ldots \cup \mathcal{D}^{(q-1,q-1)}$$

Moreover, because random variable $P^{(s_0)}(V_1)$ depends on polynomials $\delta_i(\cdot)$ and $\gamma_i(\cdot)$ selected with uniform distribution (subject to constraint that $\forall_{j\in K_1}\delta_i(j) = v_j$ where $\{v_j\}_{j\in K_1} = V_1$), the probability that $(\delta_i(n-s_0), \gamma_i(n-s_0)) = v$ is the same for every $v \in Z_q \times Z_q$, and hence the probability $Prob[P^{(s_0)}(V_1) \in \mathcal{D}^{(v)}]$ is the same for every $v \in Z_q \times Z_q$, i.e. equal to $\frac{1}{q^2}$. Therefore

$$\begin{aligned}
p_1^{(s_0)} &= Prob[A(V_1, V_2, P^{(s_0)}(V_1)) = 1] \\
&= \sum_{v\in Z_q\times Z_q} Prob[P^{(s_0)}(V_1) \in \mathcal{D}^{(v)}] * Prob[A(V_1, V_2, E(v)) = 1] \\
&= \frac{1}{q^2} * \sum_{v\in Z_q\times Z_q} Prob[A(V_1, V_2, E(v)) = 1]
\end{aligned}$$

Similar argument can be carried out valid for $E'(v)$, and hence

$$p_1^{(s_0+1)} = \frac{1}{q^2} * \sum_{v\in Z_q\times Z_q} Prob[A(V_1, V_2, E'(v)) = 1]$$

We know that $|p_1^{(s_0+1)} - p_1^{(s_0)}| > \frac{1}{2(n-l)h^c}$. Assume for a moment that $p_1^{(s_0+1)} > p_1^{(s_0)}$. Combining the two results above, we have

$$\begin{aligned}
&\sum_{v\in Z_q\times Z_q} (Prob[A(V_1, V_2, E'(v)) = 1] - Prob[A(V_1, V_2, E(v)) = 1]) &= \\
= \quad &\sum_{v\in Z_q\times Z_q} Prob[A(V_1, V_2, E'(v)) = 1] - \sum_{v\in Z_q\times Z_q} Prob[A(V_1, V_2, E(v)) = 1] &= \\
&= \quad q^2 * p_1^{(s_0+1)} - q^2 * p_1^{(s_0)} &>
\end{aligned}$$

$$> \quad q^2 * \frac{1}{2(n-l)h^c}$$

This means that there must be some value v^* in Z_q such that $Prob[A(V_1, V_2, E'(v^*)) = 1] - Prob[A(V_1, V_2, E(v^*)) = 1] > \frac{1}{2(n-l)h^c}$. If there is no such $v*$ then the sum

$$\sum_{v \in Z_q \times Z_q} Prob[A(V_1, V_2, E'(v)) = 1] - Prob[A(V_1, V_2, E(v)) = 1]$$

could never be bigger than $q^2 * \frac{1}{2(n-l)h^c}$. If we assumed that $p_1^{(s_0+1)} < p_1^{(s_0)}$ we would find an element v^* such that $Prob[A(V_1, V_2, E(v^*)) = 1] - Prob[A(V_1, V_2, E'(v^*)) = 1] > \frac{1}{2(n-l)h^c}$. In any case, there is an element v^* in Z_q for which

$$|Prob[A(V_1, V_2, E(v^*)) = 1] - Prob[A(V_1, V_2, E'(v^*)) = 1]| > \frac{1}{2(n-l)h^c} \tag{9.9}$$

Now we can use A to build a $\mathcal{BBP}$ machine A^* which distinguishes between random encryptions $ENC_{n-s_0}(0,0)$ of a pair of zeroes and random encryptions $ENC_{n-s_0}(v^*)$ of v^*. It takes as input a number e which is a result of a random encryption $ENC_{n-s_0}(\cdot)$ of either $(0,0)$ or v^*. It acts as follows: It finds two k-degree polynomials $\delta_i(\cdot), \gamma_i(\cdot)$ in Z_q such that $(\delta_i(n-s_0), \gamma_i(n-s_0)) = v^*$ and $\delta_i(j) = v_j$ for all $j \in K_1$, where $\{v_j\}_{j \in K_1} = V_1$. This involves picking $k + l - 2$ random numbers in Z_q. With these polynomials, A^* creates $P(s_0+1)(V_1)$, except that it substitutes e for a new random $ENC_{n-s_0}(0,0)$. In other words, it creates

$$\begin{aligned} P &= \{\delta_i(j), \gamma_i(j)\}_{j \in L}, \{g^{\delta_{im}} h^{\gamma_{im}}\}_{m \in \{1 \ldots k\}}, \\ &\quad \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j \in \{l+1, \ldots, n-s_0-1\}}, e, \{ENC_j(0,0)\}_{j \in \{n-s_0+1, n\}} \end{aligned}$$

Then it runs $A(V_1, V_2, P)$ and whatever A computes, it gives out as an output. Notice that so formed P has the probability distribution of $E'(v^*)$ if e is a random encryption of $(0,0)$ and the probability distribution of $E(v^*)$ if e is a random encryption of v^*. Consequently, from equation 9.9 we have

$$\begin{aligned} |Prob[A^*(ENC_{n-s_0}(0,0)) = 1] - Prob[A^*(ENC_{n-s_0}(v^*)) = 1]| &= \\ = \quad |Prob[A(V_1, V_2, E'(v^*)) = 1] - Prob[A(V_1, V_2, E(v^*)) = 1]| &> \frac{1}{2(n-l)h^c} \end{aligned}$$

which means that A^* is a $\mathcal{BPP}$ machine breaking the encryption scheme $ENC_{n-s_0}(\cdot)$ on inputs $(0,0)$ and v^*. This could be true only if $ENC_{n-s_0}(\cdot)$ is not a semantically secure encryption scheme. By contradiction, we conclude that no $\mathcal{BPP}$ distinguisher A of the envelope scheme $Partial_i(\cdot)$ can exist. In other words $Partial_i$ from equation 9.7 is a semantically secure envelope of values $V = \{\delta_i(j)\}_{j \in K_1}$, which ends the proof of the lemma. ■

It is easy to prove the following lemma, an equivalent of lemma 9.4.5 from the previous section:

Lemma 9.4.8 *Set $\{Partial_i\}_{i \in D \setminus L}, \{Full_i\}_{i \in D \cap L}$ (where $Partial_i$ is defined by equation 9.7) is a semantically secure envelope of values $V = \{\sum_{i \in D} \delta_i(j)\}_{j \in K_1}$.*

Proof: We use the original definition of semantic security (not the security in the sense of indistinguishability like in the proof of lemma 9.4.7). Even if for all $i \in D \setminus L$ but one $i_0 \in D \setminus L$, all encrypted information in $Partial_i$ were open, the adversary would learn $\{\delta_i(\cdot), \gamma_i(\cdot)\}_{i \in D \setminus \{i_0\}}$ and in particular values $v_j^* = \sum_{i \in D \setminus \{i_0\}} \delta_i(j)$ for all $j \in K_1$. Since $\forall_{j \in K_1} \sum_{i \in D} \delta_i(j) = v_j^* + \delta_{i_0}(j)$, it follows that if we assumed that the adversary seeing $\{Full_i\}_{i \in D \setminus \{i_0\}}, Partial_{i_0}$ could compute something about V (something that could not be computed without it), then it could compute something about $\{\delta_{i_0}(j)\}_{j \in K_1} = \{\sum_{i \in D} \delta_i(j) - v_j^*\}_{j \in K_1}$ with $Partial_{i_0}$ as an input (and again, the adversary would compute something which cannot be computed without $Partial_{i_0}$). This contradicts lemma 9.4.7, and consequently ends this proof.

Here is the formalization of the above argument:

Let's denote $\{Partial_i\}_{i \in D \setminus L}, \{Full_i\}_{i \in D \cap L}$ as an envelope scheme $Polys(\cdot)$ on V. Assume that $Polys(\cdot)$ is not a semantically secure envelope, i.e. assume that there is a $\mathcal{BPP}$ machine A and functions λ and θ, such that for every $\mathcal{BPP}$ algorithm A'

$$\exists_c \forall_{h_0} \exists_{q \text{ s.t. } |q| \geq h_0} Prob[A(Polys(V), \theta(V)) = \lambda(V)] > Prob[A'(\theta(V)) = \lambda(V)] + \frac{1}{|q|^c} \tag{9.10}$$

where the probability is taken over the random choices made by $A, A', Polys(\cdot)$ and over the distribution of V.

