Fachbereich Mathematik

Masterarbeit

Efficient Proactive Secret Sharing

Jacqueline Brendel

26. Januar 2016

Betreuer: Prof. Dr. Johannes Buchmann

Zweiter Gutachter: Dr. Denise Demirel

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und alle benutzten Quellen einschließlich der Quellen aus dem Internet und alle sonstigen Hilfsmittel angegeben habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 26.01.2016

Jacqueline Brendel

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Dr. Denise Demirel for her enthusiasm, invaluable advice and encouragement.

For the non-scientific side of my thesis, I particularly want to thank my partner Alex for his infinite patience, love and many coffee dates. Your ambition and kindness never fail to inspire me.

Last but not least, I would like to thank my family, especially my mother Sophie, and my sister Sandra, for always keeping me grounded, never losing faith in me and supporting me in every way imaginable throughout my life.

Contents

1 Introduction

1.1 Short Introduction

Long-term secure archival is growing in its relevance due to the ever increasing digitalization of documents throughout all areas of our lives, for example medical or legal data. There exist various meanings of the expression *long-term* to describe the life time of the data in question, ranging from a few years or decades to everlasting, but in all cases it is clear that storage of this kind of data must take special precautions to ensure the basic protection goals of authenticity, integrity and confidentiality. In the course of the stored document's life time, hardware will be replaced and network structures will change repeatedly. Long-term storage solutions must be able to handle these changes while retaining the required protection goals for the data. Long-term authenticity and integrity are well-studied (cf. e.g. [30] for a comprehensive overview over the approaches), whereas long-term confidentiality remains a major open research question. So far, the only acknowledged solution to protect the confidentiality of long-term relevant data are *Proactive Secret Sharing* schemes which will be discussed in detail in the next chapter. As opposed to traditional threshold secret sharing schemes such as Shamir's[26], proactive secret sharing schemes limit the time frame an adversary has at its disposal to compromise the threshold number of shares, by renewing the shares periodically without reconstructing the original secret. The renewed shares still combine to the original secret, but any knowledge the adversary has gained about previous shares becomes useless once the renewal was completed. Thus, proactive secret sharing schemes are especially well suited to protect sensitive data with a long life span.

1.2 Problem Statement and Contribution

The benefits of proactive secret sharing schemes compared to standard archival solutions when it comes to long-lived sensitive data are obvious. Unfortunately, the widespread adaptation of proactive secret sharing is hindered by its impracticality. As a result of the large amount of traffic generated during the periodic share renewal and redistribution processes as well as the accompanying computational overhead, these schemes do not scale well and are considered inefficient. Furthermore, due to their abstract nature, prevailing proactive secret sharing protocols omit details concerning the practical implementation of long-term security related primitives. And yet, proactive secret sharing combined with verifiable secret sharing is recognized as a sound and the only currently known approach to long-term secure archival of

confidential information. The practical implementation of such schemes is an open issue in ongoing research.

In this thesis we wish to relax the commonly made strong assumption that there exist private point-to-point communication channels between any two nodes involved in the PSS (re)distribution and reconstruction process. This is done by clustering the shareholders into groups that can only communicate securely within the cluster itself and across clusters through single distinguished nodes within the cluster. We will see that this modified situation lends itself well to structures commonly found in enterprises. This so-called enterprise scenario is then the basis for the introduction of the Enterprise PSS (EPSS) scheme that is able to accommodate the modified assumption while still retaining strong security properties.

The basis for the EPSS approach will consist of the PSS protocol by Gupta and Gopinath [16] combined with methods found in asynchronous proactive secret sharing schemes (cf. section 2.3.2) which achieves a further step towards practicability by relaxing the assumptions on the network from synchronous to asynchronous. The choice of [16] is due to its reasonable assumptions which model an apt enough representation of the real world. Additionally, it allows for dynamic addition and removal of shareholders without reconstructing the original secret. Its use of Pedersen commitments over Feldman VSS provides information-theoretic security which is agreed to be the adequate security level for storage of long-lived sensitive data, therefore making it a solid starting point not only for our but any further research in practical proactive secret sharing schemes. Additionally, we will conduct a short analysis concerning possible recommendations for concrete implementations for long-term security related features of PSS protocols such as [16].

1.3 Thesis Outline

The thesis is structured as follows: In the next section, we will portray the historic evolution of proactive secret sharing while introducing the relevant cryptographic building blocks and preliminaries. In Chapter 3, the protocol by Gupta and Gopinath [16] will be presented in detail before we specify its long-term secure aspects in Chapter 4. The remainder of the thesis is dedicated to introducing and presenting the Enterprise PSS protocols for initial distribution, redistribution and reconstruction (Chapter 6) followed by a security analysis (Chapter 7). The thesis will close with a short comparison of the presented scheme and [16] with respect to communication complexity (Chapter 8) and an outlook on future work concerning the topic of proactive secret sharing.

2 Foundations of Proactive Secret Sharing

Proactive secret sharing schemes can be said to have evolved from traditional secret sharing schemes in an iterative process where shortcomings were identified and then resolved. We want to outline this development and, in the course of doing so, introduce preliminaries and notions which are relevant for the rest of the thesis.

2.1 Shamir's Secret Sharing

Secret sharing allows for a secret s to be divided into so-called *shares* which are then distributed secretly among a number of n parties $P_1, \ldots, P_n$, commonly referred to as *shareholders*. The combination of any m shares, i.e., the collaboration of at least m shareholders, enables the reconstruction of the original secret s, while fewer than m shares will not suffice to gain any additional information about s.

This approach is especially useful for highly sensitive data which must be stored reliably and access should only be authorized if a minimum number of shareholders collaborate. A popular example is the storage of signature keys, where it should be impossible for a single person to sign e.g. the final version of a software product, whereas a distinguished group of individuals can generate the signature needed for its release.

Due to their construction such schemes are also called (m, n)-*threshold schemes*. The concept of secret sharing was first introduced independently by Adi Shamir [26] and George Blakley [4] in 1979. Shamir's approach is based on the interpolation of polynomials over finite fields and assures that an adversary which possesses less than m shares, learns no partial information about the secret s. This guarantee is not given in Blakley's original scheme, which conceals the secret as an intersection of n-dimensional hyperplanes. Furthermore, while the size of the shares in Shamir's protocol is the same as that of the secret, Blakely's protocol produces shares which are $m-1$ times the length of the secret. For these reasons, Shamir's Secret Sharing is more widely deployed and we will describe in detail how it works:

Sharing Let D be the *dealer*, who holds the secret $s \in \mathbb{Z}_p$ with p prime and $p > n$. D now chooses coefficients $a_1, \ldots, a_{m-1} \in_R \mathbb{Z}_p$ uniformly at random which determine the polynomial $a(x) = s + a_1 x + a_2 x^2 + \cdots a_{m-1} x^{m-1}$ of degree $m-1$ with constant term s, i.e., $a(0) = s$. The share s_i of participant P_i is then given by $s_i := a(i) = s + \sum_{k=1}^{m-1} a_k i^k$ with $i = 1, \ldots, n$.

Reconstruction For reconstruction of the secret s a subset B containing any m shareholders is chosen, say, $P_{i_1}, P_{i_2}, \ldots, P_{i_m}$ and their respective shares s_{i_j} are retrieved. Polynomial interpolation then yields the unique polynomial p such that $p(i_j) = s_{i_j}$ for $j = 1, \ldots, m$. It holds that $p(x) = a(x)$. Thus, the secret s is then given by evaluation of the polynomial p at zero.

Remark 2.1. It is reasonable to choose the number of participants n such that $n = 2m - 1$ as this implies that the secret can be recovered if and only if there is an honest majority. We assume that there always exists an honest majority as otherwise, the dishonest parties could collude and simply reconstruct the secret without further ado.