Let i_o be some element in $D \setminus L$. We will show that there must exist a set $V^* = \{v_j^*\}_{j \in L_1} \in Z_q^{(k-l)}$ such that if $Polys(V)$ is restricted to families of polynomials for which $\sum_{i \in D, i \neq i_0} \delta_i(j) = v_j^*$ for $j \in K_1$ then equation 9.10 still holds. We made an analogous argument in the proof of lemma 9.4.7, so we will omit it here, but the result is as follows: Let $Polys_{V^*}(V)$ be a random variable just like $Polys(V)$ except that it restricts the choice of the family of polynomials $\{\delta_i(\cdot), \gamma_i(\cdot)\}_{i \in D}$ by a constraint that $\sum_{i \in D \setminus \{i_0\}} \delta_i(j) = v_j^*$ for all $j \in K_1$. Resuming, V^* is picked (for every $q \in Q$) in such a way so that for every $\mathcal{BPP}$ machine A'

$$\exists_c \forall_{h_0} \exists_{q \text{ s.t. } |q| \geq h_0} Prob[A(Polys_{V^*}(V), \theta(V)) = \lambda(V)] > Prob[A'(\theta(V)) = \lambda(V)] + \frac{1}{|q|^c} \tag{9.11}$$

Now we construct a $\mathcal{BPP}$ machine C which (for every q) takes as input an envelope $Partial_{i_0}(V_2)$ where $V_2 = \{\delta_{i_0}(j)\}_{j \in K_1}$ is uniformly distributed in $Z_q^{(k-l)}$, and a knowledge function $\theta_2(V_2)$ where $\theta_2(\cdot)$ is defined (for every q) on $Z_q^{(k-l)}$ by equation $\theta_2(\{x_j\}_{j \in K_1}) = \theta(\{x_j + v_j^*\}_{j \in K_1})$. C uniformly picks a set $\{\delta_i(\cdot), \gamma(\cdot)\}_{i \in D \setminus \{i_0\}}$ with constraint that $\sum_{i \in D \setminus \{i_0\}} \delta_i(j) = v_j^*$ for all $j \in K_1$. From these polynomials it forms $\{Partial_i\}_{i \in D \setminus (L \cup \{i_o\})}, \{full_i\}_{i \in D \cap L}$ and then adds $Partial_{i_0}(V_2)$ to form $Polys = \{Partial_i\}_{i \in D \setminus L}, \{Full_i\}_{i \in D \cap L}$, runs $B(Polys, \theta_2(V_2))$ and returns its output.

Notice that so formed variable $Polys$ is a uniformly distributed envelope $Polys_{V^*}(V')$ of V' constructed as a function of V_2 by equation $V' = \{x_j + v_j^*\}_{j \in K_1}$ where $V_2 =$

$\{x_j\}_{j\in K_1}$. Consequently, if we define (for every q) $\lambda_2(\cdot)$ as a knowledge function on $Z_q^{(k-l)}$ by $\lambda_2(\{x_j\}_{j\in K_1}) = \lambda(\{x_j + v_j^*\}_{j\in K_1})$, then we have that $\theta_2(V_2) = \theta(V')$ and $\lambda_2(V_2) = \lambda(V')$.

Since V_2 is uniformly distributed in $Z_q^{(k-l)}$ then so is V', and therefore, V' has the same distribution as V and $Polys$ has the same distribution as $Polys_{V*}(V)$. Consequently (for every $q \in Q$)

$$Prob[C(Partial_{i_0}(V_2), \theta_2(V_2)) = \lambda_2(V_2)] = Prob[A(Polys_{V^*}(V), \theta(V)) = \lambda(V)] \tag{9.12}$$

From equations 9.11 and 9.12 it follows that for every $\mathcal{BPP}$ machine A'

$$\begin{aligned}\exists_c \forall_{h_0} \exists_{q \text{ s.t. } |q| \geq h_0} Prob[C(Partial_{i_0}(V_2), \theta_2(V_2)) = \lambda_2(V_2)] \;&>\\ Prob[A'(\theta_2(V_2)) = \lambda_2(V_2)] \;&+\; \frac{1}{|q|^c}\end{aligned}$$

which contradicts the fact that $Partial_{i_0}(\cdot)$ is a semantically secure scheme, as proven in 9.4.7. This ends the proof of the lemma. ∎

Now we prove a semantically secure equivalent of theorem 9.4.2:

Theorem 9.4.9 *The view $View^{(t)}$ of the adversary during the update between time period $t-1$ and t, is a semantically secure envelope of the secret x for every t.*

The information $View^{(t)}$ available to the adversary is equal to

$$View^{(t)} = (Shares^{(t-1)}, Shares^{(t)}, Ims^{(t-1)}, Ims^{(t)}, \{Full_i\}_{i\in D\cap L}, \{Partial_i\}_{i\in D\setminus L}) \tag{9.13}$$

where:

$$\begin{aligned}
Shares^{(t-1)} &= \{x_i^{(t-1)}, z_i^{(t-1)}\}_{i\in L\cup K_1}\\
Shares^{(t)} &= \{x_i^{(t)}, z_i^{(t)}\}_{i\in L\cup K_2}\\
Ims^{(t-1)} &= \{y_i^{(t-1)}\}_{i\in \mathcal{A}}\\
Ims^{(t)} &= \{y_i^{(t)}\}_{i\in \mathcal{A}}\\
Full_i &= \delta_i(\cdot), \gamma_i(\cdot)\\
Partial_i &= \{\delta_i(j), \gamma_i(j)\}_{j\in L}, \{g^{\delta_{im}} h^{\gamma_{im}}\}_{m\in\{1\ldots k\}}, \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j\notin L}
\end{aligned}$$

Proof Idea: In the case of black-box encryption, the proof of theorem 9.4.2 followed from lemma 9.4.5. Unfortunately, in the case of semantically secure encryption, reduction of theorem 9.4.9 to lemma 9.4.8 does not work. Ideally, we would like to show that the assumption that $View^{(t)}(\cdot)$ ($View^{(t)}$ treated as an envelope scheme on x) is not a semantically secure envelope scheme contradicts lemma 9.4.8. However, the two standard reduction methods used in such proofs failed here:

- We cannot build a $\mathcal{BPP}$ machine B that would distinguish between random envelopes $Polys(\cdot)$ (defined like in lemma 9.4.8) of some two sets V_1 and V_2, using a $\mathcal{BPP}$ machine A that distinguishes between random envelopes $View^{(t)}(\cdot)$ of some two elements x_1 and x_2. It is because from inputs V_1, V_2 and P to machine B ($P = Polys(V_1)$ or $P = Polys(V_2)$) we cannot build a V such that $V = View^{(t)}(x_1)$ if $P = Polys(V_1)$ and $V = View^{(t)}(x_2)$ if $P = Polys(V_2)$. Similarly if instead of $Polys(\cdot)$ we use some more restricted scheme that, say, produces envelops with known values $\{\delta(i)\}_{i\in L}$, or, say, envelops only one value $v = \delta(k)$ instead of the set $V = \{\delta(i)\}_{i\in K_1}$.

- For the same reason, we cannot find a $\mathcal{BPP}$ machine B that would compute $\lambda_2(V)$ from some $\theta_2(V)$ and a random $Polys(V)$ with non-negligibly higher probability that any machine B' having $\theta_2(V)$ as the only input, if we are to use a machine A that computes $\lambda(x)$ from $View^{(t)}(x)$ and some $\theta(x)$ with non-negligibly higher probability than any $\mathcal{BPP}$ machine A' having $\theta(x)$ as the only input. Again, from inputs of machine B we cannot build a valid envelope $View(x)$ for any x.

Instead, our proof below uses theorem 9.4.2 and a hybrid method of cascading random variables, to show that if $View^{(t)}(\cdot)$ is polynomially indistinguishable, then so is the encryption scheme $ENC(\cdot)$.