2.2 Verifiable Secret Sharing

In the above described secret sharing shemes, less than m shareholders are not able to determine the interpolating polynomial and can therefore not reconstruct the secret s as long as the dishonest parties remain *passive*. But what happens if the $m - 1$ shareholders collude, and do not remain passive eavesdroppers but *actively* try to gain unwarranted access to information about the secret through malicious behaviour? What if the dealer is dishonest? To answer these questions, we must first define what it means for a dealt share to be valid:

Definition 2.2. We say that a share s_i of a secret s is *valid* (or *consistent*) if $s_i = a(i)$ for the previously determined secret sharing polynomial $a(x)$ with constant term s.

A dishonest dealer could, for example, distribute invalid shares to (some of) the participants which - if used for reconstruction - would not yield the polynomial associated with the original secret and the secret could be lost. Furthermore, during the reconstruction phase there is no way to ascertain that the shares provided by the select group of shareholders are indeed the ones which were originally dealt which could again impact the recovery of the secret. We recognize the need to be able to confirm the consistency of shares with the original secret. This is achieved by *Verifiable Secret Sharing (VSS)* schemes. The first interactive VSS scheme (along with the general idea) was proposed in 1985 by Chor, Goldwasser, Micali and Awerbuch [9]. Two commonly used examples are the computationally secure Feldman VSS [14] and the information-theoretically secure Pedersen VSS [22]. Both are non-interactive and in the following their application to secret sharing schemes is laid out.

2.2.1 Feldman VSS

Let p and q be sufficiently large prime numbers with $p = qr + 1$. Now let $s \in \mathbb{Z}_q$ be the secret which is to be distributed by a dealer D according to an (m, n)-threshold secret sharing scheme. Furthermore, let $g \in \mathbb{Z}_p$ of order q. It is assumed that the computation of discrete logarithms is intractable.

1. The dealer shares the secret s as described in section 2.1. Additionally, the dealer computes the values $g^{a_1}, g^{a_2}, \ldots, g^{a_{m-1}}$ and broadcasts them along with the concealed secret g^s.

2. Each P_i can now verify its received share s_i by checking if the congruence

$$g^{s_i} \equiv (g^s) \cdot (g^{a_1})^i \cdot (g^{a_2})^{i^2} \cdots (g^{a_{m-1}})^{i^{m-1}} \tag{1}$$

   is satisfied.

Upon reconstruction of the secret, the provided shares can be checked for consistency with equation (1) by the reconstructing party. The occurrence of discrepancies in the verification equation (1) is handled differently in protocols but is most commonly dealt with by implementing a complaint mechanism where shareholders can accuse the dealer of cheating and the issue is then resolved by identifying the dishonest party. Since usually an honest majority of shareholders is assumed, the existence of at least m participants with consistent shares is guaranteed and hence the secret remains reconstructible even in the presence of dishonest participants.

2.2.2 Pedersen VSS

Besides being very efficient, Feldman VSS [14] is only computationally secure as the confidentiality of the secret relies on the hardness of computing discrete logarithms over finite fields. Information-theoretic security is achieved by Pedersen's verifiable secret sharing protocol [22] which works as follows:

Again, let p and q be sufficiently large prime numbers with $p = qr + 1$ and $s \in \mathbb{Z}_q$ be the secret which is to be distributed by a dealer D according to an (m, n)-threshold secret sharing scheme. Generators g and h of a q-th order subgroup of $\mathbb{Z}_p^*$ are selected by D, i.e., it holds that $g^q \equiv 1 \mod p$ and $h^q \equiv 1 \mod p$ and made public. It is assumed that $log_g(h)$ is not known as otherwise the dealer D would be enabled to distribute invalid shares without detection.

1. The dealer shares the secret s as described in section 2.1 without sending the shares to the participants just yet. Let $t \in_R \mathbb{Z}_q$ be chosen by D arbitrarily at random. The secret sharing procedure is also applied to the value t, i.e., the polynomial $b(x) = t + b_1 x + b_2 x^2 + \cdots + b_{m-1} x^{m-1}$ with uniformly random coefficients is defined and shares $t_i := b(i)$ for P_i, $i = 1, \ldots, n$ are computed. The share pair (s_i, t_i) is then transmitted privately to the respective participant.