Proof of theorem 9.4.9: In theorem 9.4.2 we show $View^{(t)}(\cdot)$ defined by equation 9.1 is an information theoretic secure envelope of scheme. Notice that the difference between the two definitions of $View$ is not only the addition of encrypted shares of partial update polynomials but also the fact that the partial information about update polynomial is now available for $i \in D \setminus L$ instead of $i \notin L$ and the full information for $i \in D \cap L$ instead of $i \in L$. However, the same proof as in 9.4.2 goes through even if the adversary has a partial information about only polynomial $\delta_{i_0}(\cdot), \gamma_{i_0}(\cdot)$ for some $i_0 \in D \cap L$, and a full information $\{Full_i\}_{i\in D\setminus\{i_0\}}$ about all the other polynomials (we used this idea to prove lemmas 9.4.5 and 9.4.8). Therefore, the envelope scheme

$$\begin{aligned} View_2(\cdot) \;=\; & Shares^{(t-1)}, Shares^{(t)}, Ims^{(t-1)}, Ims^{(t)}, \{Full_i\}_{i\in D\setminus\{i_0\}}, \\ & \{\delta_{i_0}(j), \gamma_{i_0}(j)\}_{j\in L}, \{g^{\delta_{i_0m}} h^{\gamma_{i_0m}}\}_{m\in\{1\ldots k\}} \end{aligned}$$

is also an information theoretic secure envelope scheme. Consequently scheme

$$\begin{aligned} View_3(\cdot) \;=\; & Shares^{(t-1)}, Shares^{(t)}, Ims^{(t-1)}, Ims^{(t)}, \{Full_i\}_{i\in D\setminus\{i_0\}}, \\ & \{\delta_{i_0}(j), \gamma_{i_0}(j)\}_{j\in L}, \{g^{\delta_{i_0m}} h^{\gamma_{i_0m}}\}_{m\in\{1\ldots k\}}, \{ENC_j(0,0)\}_{j\notin L} \end{aligned}$$

is information theoretic secure (we used the same idea in the proof of lemma 9.4.7). Now, if we assume that scheme $View^{(t)}(\cdot)$ as defined in equation 9.13 (let us call it $View_1(\cdot)$) is not semantically secure, then the following scheme

$$\begin{aligned} View_4(\cdot) \;=\; & Shares^{(t-1)}, Shares^{(t)}, Ims^{(t-1)}, Ims^{(t)}, \{Full_i\}_{i\in D\setminus\{i_0\}}, \\ & \{\delta_{i_0}(j), \gamma_{i_0}(j)\}_{j\in L}, \{g^{\delta_{i_0m}} h^{\gamma_{i_0m}}\}_{m\in\{1\ldots k\}}, \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j\notin L} \end{aligned}$$

cannot be semantically secure either, because for every x, one can compute $View_1(x)$

from $View_4(x)$, i.e. $View_4(x)$ contains strictly more information than $View_1(x)$.

Let A be a $\mathcal{BPP}$ distinguisher of $View_4(\cdot)$, i.e.

$$\exists_c \forall_{h_o} \exists_{q \geq 2^{h_0}} |Prob[A(x_1, x_2, View_4(x_1)) = 1] - Prob[A(x_1, x_2, View_4(x_2)) = 1]| > \frac{1}{|q|^c}$$

where the probability is taken over the random choices made by $View_4(\cdot)$.

Following the reasoning used in proof of lemma 9.4.7 we notice that variables $View_3(x_1)$ and $View_3(x_2)$ have the same probability distributions, and hence we deduce that for either for $x_0 = x_1$ or for $x_0 = x_2$ the following must be true

$$|Prob[A(x_1, x_2, View_4(x_0)) = 1] - Prob[A(x_1, x_2, View_3(x_0)) = 1]| > \frac{1}{2h^c} \quad (9.14)$$

Now, just like in the proof of lemma 9.4.7 we can use the hybrid method to create a sequence of $n - l + 1$ envelopes of x_0, $\{V^{(s)}(x_0)\}_{s \in \{0,\ldots,n-l\}}$ and corresponding probabilities $\{p^{(s)}\}_{s \in \{0,\ldots,n-l\}}$:

$$\begin{aligned} V^{(s)}(x_0) &= Shares^{(t-1)}, Shares^{(t)}, Ims^{(t-1)}, Ims^{(t)}, \{Full_i\}_{i \in D \setminus \{i_0\}}, \\ &\quad \{\delta_{i_0}(j), \gamma_{i_0}(j)\}_{j \in L}, \{g^{\delta_{i_0 m}} h^{\gamma_{i_0 m}}\}_{m \in \{1 \ldots k\}}, \\ &\quad \{ENC_j(\delta_i(j), \gamma_i(j))\}_{j \in \{l+1,\ldots,n-s\}}, \{ENC_j(0,0)\}_{j \in \{n-s+1,\ldots,n\}} \\ p^{(s)} &= Prob[A(x_1, x_2, V^{(s)}(x_0)) = 1] \end{aligned}$$

We follow the proof exactly like in the proof of lemma 9.4.7: From triangle inequality there must be an element s_0 for which $|p^{(s_0+1)} - p^{(s_0)}| > \frac{1}{2(n-l)h^c}$. Next we show that there must be a pair v^* such that $|Prob[A(x_1, x_2, E_1) = 1] - Prob[A(x_1, x_2, E_2) = 1]| > \frac{1}{2(n-l)h^c}$, where E_1 and E_2 are random variables that have distributions of the envelope schemes $V^{(s_0)}(\cdot)$ and $V^{(s_0+1)}(\cdot)$ respectively, where these envelope schemes are restricted to picking polynomials $\delta_{i_0}(\cdot), \gamma_{i_0}(\cdot)$ for which $(\delta_{i_0}(n - s_0), \gamma_{i_0}(n - s_0)) = v^*$. Finally, we build a $\mathcal{BPP}$ machine A^* which distinguishes between random encryptions $ENC_{n-s_0}(0,0)$ of pair of zeroes and random encryptions $ENC_{n-s_0}(v^*)$ of v^*: It takes as input a number e which is a result of a random encryption $ENC_{n-s_0}(\cdot)$ of either $(0,0)$ or v^*, finds $2 * |D| + 4$ random k-degree polynomials $f^{(t)}(\cdot), f^{(t+1)}(\cdot), r^{(t)}(\cdot), r^{(t+1)}(\cdot), \{\delta_i(\cdot), \gamma_i(\cdot)\}_{i \in D}$ in Z_q such that $(\delta_{i_0}(n - s_0), \gamma_{i_0}(n - s_0)) = v^*$, $\delta_i(0) = \gamma_i(0) = 0$ for all i, $f^{(t)}(0) = f^{(t+1)}(0) = x_0$, $f^{(t+1)}(\cdot) - f^{(t)}(\cdot) = \sum_{i \in D} \delta_i(\cdot)$ and $g^{(t+1)}(\cdot) - g^{(t)}(\cdot) = \sum_{i \in D} \gamma_i(\cdot)$. Then A^* creates $P = P^{(s_0)}(x_0)$ using these polynomials but substitutes e for a random encryption $ENC_{n-s_0}(\delta_{i_0}(n - s_0), \gamma_{i_0}(n - s_0))$, runs $A(x_1, x_2, P)$ and returns the output. It follows that

$$|Prob[A^*(ENC_{n-s_0}(0,0)) = 1] - Prob[A^*(ENC_{n-s_0}(v^*)) = 1]| > \frac{1}{2(n-l)h^c}$$

which means that A^* breaks encryption scheme $ENC(\cdot)$, which is a contradiction. Consequently, the claim of our theorem follows. ■

Now we present the semantic security equivalent of theorem 9.4.1

Theorem 9.4.10 *The view* $\{View^{(t)}\}_{t\in\{0,\ldots,t_0\}}$ *of consecutive runs of the share renewal protocol (*$View^{(t)}$ *is described in equation 9.13) is a semantically secure envelope of the secret* x.