   The dealer D then computes the values $g^s h^t$ and $g^{a_1} h^{b_1}, g^{a_2} h^{b_2}, \ldots, g^{a_{m-1}} h^{b_{m-1}}$ and broadcasts them.

2. The shareholders can now verify their received share pair with the equation

$$g^{s_i} h^{t_i} \equiv g^s h^t \prod_{j=1}^{m-1} \left(g^{a_j} h^{b_j}\right)^{i^j}. \tag{2}$$

2.3 Proactive Secret Sharing

We have seen that secret sharing schemes can be made secure against active adversaries, who willingly distribute invalid shares with the introduction of verifiability features as provided by Feldman or Pedersen commitments to the original secret s. Unfortunately, when dealing with long-lived sensitive documents which must remain confidential over decades, even verifiable secret sharing schemes are ill-suited. They cannot provide confidentiality and integrity over the whole life span of such long-lived secret since their security depends on the assumption that a fixed upper bound for the number of compromised shares exists, namely less than m. In the long term setting this is inappropriate as adversaries can use the entire lifetime of the secret to incrementally gain access to sufficiently many shares to reconstruct the secret.

2.3.1 Synchronous Proactive Secret Sharing

With the introduction of proactive secret sharing in 1995, Herzberg et al. [17] resolved the above mentioned shortcoming. Their simple but yet revolutionary idea was to prevent the number of exposed or otherwise lost shares from exceeding the threshold m by renewing the shares periodically without changing the secret. This periodic refreshment is performed even if there is no indication of faults or adversarial activity in the system. Therefore, the time frame in which an adversary must collect m shares is dramatically reduced since any $m - 1$ or less shares collected before the renewal are rendered useless. The protocol by Herzberg et al. supports either Feldman [14] or Pedersen [22] commitments. In Herzberg et al.'s as well as in all the

schemes mentioned within this paragraph, a synchronous network is assumed. There exist schemes which support asynchronous networks and they will be discussed in short in the next section.

Two years later, a further advancement was made by Desmedt and Jajoda [12]. Unlike Herzberg et al.'s approach, their scheme was able do deal with permanent compromise by enabling the dynamic removal and addition of shareholders. However, this came at the cost of the verifiability feature and so their scheme allowed faulty shareholders to undetectably invalidate shares during the redistribution process. This drawback was solved in 2002 by Wong et al. [32] who ensured that new shareholders could verify the correctness of the shares they were dealt. Unfortunately this was only possible under the strong (and perhaps in some scenarios unrealistic) assumption that all receiving shareholders must be honest during the redistribution process.

In their first paper from 2006 [15], Gupta and Gopinath built on the protocol by Wong et al. and achieved to relax this controversial condition insofar as only a simple majority of the recipients needed to be honest during the (re)distribution process. As Wong et al., Gupta and Gopinath used Feldman's VSS and therefore only established computational security. In their revised protocol called G^2_{its} from 2007 [16] they incorporated Pedersen commitments, achieving information-theoretic security.

2.3.2 Asynchronous Proactive Secret Sharing

The main body of work of PSS schemes that remove the synchronous network assumption, is covered by the papers of Cachin et al.[6], Zhou at al.[35] and Schultz and Liskov[25]. In contrast to synchronous network, asynchronous networks have no access to a common global clock. This kind of environment eliminates the ease with which synchronous proactive secret sharing protocols could tell when redistributions would be initiated according to their schedule. In asynchronous proactive secret sharing schemes, these time intervals need now be defined in terms of events in the protocol themselves (e.g. in [25]) or by conservative estimates on the execution time of redistribution (cf. [35]). Additionally, there exist no upper bounds on message delivery delays and processor execution speeds which makes it more problematic to identify faulty parties as e.g. not receiving a message from a server could either mean that it is corrupted or it is simply slow to respond (but otherwise functioning correctly). Under these demanding circumstances, agreement has to be achieved for the share (re)distribution process to be successful. Nevertheless, asyn-

chronous proactive secret sharing schemes tend to be closer to applications in the sense that they are less susceptible to attacks that slow down processor execution speeds or delay messages within the network. Also many networks found in the wild are not synchronous (e.g. the Internet).

All proactive secret resharing (or redistribution) protocols can be divided into two abstract tasks:

1. The **computation of the resharing** by the current shareholders and
2. the **agreement on a resharing** and the subsequent transfer of the new shares to the new shareholders.

While the first task of resharing computation can be solved by methods already presented in synchronous proactive secret schemes such as the creation of a sub-sharing (e.g. in [12],[32],[16],[6]) or by adding a sharing of the zero element to already existing shares (cf. [17], [25]), the agreement procedure has to be adjusted to the new asynchronous situation. This is due to the fact that one can no longer rest assured that all honest nodes will be able to broadcast their vote e.g. during a majority vote execution. The scheduling of messages within the network is commonly assumed to be under the control of the adversary and therefore such message could be held back. Agreement is then ordinarily reached by so-called *Byzantine Agreement* protocols such as the Byzantine Fault Tolerance protocol by Castro and Liskov[8]. BFT allows arbitrary faulty behaviour of $m-1 < \lfloor \frac{n}{3} \rfloor$ of the n involved parties without relying on synchrony to guarantee optimal resilience. BFT requires one of the involved parties to serve as *primary*. In a nutshell, BFT itself is divided up into three parts:

1. PRE-PREPARE: the primary broadcasts the value over which agreement is to be established.
2. PREPARE: the nodes which are inclined to accept the value broadcast their agreement.
3. COMMIT: if a node sees $2m-1$ agreeing PREPARE messages on the broadcast channel it commits to the value. After again $2m-1$ COMMIT messages have been seen, the protocol commits locally.[1]

In asynchronous PSS schemes the primary is often referred to as the *coordinator*. It oversees for example the redistribution process and handles the agreement over

[1] The value of $2m-1$ guarantees that even if up to $m-1$ messages have been sent by misbehaving nodes, at least m messages by non-faulty parties have been recorded.

which values are used for the share renewal. The coordinator is not assumed to be fully trusted and its behaviour is scrutinized by the other nodes such that if there exists doubt about the honesty of the coordinator, it can be replaced.

3 G^2_{its} Verifiable Proactive Secret Sharing Protocol

3.1 Description

In this section a detailed description of the verifiable proactive secret sharing protocol by Gupta and Gopinath [16] will be given. As mentioned before, their G^2_{its} VSR protocol will constitute the foundation for the modification within the enterprise scenario which will be introduced in section 5.

3.1.1 Setup and Terminology

Let p and r be sufficiently large prime numbers with $r = pq + 1$. As usual let $\mathbb{Z}_p$ and $\mathbb{Z}_r$ denote the prime fields with modulo p and modulo r arithmetic, respectively. Now let $k \in \mathbb{Z}_p$ be the secret which is initially held by the distinguished client node C. The secret will be shared proactively among the access structure $[n, m]$ where n is the number of nodes to receive a share and m is the minimum (threshold) number of shares which are necessary to reconstruct the secret. Note that the secret should be reconstructible if and only if there is an honest majority, i.e., for any access structures $[n, m]$ it holds that $n = 2m - 1$. Since the protocol allows for redistribution of the shares to a new set of nodes with accordingly chosen threshold numbers, let $[n', m'], [n'', m'']$, etc. denote these different access structures. To be able to establish witnesses for the Pedersen commitment, generators $g, h \in \mathbb{Z}_r$ are selected by the client C such that $g^p \equiv 1 \mod r$, $h^p \equiv 1 \mod r$ and $log_g(h)$ is not known. Furthermore let $t \in \mathbb{Z}_p$ be chosen arbitrarily at random as a one-time pad for the secret k.

3.1.2 Assumptions and Requirements

General Assumptions In addition to the above, it is required that at any point in the protocol at most $m - 1$ servers are compromised, i.e., there always exists an honest majority. In particular, it is assumed that during (re)distribution there is always a (simple) honest majority among the receiving nodes. At a certain point in the protocol it will be necessary that all honest participating nodes agree on a single set of nodes within a precomputed sequence that satisfy a given validity condition. To achieve this an algorithm $\mathcal{A}(x, I, \mathcal{Z})$ is established during initialization which will output a sequence $\mathcal{Z}$ of subsets of I containing x elements each. The algorithm $\mathcal{A}(x, I, \mathcal{Z})$ is known to all participating nodes.