Proof: This proof will be very similar to the proof above and so we will only outline its steps here. We adopt the notation from the proofs of theorems 9.4.1 and 9.4.9. Let us denote the full history available to the adversary as an envelope scheme $View_1(\cdot)$ on secret x, i.e.

$$\begin{aligned} Views_1(\cdot) &= Shares(0), Ims^{(0)}, \{View^{(t)}\}_{t\in\{0,\ldots,t_0\}} \\ View^{(t)} &= Shares^{(t)}, Ims^{(t)}, \{Full_i^{(t)}\}_{i\in D^{(t)}\cap L^{(t)}}, \{Partial_i^{(t)}\}_{i\in D^{(t)}\setminus L^{(t)}}, \\ Partial_i^{(t)} &= \{\delta_i^{(t)}(j), \gamma_i^{(t)}(j)\}_{j\in L^{(t)}}, \{g^{\delta_{im}^{(t)}} h^{\gamma_{im}^{(t)}}\}_{m\in\{1\ldots k\}}, \{ENC_j(\delta_i^{(t)}(j), \gamma_i^{(t)}(j))\}_{j\notin L^{(t)}} \end{aligned}$$

Let $Views_2(\cdot)$ be an envelope scheme defined like in equation 9.6, except that, just like in the proof above, in every update phase t the adversary has a full information about update polynomials except of one update polynomial $\delta_{i_t}^{(t)}(\cdot), \gamma_{i_t}^{(t)}(\cdot)$ about which only partial information is available:

$$\begin{aligned} Views_2(\cdot) &= Shares(0), Ims^{(0)}, \{View^{(t)}\}_{t\in\{0,\ldots,t_0\}} \\ View^{(t)} &= Shares^{(t)}, Ims^{(t)}, \{Full_i^{(t)}\}_{i\in D^{(t)}\setminus\{i_t\}}, \\ & \quad \{\delta_{i_t}^{(t)}(j), \gamma_{i_t}^{(t)}(j)\}_{j\in L}, \{g^{\delta_{i_t m}^{(t)}} h^{\gamma_{i_t m}^{(t)}}\}_{m\in\{1\ldots k\}} \end{aligned}$$

The same proof as in theorem 9.4.1 goes through for the above envelope, and hence it is information theoretically secure. Similarly to the proof of theorem 9.4.9, scheme $View_3(\cdot)$ which is just like $View_2(\cdot)$ except that we add $(n - l^{(t)})$ encryptions of a pair of zeroes to each $View^{(t)}$, is also information theoretically secure. We then build $Views_4(\cdot)$ which is just like $Views_1(\cdot)$ except that every $View^{(t)}$ has a partial information only about $\delta_{i_t}^{(t)}(\cdot), \gamma_{i_t}^{(t)}(\cdot)$ and full information about all the other update polynomials. Just like in the proof of theorem 9.4.9, if we assume that $View_1(\cdot)$ is not a semantically secure envelope scheme, then neither is $View_4(\cdot)$. By triangle inequality we show that there is a $\mathcal{BPP}$ machine A such that

$$\exists_c \forall_{h_0} \exists_q \text{ s.t. }_{|q|\geq h_0} \exists_{x_0\in Z_q} |Prob[A(Views_4(x_0)) = 1] - Prob[A(Views_3(x_0)) = 1]| > \frac{1}{2|q|^c} \tag{9.15}$$

By the hybrid method we create a sequence of variables $\{V_s^{(t)}\}_{s\in\{0,\ldots,n-l^{(t)}\},t\in\{0,\ldots,t_0\}}$ and corresponding probabilities $\{p_s^{(t)}\}_{s\in\{0,\ldots,n-l^{(t)}\},t\in\{0,\ldots,t_0\}}$, $p_s^{(t)} = Prob[A(V_s^{(t)}) = 1]$ where $V_0^{(0)}$ has the same distribution as $View_4(x_0)$, $V_{n-l^{(t)}}^{(t_0)}$ has the same distribution as $View_3(x_0)$ and the other random variables $V_s^{(t)}$ fill the spectrum between these two distributions:

$$V_s^{(\tau)}(\cdot) \;=\; \{Shares^{(t)}, Ims^{(t)}\}_{t\in\{0,\ldots,t_0\}},$$

$$\begin{aligned}
& \{Full_i^{(t)}\}_{i\in D^{(t)}\setminus\{i_t\},t\in\{1,\dots,t_0\}}, \\
& \{PartialZero^{(t)}\}_{t\in\{1,\dots,\tau-1\}}, \\
& PartialHalf^{(\tau)}, \\
& \{PartialFull^{(t)}\}_{t\in\{\tau+1,\dots,t_0\}} \\
PartialZero^{(t)} &= \{\delta_{it}^{(t)}(j),\gamma_{it}^{(t)}(j)\}_{j\in L^{(t)}},\{g^{\delta_{itm}^{(t)}}h^{\gamma_{itm}^{(t)}}\}_{m\in\{1\dots k\}}, \\
& \{ENC_j(0,0)\}_{j\in\{l^{(t)},\dots,n\}} \\
PartialHalf^{(t)} &= \{\delta_{it}^{(t)}(j),\gamma_{it}^{(t)}(j)\}_{j\in L^{(t)}},\{g^{\delta_{itm}^{(t)}}h^{\gamma_{itm}^{(t)}}\}_{m\in\{1\dots k\}}, \\
& \{ENC_j(\delta_{it}^{(t)}(j),\gamma_{it}^{(t)}(j))\}_{j\in\{l^{(t)}+1,\dots,n-s\}}, \\
& \{ENC_j(0,0)\}_{j\in\{n-s+1,\dots,n\}} \\
PartialFull^{(t)} &= \{\delta_{it}^{(t)}(j),\gamma_{it}^{(t)}(j)\}_{j\in L^{(t)}},\{g^{\delta_{itm}^{(t)}}h^{\gamma_{itm}^{(t)}}\}_{m\in\{1\dots k\}}, \\
& \{ENC_j(\delta_{it}^{(\tau)}(j),\gamma_{it}^{(\tau)}(j))\}_{j\in\{l^{(t)}+1,\dots,n\}}
\end{aligned}$$

In other words, for every t, $V_{s+1}^{(t)}$ looks just like $V_s^{(t)}$ except that it has one more random encryption of a pair of zeroes instead of a share of the update polynomial $\delta_{it}^{(t)}(\cdot),\gamma_{it}^{(t)}(\cdot)$

There are $1+\sum_{t\in\{1,\dots,t_0\}}(n-l^{(t)})$ of these variables. In the worst case, $l^{(t)}=0$ for all t and hence, there would be t_0n+1 of variables $V_s^{(t)}$. Using the same arguments as before we argue that there must be a pair s^*,t^* such that $|p_{s^*+1}^{(t^*)}-p_{s^*}^{(t^*)}|>\frac{1}{2nt_0|q|^c}$ (it could be also that $s^*=n-l^{(t^*)}$ and $|p_0^{(t^*+1)}-p_{s^*}^{(t^*)}|>\frac{1}{2nt_0|q|^c}$). Just like in the proofs of theorem 9.4.9 and lemmas 9.4.7 and lm-ss2, we argue that there must be a particular pair v^* such that if envelopes $V_{s^*}^{(t^*)}(x_0)$ and $V_{s^*+1}^{(t^*)}(x_0)$ are restricted by equation $(\delta_{i_{t^*}}^{(t^*)}(n-s^*),\gamma_{i_{t^*}}^{(t^*)}(n-s^*))=v^*$ then A distinguishes these envelopes with probability greater than $\frac{1}{2nt_0|q|^c}$. Now we create A^* which takes as input a number e which can be equal to $ENC_{n-s^*}(0,0)$ or $ENC_{n-s^*}(v^*)$, creates an envelope E equal to $V_{s^*+1}^{(t^*)}(x_0)$ except that it substitutes e for a random encryption $ENC_{n-s^*}(0,0)$, runs $A(E)$ and returns its output. It follows that A^* breaks the encryption scheme $ENC_{n-s^*}(\cdot)$ and so the theorem is proven. ■

Optimality of the Reductions

In a single run of the share renewal protocol (theorem 9.4.9), our result that the strength of the secrecy protection of our scheme is $2(n-l)$ times weaker than the strength of encryption, seems only a factor of 2 worse than an optimal result: The adversary knows a sequence of $n-l$ encrypted shares $share_i$ where knowledge of any of them implies knowledge of the secret x. If for every $share_i$, encryption $ENC(share_i)$ gives out $\frac{1}{|q|^c}$ amount of knowledge about $share_i$ and if $share_i$ gives out all knowledge about x then it seems intuitive that sequence $\{ENC(share_i)\}_{i\in\{l+1,\dots,n\}}$ can give out $\frac{1}{(n-l)|q|^c}$ amount of knowledge about x.