Adversary Assumptions The underlying threat model is that of a mobile and active adversary. I.e., an adversary that can dynamically attack nodes within the network by moving from node to node and that can cause arbitrary (malicious) behaviour of the compromised nodes. All (secret) information stored within a compromised server becomes available to the adversary. In particular this implies that the adversary can spoof and decrypt messages. The threshold m in the secret sharing scheme was chosen as such that an adversary can never corrupt more than $m-1$ nodes within the network at any given time.

Network Assumptions The communication capability between the nodes is provided by a group communication system $\mathcal{G}$ which enjoys the following properties:

- guaranteed reliable delivery of messages
- private point-to-point communication channels between any two nodes
- reliable broadcast channel including all participating nodes
- authentication measures are in place (no spoofing is possible)

This implies in particular, that a synchronous network is assumed, i.e., a network in which there exist fixed upper bounds on message delays within the network.

3.1.3 Protocol Description

Initial Distribution The secret k is initially distributed by the client C to the access structure $[n, m]$ the following way:

1. With p, r and t chosen as described above, the client C picks coefficients a_l and b_l, $l = 1, \ldots, m-1$ to form the polynomials $a(x) = k + \sum_{l=1}^{m-1} a_l x^l$ and $b(x) = t + \sum_{l=1}^{m-1} b_l x^l$. The shares $s_i := a(i)$ of k and $t_i := b(i)$ of t can now be computed and the share pair (s_i, t_i) is then sent privately to node i of the access structure $[n, m]$ for each $i = 1, \ldots, n$.

2. To share the secret in an information-theoretic secure way, the client uses the generators g and h to compute the witnesses $g^k h^t, g^{a_1} h^{b_1}, \ldots, g^{a_{m-1}} h^{b_{m-1}}$ which are then broadcast to all n share holders.

3. Each node i uses its received share pair and the witnesses to verify:

$$g^{s_i} h^{t_i} \equiv g^k h^t \prod_{l=1}^{m-1} (g^{a_l} h^{b_l})^{i^l} \tag{3}$$

   (i) If equation (3) holds, node i accepts (s_i, t_i) and saves it as its share pair along with the witness $g^k h^t$.

   (ii) If the verification check fails, node i broadcasts an authenticated complaint stating that it did not receive a correct share from client C. In response the client gets the chance to defend itself by broadcasting the share pair $(\hat{s}_i, \hat{t}_i)$ that it claims to have sent to node i. All members of $[n, m]$ now use these values to check equation (3). If the check is successful, i saves $(\hat{s}_i, \hat{t}_i)$ as its share pair along with the witness $g^k h^t$. Otherwise the nodes that detect a discrepancy in equation (3) remark that there exists a valid complaint against C. A majority vote can now determine if the complaint was justified since the majority of the nodes are assumed to be honest. If necessary, i.e., if the complaint was valid, the protocol is then aborted.

Redistribution Periodically the shares get redistributed from the current access structure $[n, m]$ (sender nodes) to a new access structure $[n', m']$ (receiver nodes). Note that the two sets need not be disjoint.

1. Each sender node i applies the secret sharing procedure to its shares s_i and t_i, i.e., node i picks coefficients a'_{il} and b'_{il}, $l = 1, \ldots, m' - 1$ to form the polynomials $a'(x) = s_i + \sum_{l=1}^{m'-1} a'_{il} x^l$ and $b'(x) = t_i + \sum_{l=1}^{m'-1} b'_{il} x^l$. From this, sub-shares $\hat{s}_{ij} := a'(j)$ and $\hat{t}_{ij} := b'(j)$ for each receiver node j in $[n', m']$ are obtained and are then sent privately to the respective node j.

2. To generate the witnesses every sender node i computes $g^{s_i}, g^{a'_{i1}}, \ldots, g^{a'_{i(m'-1)}}$ and $h^{t_i}, h^{b'_{i1}}, \ldots, h^{b'_{i(m'-1)}}$ where g, h are the same generators as chosen in the initial distribution. Their respective products $g^{s_i} h^{t_i}, g^{a'_{i1}} h^{b'_{i1}}, \ldots, g^{a'_{i(m'-1)}} h^{b'_{i(m'-1)}}$ are then broadcast to the n' receiver nodes.

3. Each node $j \in [n', m']$ uses its received sub-share pairs and the respective witnesses to verify that for all $i = 1, \ldots, n$:

$$g^{\hat{s}_{ij}} h^{\hat{t}_{ij}} \equiv g^{s_i} h^{t_i} \prod_{l=1}^{m'-1} \left(g^{a'_{il}} h^{b'_{il}}\right)^{j^l} \tag{4}$$

(i) If equation (4) holds, node j accepts the $(\hat{s}_{ij}, \hat{t}_{ij})$ and saves it as its sub-share pair from node i.

(ii) If the verification check fails, node j broadcasts an authenticated complaint stating that it did not receive a correct share from sender node i. In response node i can defend itself by broadcasting the share pair $(\hat{\hat{s}}_{ij}, \hat{\hat{t}}_{ij})$ that it claims to have sent to node j. All members of $[n', m']$ now use these values to check equation (4). If the check is successful, j saves $(\hat{\hat{s}}_{ij}, \hat{\hat{t}}_{ij})$ as its sub-share pair. Otherwise the nodes that detect a discrepancy in equation (4) mark node i with the valid complaint label **VC**.

4. Since all honest receiver nodes, which again form a majority, flagged the same sender nodes as **VC**, the receiver nodes can now exclude these sender nodes from the protocol. Remark that each honest member of $[n', m']$ has the same set of reduced sender nodes $\mathcal{R}$ and since there exists an honest majority among the sender nodes, $\mathcal{R}$ contains at least m nodes.

   Let $|\mathcal{R}| = u$. The receiver nodes now apply algorithm $\mathcal{A}$ to $\mathcal{R}$ which constructs a sequence $\{\mathcal{B}_u\}_{u=1,\ldots,M}$ of m-subsets where $1 \leq M \leq \binom{u}{m}$.

5. Each node $i \in [n, m]$ also broadcasts its $g^k h^t$ witness. Each receiver node j stores the value $g^k h^t$ which was sent by the (honest) majority of sender nodes.

6. For each $u = 1, \ldots, M$ the receiver nodes now check

$$g^k h^t \equiv \prod_{i \in \mathcal{B}_u} (g^{s_i} h^{t_i})^{b_i} \qquad \text{where} \qquad b_i = \prod_{\substack{l \in \mathcal{B}_u, \\ l \neq i}} \frac{l}{l - i} \tag{5}$$

   Obviously, there exists at least one such $\mathcal{B}_u$. Now let $\mathcal{B}_{\tilde{u}}$ be the first one to satisfy equation (5) for all its members.

7. Each node $j \in [n', m']$ now establishes its pair of shares (s'_j, t'_j):

$$s'_j = \sum_{i \in \mathcal{B}_{\tilde{u}}} b_i \hat{s}_{ij} \qquad \text{and} \qquad t'_j = \sum_{i \in \mathcal{B}_{\tilde{u}}} b_i \hat{t}_{ij} \qquad \text{where} \qquad b_i = \prod_{\substack{l \in \mathcal{B}_{\tilde{u}}, \\ l \neq i}} \frac{l}{l - i} \tag{6}$$

   and stores it along with the witness $g^k h^t$.

Reconstruction If the reconstruction of the original secret k is wished by the client the following steps need to be taken:

1. The client C requests the shares from each node i in the current access structure, say $[n, m]$. Each node then returns its share pair (s_i, t_i) privately to the client.

2. The client applies the aforementioned algorithm $\mathcal{A}$ which results in a sequence $\{\mathcal{B}_u\}_{u=1,\ldots,M}$ where $M \leq \binom{n}{m}$ of m-subsets of the n nodes.