Similarly, in the case of a sequence of share renewals (theorem 9.4.10), the adversary knows $\sum_{t\in\{1,\ldots,t_0\}}(n-l^{(t)})$ encrypted shares, where knowledge about any of them implies knowledge of the secret x. It makes sense then that the strength of secrecy protection of our scheme is $\sum_{t\in\{1,\ldots,t_0\}}(n-l^{(t)})$ times weaker than the strength of the encryption scheme itself. In the worse case, $l^{(t)}=0$ for all t and then the above sums up to $t_0 * n$. Our result is factor of 2 weaker than this intuitively optimal bound.

However, these bounds might not be optimal at all. In the above argument, we assume for simplicity that all polynomials in D except of one are fully revealed to the adversary, and hence, knowledge of every share of the total update polynomial $\delta(\cdot),\gamma(\cdot)$ depends on the knowledge of only one encrypted share of a partial update polynomial $\delta_{i_0}(\cdot),\gamma_{i_0}(\cdot)$ (other partial update polynomials are fully known to the adversary). This is not really the case: the total update polynomial depends on $n-l$ polynomials $\{\delta_i(\cdot),\gamma_i(\cdot)\}_{i\in\{l+1,\ldots,n\}}$ about which the adversary knows only partial information. Therefore, for every $share_i$, the adversary does not know $ENC(share_i)$ but a sequence $E=\{ENC(share_{ij})\}_{j\in\{l+1,\ldots,n\}}$ where $share_i=\sum_{j\in\{l+1,\ldots,n\}} share_{ij}+\sum_{j\in D\cap L} share_{ij} \ (mod\ q)$. In our proofs, we assumed that since $ENC(share_{ij})$ gives out at most $\frac{1}{|q|^c}$ information about $share_{ij}$ then set E cannot give out more information about $share_i$ than $\frac{1}{|q|^c}$. This, however, is only a higher bound. There could in principle exist an encryption scheme $ENC(\cdot)$ such that the set of encryptions of elements that sum up to $share_i$ would gives out much less knowledge about $share_i$ than a single encryption $ENC(share_i)$: If $ENC(a)$ gives out $\frac{1}{|q|^c}$ amount of information about $a\in Z_q$ and $Enc(b)$ gives out $\frac{1}{|q|^c}$ amount of information about $b\in Z_q$, then $ENC(a),ENC(b)$ can give out only $\frac{1}{|q|^c}*\frac{1}{|q|^c}=\frac{1}{|q|^{2c}}$ amount of information about $a+b\ (mod\ q)$. Consequently, E would give out $\frac{1}{|q|^{(n-l)c}}$ amount of knowledge about $share_i$, and if there was a public key encryption function for which the above was true, our scheme would give out only $\frac{1}{(n-l)|q|^{(n-l)c}}$ amount of information about the secret x.

The Average Case Strength of the Secrecy Protection

Theorems 9.4.9 and 9.4.10 deal with the worst case scenarios when the adversary compromises the maximum number of servers our scheme allows. If we take not the worst case, the strength of the secrecy protection that our scheme offers varies with the number of servers that are compromised. For example, in the case of a single execution of the share renewal protocol between time period t and $t+1$, if $|K_1\cup L\cup K_2|\leq k$ then even if the adversary breaks n^2 encryptions $\{ENC_i(\delta_j(i),\gamma_j(i))\}_{i,j\in\{1\ldots n\}}$ (a subset of $(n-|L|)k$ encryptions $\{ENC_i(\delta_j(i),\gamma_j(i))\}_{i\in S,j\notin L}$, $|S|=k$, would give the same knowledge) and learn the total update polynomial $\delta(\cdot),\gamma(\cdot)$, the secret x would be hidden from the adversary with actually an information theoretical secrecy: The adversary could compute $\{x_i,z_i\}_{i\in K_1\cup L\cup K_2}$ for both time period t and $t+1$, but these shares will be consistent with any value of x.

9.4.3 Secrecy Protection in the Share Recovery Protocol

The secrecy protection results for the share renewal protocol (theorem 9.4.9 and 9.4.10) can be extended to the share recovery protocol. Steps 1 to 4 of the share recovery protocol (as described in section 5.3 with Feldman's VSS or in section 7.3 with Pedersen's VSS) essentially copy the share renewal protocol: the servers re-randomize the secret sharing polynomial $f(\cdot)$ (or the pair of polynomials $f(\cdot), b(\cdot)$ in the case of Pedersen's VSS). The difference is that in the share renewal protocol they re-randomize it with a constraint that $(f'(0), b'(0)) = (f(0), b(0)$, and in the share recovery protocol they re-randomize it with a constraint that $(f'(r), b(r)) = (f(r)$ where r is the index of the server whose share is to be recovered.

The more important difference between the two protocols, however, is the worst-case number of shares contained in the adversary's view $RecView_r^{(\mathcal{B})}$, where P_r is the server whose share is being recovered and

- $\mathcal{B}$ is the set of all servers who need share recovery: $\mathcal{B}_2$, are those that are compromised during the update phase (i.e. the adversary moved back to them just after they were rebooted) and $\mathcal{B}_1$, are those who were compromised only in the time period proper and are not compromised during the update.
- K, $|K| = k_1$, is the set of servers compromised during the time period preceding the update, but not during the update. Clearly, $\mathcal{B}_1 \subset K_1$.
- L, $|L| = l - k_1$, is the set of servers compromised by the adversary during share recovery (and also, we can assume, throughout the preceding time period). Clearly, $\mathcal{B}_2 \subset L$ and $\mathcal{K}_1 \cap L = \{\}$.
- $\mathcal{D}'$, is the set of servers whose rerandomization polynomials are well formed and distributed, and hence included in the total rerandomization polynomial. Clearly, $\mathcal{D}' \cap B = \{\}$ and $(\mathcal{A} \setminus L) \subset \mathcal{D}'$.

Clearly worst case is when $\mathcal{B}_1 = K$. For simplicity let $K = \mathcal{B}_1 = \{1, \ldots, k_1\}$, $\mathcal{B}_2 = \{k_1, \ldots, l'\}$, $L = \{k_1, \ldots, l\}$. The view of the adversary during the share recovery protocol is as follows:

$$RecView_r^{(\mathcal{B})} = (Shares, Ims, Shares', \{Full_i\}_{i\in\mathcal{D}'\cap L}, \{Partial_i\}_{i\in\mathcal{D}'\setminus L}) \tag{9.16}$$

where:

$$\begin{aligned}
Shares &= \{x_i, z_i\}_{i\in\{1,\ldots,l\}} \\
Ims &= \{y_i\}_{i\in\{1\ldots n\}} \\
Shares' &= \{x'_i, z'_i\}_{i\in\{l'+1,\ldots,l\}} \text{ if } P_r \in \mathcal{B}_1 \\
Shares' &= \{x'_i, z'_i\}_{i\in\{1\ldots n\}} \text{ if } P_r \in \mathcal{B}_2 \\
Full_i &= \delta_i(\cdot), \gamma_i(\cdot) \\
Partial_i &= \{\delta_i(j), \gamma_i(j)\}_{j\in\{l'+1,\ldots,l\}}, \{g^{\delta_{im}} h^{\gamma_{im}}\}_{m\in\{1\ldots k\}},
\end{aligned}$$

$$\{ENC_j(\delta_i(j), \gamma_i(j))\}_{j \in \{l+1,\ldots,n\}}$$

We state the following theorem:

Theorem 9.4.11 *The view $RecView_r^{(\mathcal{B})}$ of the adversary during a single execution of the share recovery protocol for server $P_r, r \in \mathcal{B}$, is a semantically secure envelope of the secret x.*

Proof: We will only sketch the proof of this theorem, because it is very similar to the proofs of the secrecy protection of the share renewal protocol.

Just like in the share renewal protocols, we argue that the view of the adversary can be treated as an envelope scheme $RecView_r^{(\mathcal{B})}(\cdot)$ of the secret x. Notice that adversary has the most information when $l = k$, $l' = 1$ and $P_r \in \mathcal{B}_2$, which means that $K = \mathcal{B}_1 = \{\}$ and $\mathcal{B} = \mathcal{B}_2 = \{1\}$.

Since $1 \leq l' \leq l$, set $\{\delta_i(j), \gamma_i(j)\}_{j \in \{l'+1,\ldots,l\}}, \{g^{\delta_{im}} h^{\gamma_{im}}\}_{m \in \{1\ldots k\}}$ looks like a subset of a view of the adversary during Pedersen's VSS. This means that if there were no encryptions of shares in $Partial_i$ then $Partial_i$ would be an information theoretical secure envelope of $\delta_i(0)$. Similarly to lemma 9.4.5 we show that in this case, $\{Full_i\}_{i \in L \cap \mathcal{D}'}, \{Partial_i\}_{i \in \mathcal{D}' \setminus L}$ would be an information theoretical secure envelope of $\delta(0)$ where $\delta(\cdot) = \sum_{\mathcal{D}'} \delta_i(\cdot)$. For every $x_0 \in Z_q$, setting $f(0) = x_0$ sets $f(\cdot), b(\cdot)$. Since $f'(\cdot), b'(\cdot)$ is known to the adversary from $Shares'$, this fixes $\delta(\cdot), \gamma(\cdot)$ and hence $\delta(0)$. Similarly to the proof of theorem 9.4.2 we show that for every x_0, so formed polynomials $\delta(\cdot), \gamma(\cdot)$ are consistent with the information in $\{Full_i\}_{i \in L \cap \mathcal{D}'}, \{Partial_i\}_{i \in \mathcal{D}' \setminus L}$. This shows that if $Partial_i$ had no encryptions $\{ENC_j(\delta_i(j), \gamma_i(j))\}_{j \in \{l+1,\ldots,n\}}$ for every i, then $RecView_r^{(\mathcal{B})}(\cdot)$ would be an information theoretically secure envelope of x.

When we put back encryptions to $RecView_r^{(\mathcal{B})}$, we can prove that $RecView_r^{(\mathcal{B})}(\cdot)$ is a semantically secure envelope of x. The proof is analogous to the proof of the theorem 9.4.9: by a hybrid method we reduce the semantic security of $RecView_r^{(\mathcal{B})}(\cdot)$ to the semantic security of the encryption scheme $ENC(\cdot)$: In the worst case, $RecView_r^{(\mathcal{B})}(\cdot)$ protects the secrecy of x with at most $2(n-k)$ times weaker strength then the strength of the secrecy protection offered by the encryption scheme $ENC(\cdot)$ (learning any $\delta(i), \gamma(i)$, $i \in \{l+1,\ldots,n\}$ implies learning the secret x). ∎

Just like we can extend theorem 9.4.9 about a single execution of the share renewal protocol into theorem 9.4.10 concerned with a sequence of its executions, we extend the above theorem:

Theorem 9.4.12 *The view of the adversary of a series of executions of the share recovery protocol $\{RecView_r^{(\mathcal{B})}\}_{r \in \mathcal{B}}$ is a semantically secure envelope of the secret x.*

Proof: We will only sketch this proof. Analogously to the proof of theorem 9.4.10, we need an extension of the information theoretical version of theorem 9.4.11 for the case of multiple executions of the share recovery protocol. If we take away the encrypted shares from each $RecView_r^{(\mathcal{B})}$, setting x_0 sets $f(\cdot), g(\cdot)$. Since every execution of share recovery gives to adversary different sets $Shares', \{Full_i\}_{i \in L \cap \mathcal{D}'}$ and $\{Partial_i\}_{i \in \mathcal{D}' \setminus L}$ (let's denote them as $Shares^{(r)}, \{Full_i^{(r)}\}_{i \in L \cap \mathcal{D}'}, \{Partial_i^{(r)}\}_{i \in \mathcal{D}' \setminus L}$ for each $P_r \in \mathcal{B}$), each $Shares^{(r)}$ together with $f(\cdot), g(\cdot)$ defines $\delta^{(r)}(\cdot), \gamma^{(r)}(\cdot)$ which are consistent with the information in $\{Full_i^{(r)}\}_{i \in L \cap \mathcal{D}'}, \{Partial_i^{(r)}\}_{i \in \mathcal{D}' \setminus L}$.

Then we prove that putting the encrypted shares back reduces the secrecy protection to semantic security. The adversary's view is clearly best if $l = k$ and $\mathcal{B}_1 = K = \{\}$. For every P_r, $r \in \mathcal{B}_2$ so the view of the adversary includes all values $\{x_i', z_i'\}_{i \in \{1 \ldots n\}}$ for that rerandomized polynomial $f^{(r)}(\cdot), b^{(r)}(\cdot)$. This means that to learn the secret the adversary needs to know one additional value among $\{\delta^{(r)}(j), \gamma^{(r)}(j)\}_{j \in \{l+1, \ldots, n\}}$. This reduction gives us the following lower bound of the strength of the secrecy of the recovery protocol: The secrecy protection of the multiple executions of the share recovery protocol is at most $2l'(n-k)$ worse then the secrecy protection of the encryption scheme. In the worst case $l' = l = k$ and the secrecy protection is at most $2k(n-k)$ worse then $ENC(\cdot)$. ■

9.4.4 Secrecy Protection during the Lifetime of the Proactive Algorithm

We can now connect theorem 9.4.10 in section 9.4.2 (i.e. the secrecy protection of repeated execution of the share renewal protocol) and theorem 9.4.12 in section 9.4.3 (i.e. the secrecy protection of repeated execution of the share recovery protocol) to prove the secrecy characteristics of the whole proactive algorithm using Pedersen's VSS, i.e. to prove the secrecy part of theorem 3.2.0. We use the same notation as in the previous section.

Proof of the secrecy part of theorem 3.2.0: The maximal view $Info$ of the computation that the adversary can have during the lifetime of the proactive secret sharing algorithm is as follows:

$$Info = (Shares^{(0)}, Ims^{(0)}, \{View^{(t)}, \{RecView_r^{(\mathcal{B}^{(t)})}\}_{r \in \mathcal{B}^{(t)}}\}_{t \in \{1, \ldots, t_0\}}) \tag{9.17}$$

where $View^{(t)}$ is described in equation 9.13 and $RecView_r^{(\mathcal{B}^{(t)})}$ is described in equation 9.16. To make this formula clear, note that in each update phase there are $|\mathcal{B}^{(t)}|$ executions of the share recovery protocol followed by a single execution of the share renewal protocol.

We will only sketch this proof because it is similar to the proofs we wrote for theorems 9.4.10 and 9.4.12. As in those proofs, we first strip $Info$ of encryptions of shares of update polynomials in each $View^{(t)}$ or rerandomization polynomials in each $View_r^{(\mathcal{B}^{(t)})}$. We show that this stripped view is an information theoretically secure envelope of x: Let $x = x_0 \in Z_q$ for any x_0. For each time period t separately, x_0 together with $Shares^{(t)}$ defines $f^{(t)}(\cdot), g^{(t)}(\cdot)$. For each of these polynomials we can make the same argument as in the proof of theorem 9.4.12 to show that they are consistent with $\{RecView_r^{(\mathcal{B}^{(t)})}\}_{r \in \mathcal{B}^{(t)}}$. Also, for each pair $f^{(t-1)}(\cdot), g^{(t-1)}(\cdot)$ and $f^{(t)}(\cdot), g^{(t)}(\cdot)$ we can make the same argument as in the proof of theorem 9.4.10 to show that they are consistent with $View^{(t)}$.

Now we put the encryptions back and we use the hybrid method to show that if we assume that $Info(\cdot)$ is not a semantically secure envelope scheme then neither is $ENC(\cdot)$. The exact worst case coefficient of this reduction does not follow from the worst case coefficients of reductions we made in the proofs of theorems 9.4.10 and 9.4.12. It is because the worst case in theorem 9.4.10 is when the adversary is maximally mobile, i.e. when the sets of servers compromised during each update phase $L^{(t)}$ are all empty and the sets of servers compromised only during each proper time period $K^{(t)}$ are maximally big, i.e. $|K^{(t)}| = k$ and $K^{(t-1)} \cap K^{(t)} = \{\}$, for all t. In contrast, the worst case in theorem 9.4.12 is when the adversary does not move at all and stays in one set of servers throughout the lifetime of the algorithm, i.e. when for all t, $K^{(t)} = \{\}$ and $L^{(t)} = L$ where $|L| = k$.

However, instead of the detailed analysis we argue that the information contained in $Info(\cdot)$ cannot be worse then if during each time period and phase update there occurred the worst case from both the perspective of the secrecy of share renewal protocol and the secrecy of share recovery protocol. In other words, the secrecy of our scheme is no worse then if the adversary sees $t_0 k(n-k)$ (from executions of share recovery) plus $t_0 n$ (from executions of share renewal) encrypted values such that if she decrypts any of them she learns the secret. This makes the strength of the secrecy protection of $Info(\cdot)$ at most $2t_0(nk+n-k^2)$ less secure then the strength of $ENC(\cdot)$.

■

9.5 Protection of Secrecy in the version with Feldman's VSS

In this section we will prove the secrecy characteristics of our scheme with Feldman's VSS, i.e. we will prove the secrecy part of theorem 3.1.0. We will first present the proof technique we will use by proving an equivalent of theorem 9.4.9 (semantic security of a single execution of the share renewal protocol using Pedersen's VSS) for the proactive share renewal protocol using Feldman's VSS (theorem 9.5.1 below). The proof of the secrecy part of the main theorem 3.1.0 will be very similar.

Theorem 9.5.1 *The view $FelView^{(t)}$ of the adversary of a single update phase between time period $t-1$ and t in the share renewal algorithm using Feldman's VSS is an envelope of secret x semantically secure relative to the knowledge of g^x $(mod\ q)$.*

Proof: We will prove that if envelope $FelView^{(t)}(x)$ is not semantically secure relative to knowledge of g^x $(mod\ p)$ then the envelope $View^{(t)}(x)$ (the view of the adversary of a single run of the share renewal using Pedersen's VSS, as in theorem 9.4.9) is not semantically secure.

Notation K_1, K_2, L and D is just like in the proof of theorem 9.4.9. We have:

$$FelView^{(t)} = (Shares^{(t-1)}, Shares^{(t)}, Ims^{(t-1)}, Ims^{(t)}, \{\delta_i(\cdot)\}_{i \in D \cap L}, \{Partial_i\}_{i \in D \setminus L})$$

where:

$$\begin{aligned} Shares^{(t-1)} &= \{x_i^{(t-1)}\}_{i \in L \cup K_1} \\ Shares^{(t)} &= \{x_i^{(t)}\}_{i \in L \cup K_2} \\ Ims^{(t-1)} &= \{g^{x_i^{(t-1)}}\}_{i \in \mathcal{A}} \\ Ims^{(t)} &= \{g^{x_i^{(t)}}\}_{i \in \mathcal{A}} \\ Partial_i &= \{\delta_i(j)\}_{j \in L}, \{g^{\delta_{im}}\}_{m \in \{1 \ldots k\}}, \{ENC_j(\delta_i(j))\}_{j \notin L} \end{aligned}$$

Assume that $FelView^{(t)}(x)$ ($Fel(x)$ for short) is not a semantically secure envelope relative to the knowledge of g^x [3]. Let i_0 be some element in $D \setminus L$. (we used this technique in lemmas 9.4.5, 9.4.8 and theorem 9.4.9). Let $Fel_0(x)$ be an envelope just like $Fel(x)$, except that the only partial update polynomial about whom there is no full information is $\delta_{i_0}(\cdot)$. In other words, while $Fel(x)$ contains $\{\delta_i(\cdot)\}_{i \in D \cap L}, \{Partial_i\}_{i \in D \setminus L})$, $Fel_0(x)$ has $\{\delta_i(\cdot)\}_{i \in D \setminus \{i_0\}}, Partial_{i_0}$. Then $Fel_0(x)$ also cannot be a semantically secure relatively to g^x. And hence, there is a $\mathcal{BPP}$ machine A and functions λ and θ, such that for every $\mathcal{BPP}$ algorithm A'

$$\exists_c \forall_{h_0} \exists_{q \text{ s.t. } |q| \geq h_0} Prob[A(Fel_0(x), \theta(x)) = \lambda(x)] > Prob[A'(g^x, \theta(x)) = \lambda(V)] + \frac{1}{|q|^c} \tag{9.18}$$

where the probability is taken over the random choices made by $A, A', Fel(x)$ and over the distribution of x.

From the proof of theorem 9.4.9, envelope $View^{(t)}(x)$ ($View(x)$ for short) is semantically secure even if it contains $\{Full_i\}_{i \in D \setminus \{i_0\}}, Partial_{i_0}$ instead of $\{Full_i\}_{i \in D \cap L}$, $\{Partial_i\}_{i \in D \setminus L})$. Let's call such modified envelope $View_0(x)$.

As mentioned in section 9.4.2, in the discussion of the proactive algorithm using Pedersen's VSS we used notation $ENC_i(\delta_j(i), \gamma_j(i))$ for encrypted transmission of shares of the partial update polynomials $\{\delta_i(\cdot), \gamma_i(\cdot)\}_{i \in D}$. This way it was easier to talk about accusation protocol and about reducing the secrecy of our algorithm to that of encryption $ENC(\cdot)$. However, the same proofs go through if we replace $ENC_i(\delta_j(i), \gamma_j(i))$, with $ENC_i(\delta_j(i)), Enc_i(\gamma_j(i))$. The only difference would be that because we now use $ENC(\cdot)$ as acting on input of size $|q|$ and not $2|q|$, the coefficients that express the relative strength of secrecy protection of our scheme compared to

[3]We will write g^x and g^{x_i} for short when we really mean $g^x \ (mod\ p), g^{x_i} \ (mod\ p)$ etc.

that of $ENC(\cdot)$ would be bigger by a factor of two. Therefore, we will denote:

$$View_0 = (Shares^{(t-1)}, Shares^{(t)}, Ims^{(t-1)}, Ims^{(t)}, \{\delta_i(\cdot), \gamma_i(\cdot)\}_{i\in D\setminus\{i_0\}}, Partial_{i_0})$$

where:

$$\begin{aligned}
Shares^{(t-1)} &= \{x_i^{(t-1)}, z_i^{(t-1)}\}_{i\in L\cup K_1} \\
Shares^{(t)} &= \{x_i^{(t)}, z_i^{(t)}\}_{i\in L\cup K_2} \\
Ims^{(t-1)} &= \{g^{x_i^{(t-1)}} h^{z_i^{(t-1)}}\}_{i\in \mathcal{A}} \\
Ims^{(t)} &= \{g^{x_i^{(t)}} h^{z_i^{(t)}}\}_{i\in \mathcal{A}} \\
Partial_i &= \{\delta_i(j), \gamma_i(j)\}_{j\in L}, \{g^{\delta_{im}} h^{\gamma_{im}}\}_{m\in\{1\ldots k\}}, \{ENC_j(\delta_i(j)), ENC_j(\gamma_i(j))\}_{j\notin L}
\end{aligned}$$

We construct a $\mathcal{BPP}$ machine C that takes as input g^x, $View_0(x)$ and $\theta(x)$ (with probability distribution as in the share renewal algorithm using Feldman's VSS) and does the following: It computes $\{g^{x_i^{(t-1)}}\}_{i\in L\cup K_1}$ from $Shares^{(t-1)}$ and $\{g^{x_i^{(t)}}\}_{i\in L\cup K_2}$ from $Shares^{(t)}$. Together with $g^x = g^{f^{(t-1)}(0)} = g^{f^{(t)}(0)}$, C has $k+1$ values $\{g^{f^{(t-1)}(i)}\}_{i\in\{0,\ldots,k\}}$ and $k+1$ values $\{g^{f^{(t)}(i)}\}_{i\in\{0,\ldots,l\}\cup\{k+1,\ldots,2k-l\}}$. Using the same technique as in the proof of lemma 9.4.6, C interpolates these values for both $f^{(t-1)}(\cdot)$ and $f^{(t)}(\cdot)$ to get the remaining values $\{g^{x_i^{(t-1)}}\}_{i\in\{k+1,\ldots,n\}}$ and $\{g^{x_i^{(t)}}\}_{i\in\{l+1,\ldots,k\}\cup\{2k-l+1,\ldots,n\}}$. Then C computes for every $i \in \{1\ldots n\}$

$$g^{\delta_{i_0}(i)} = g^{x_i^{(t)}} \left(g^{x_i^{(t-1)}} \prod_{j\in D\setminus\{i_0\}} g^{\delta_j(i)} \right)^{(-1)} \quad (mod\ p)$$

and interpolates any k of these values (again as in the proof of lemma 9.4.6) to get $\{g^{\delta_{im}}\}_{m\in\{1\ldots k\}}$. C puts these values together and constructs

$$\begin{aligned}
V = \ & \{x_i^{(t-1)}\}_{i\in L\cup K_1}, \{x_i^{(t)}\}_{i\in L\cup K_2}, \\
& \{g^{x_i^{(t-1)}}\}_{i\in\{1\ldots n\}}, \{g^{x_i^{(t)}}\}_{i\in\{1\ldots n\}}, \\
& \{\delta_i(\cdot)\}_{i\in D\setminus\{i_0\}}, \\
& \{\delta_{i_0}(j)\}_{j\in L}, \{g^{\delta_{i_0 m}}\}_{m\in\{1\ldots k\}}, \{ENC_j(\delta_{i_0}(j))\}_{j\notin L}
\end{aligned}$$

Then C runs $A(V, \theta(x))$ and returns its output. Notice that V has the same probability distribution as $Fel_0(x)$. Therefore,

$$Prob[C(g^x, View_0(x), \theta(x)) = \lambda(x)] = Prob[A(Fel_0(x), \theta(x)) = \lambda(x)]$$

and consequently, from equation 9.18, for every $\mathcal{BPP}$ machine A'

$$\exists_c \forall_{h_0} \exists_{|q|\geq h_0} Prob[C(g^x, View_0(x), \theta(x)) = \lambda(x)] > Prob[A'(g^x, \theta(x)) = \lambda(V)] + \frac{1}{|q|^c}$$

We define a new knowledge function $\theta'(x) = (\theta(x), g^x)$. It follows that for every $\mathcal{BPP}$ machine A'

$$\exists_c \forall_{h_0} \exists_{|q| \geq h_0} Prob[C(View_0(x), \theta'(x)) = \lambda(x)] > Prob[A'(\theta'(x)) = \lambda(V)] + \frac{1}{|q|^c}$$

where probability is computed over probability distributions of $View_0(x)$, x and random choices of C and A'. This contradicts the fact that $View_0(x)$ is a semantically secure envelope scheme and concludes the proof of the theorem. ■

Proof of the secrecy part of theorem 3.1.0: This proof is analogous to the one above. We proceed as follows: We denote the maximal adversarial view of the history of proactive algorithm using Pedersen's VSS (as in the proof of the secrecy part of theorem 9.4.9 in section 9.4.4) as an envelope scheme $Ped(x)$. We define the analogous view for Feldman's VSS and call it $Fel(x)$. We claim that $Fel(x)$ is not semantically secure relatively to the knowledge of g^x. Let $\theta(x)$ be the a priori information and $\lambda(x)$ be the knowledge that one can learn non-negligibly better with $Fel(x)$ then with having only $\theta(x)$ and g^x. We define an envelope $Fel_0(x)$ to be just like $Fel(x)$ except that whenever there is a collection of update or recovery polynomials (polynomials $\delta_i(\cdot)$) about whom $Fel(x)$ has only partial information, we make $Fel_0(x)$ so that it has full information about all of them except of one (for every collection of partial information about re-randomization polynomials $\{Partial_i(\cdot)\}_{i \in D \setminus L}$ in $Fel(x)$, we pick some $i_0 \in D \setminus L$ and we replace this collection with $\{Full_i(\cdot)\}_{i \in D \setminus (L \cup \{i_0\})}, Partial_{i_0}$. Since $Fel(x)$ is polynomial-time computable from $Fel_0(x)$, if $Fel(x)$ is not semantically secure relative to g^x, then neither is $Fel_0(x)$. Analogously, we define $Ped_0(x)$ as a variation of $Ped(x)$ and we argue that even though $Ped_0(x)$ has more information then $Ped(x)$ (i.e. $Ped(x)$ is polynomial-time computable from $Ped_0(x)$), all the proofs of security from section 9.4, and the proof of the secrecy part of theorem 9.4.9 in section 9.4.4 in particular, go through for $Ped_0(x)$ just like for the original envelope $Ped(x)$. Consequently, $Ped_0(x)$ is a semantically secure envelope of x. Now, we use the technique from the proof above to show that with the additional knowledge of g^x, one can construct $Fel_0(x)$ from $Ped_0(x)$ in polynomial-time. We conclude that if we define $\theta'(x) = (\theta(x), g^x)$, then a $\mathcal{BPP}$ machine C that takes $\theta'(x)$ and $Ped_0(x)$ can construct $Fel_0(x)$ and learn $\lambda(x)$ with non-negligibly better probability then any $\mathcal{BPP}$ machine that takes only $\theta'(x)$ as an input. This is a contradiction and hence, $Fel(x)$ must be a semantically secure envelope of x, relative to the knowledge of g^x, which concludes the proof of the theorem. ■

Bibliography

[AMV90] G. Agnew, R.C. Mullin, and S. Vanstone. Improved digital signature scheme based on discrete exponentiation. *Electronics Letters*, 26:1024–1025, 1990.

[BGW88] M. BenOr, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *ACM STOC*, pages 1–10, 1988.

[BM84] M. Blum and S. Micali. How to construct cryptographically strong sequences of pseudorandom bits. *Advances in Cryptology - EUROCRYPT*, SIAM J. Comp.(13):109–125, 1984.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption. *Advances in Cryptology - EUROCRYPT*, LNCS ?:109–125, 1994.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneous broadcast. *Proc. of IEEE Fund. of Comp. Sci.*, pages 335–344, 1985.

[CH94] R. Canetti and A. Herzberg. Maintaining security in the presence of transient faults. *Advances in Cryptology - CRYPTO*, LNCS 839:425–438, 1994.

[CH95] R. Canetti and A. Herzberg. Proactive maintenance of authenticated communication. *unpublished manuscript*, 1995.

[DDFY94] A. DeXSSantis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. *ACM STOC*, 1994.

[DF90] Y. Desmedt and Y. Frankel. Threshold cryptosystems. *Advances in Cryptology - CRYPTO*, LNCS 435:307–315, 1990.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[Fel87] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. *Proc. of IEEE Fund. of Comp. Sci.*, pages 427–437, 1987.

[FY92] M. Franklin and M. Yung. Communication complexity of secure computation. *ACM STOC*, pages 699–710, 1992.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Com. Sys. Sci.*, 28:270–299, 1984.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A secure digital signature scheme. *Siam Journal on Computing*, 17(2):281–308, 1988.

[Gol89] O. Goldreich. *Foundations of Cryptography.* Class Notes, Technion, Israel, 1989.

[Har94] L. Harn. Group oriented (t,n) digital signature scheme. *IEEE Proc.-Comput.Digit.Tech.*, 141(5), September 1994.

[HJJ$^+$95] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signatures systems. *draft*, 1995.

[HPM94] P. Horster, H. Petersen, and M. Michels. Meta-elgamal signature schemes. *ACM 2-d CCS*, pages 96–107, 1994.

[JJKY95] M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. *Manuscript on proactive RSA*, 1995.

[LW88] D.L. Long and A. Wigderson. The discrete log problem hides o(log n) bits. *SIAM J. Comp.*, 17:363–372, 1988.

[NR94] K. Nyberg and R. Rueppel. Message recovery for signature schemes based on the discrete logarithm. *Advances in Cryptology - EUROCRYPT*, pages 175–190, 1994.

[OY91] R. Ostrovsky and M Yung. How to withstand mobile virus attacks. *Proc. of the 10th ACM Symp. on the Princ. of Distr. Comp.*, pages 51–61, 1991.

[Ped91] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. *Advances in Cryptology - CRYPTO*, LNCS 576:129–140, 1991.

[Rab88] T. Rabin. *Robust Sharing of Secrets when the Dealer is Honest or Cheating.* Masters Thesis, Hebrew University, Jerusalem, 1988.

[RB89] T. Rabin and M. BenOr. Verifiable secret sharing and multiparty protocols with honest majority. *ACM STOC*, ?:73–85, 1989.

[Riv90] R. Rivest. Cryptography. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 13 (section 10.1.1), pages 746–747. Elsevier Science Publishers, 1990.

[Sha79] A. Shamir. How to share a secret. *Commun. ACM*, 22:612–613, 1979.

[Sim92] G. J. Simmons. An introduction to shared secret and/or shared control schemes and their applications. In G. J. Simmons, editor, *Contemporary Cryptology, The Science of Information Integrity*, chapter 9, pages 441–497. IEEE Press, New York, 1992.