3. The client successively checks each $\mathcal{B}_u$ for its share validity through the following equation:

$$g^k h^t \equiv \prod_{i \in \mathcal{B}_u} (g^{s_i} h^{t_i})^{b_i} \qquad \text{where} \qquad b_i = \prod_{\substack{l \in \mathcal{B}_u, \\ l \neq i}} \frac{l}{l-i} \tag{7}$$

4. Upon finding the first $\mathcal{B}_u$ such that all m shares in it are valid, say $\mathcal{B}_{\tilde{u}}$, the client can now use $\mathcal{B}_{\tilde{u}}$ to reconstruct the original secret k via

$$k = \sum_{i \in \mathcal{B}_{\tilde{u}}} s_i b_i \qquad \text{where} \qquad b_i = \prod_{\substack{l \in \mathcal{B}_{\tilde{u}}, \\ l \neq i}} \frac{l}{l-i} \tag{8}$$

   Note that such a $\mathcal{B}_{\tilde{u}}$ is always found since there are at least m honest nodes (by assumption).

3.2 Remark

Though being one of the most advanced proactive secret sharing protocols currently available, the protocol by Gupta and Gopinath has certain shortcomings which we shortly want to present and which will be addressed in the following chapters where the EPSS protocol will be introduced. First and foremost, for the concrete applicability of the scheme, the synchronous network assumption should be removed or relaxed since such guarantees can usually not be given within real world systems. Furthermore, there are no indications as to how to proceed when the protocol is aborted. Is it re-initiated? If so, with what changes to e.g. the client or the access structure? Lastly, we want to mention that if long-term security is to be achieved, the issue of expiring bindingness properties of Pedersen commitments should be discussed. Otherwise the long-term integrity of the secret cannot be assured. We will address this issue shortly in the next chapter, as well as the details concerning the concrete implementation of other long-term security relevant features within the protocol.

4 Long-term Secure Implementation of G^2_{its}

In this section we want to put the long-term secure implementation of the abstract G^2_{its} protocol in concrete terms by first pinpointing the relevant elements with respect to long-term security and then providing suitable solutions, all the while bearing in mind the information-theoretic secure setting which is to be preserved.

4.1 The Commitment Scheme

Pedersen VSS [22] is used in G^2_{its} to guarantee the integrity of the (re)distributed shares at any point in the protocol. Pedersen commitment schemes are *perfectly hiding*, i.e., the value committed to remains secret until opened, but unfortunately only *computationally binding*. This means that after a certain period of time, the integrity of the value committed to by the sender is no longer given. The authors of the G^2_{its} Verifiable Secret Redistribution protocol give no indication as to how to proceed in this case, but in their 2015 paper *How to Securely Prolong the Computational Bindingness of Pedersen Commitments* [11], Denise Demirel and Jean Lancrenon show how to resolve this issue for the G^2_{its} protocol. Once a Pedersen commitment with a larger security parameter to the secret value is created, their protocol describes how to provide an efficient perfect zero-knowledge proof that the newly created commitment with updated security parameter commits to the same secret as the one of which the bindingness was about to run out.

4.2 The Communication System

With G^2_{its} VSR being an interactive protocol, the correct implementation of communication between the participants plays an integral part. The key requirements given in [16] are the reliable delivery of messages, the existence of a broadcast channel and the possibility to have both authenticated and private communication links between any two nodes, which we will start our analysis with.

It is crucial for all private transmissions to be protected in the long term since an attacker could simply store all occurring traffic until it becomes feasible to break the applied encryption, thereby compromising each share and consequently the whole secret. With this, it is easy to see that public key encryption and key agreement schemes cannot provide the desired level of confidentiality due to their underlying computational assumptions and vulnerability to exhaustive key search.

At this point it is worth noting that this argument does not apply to the authentication requirement. Since authenticity of the sender is only relevant within a

narrow time frame of the interaction between the nodes, public key signatures are sufficient and need not be replaced by long-term secure signatures. All in all, we are now left looking for information-theoretic secure encryption, i.e., schemes which are provably secure even against an adversary with unbounded computational power. A special case of these are *perfectly secret* schemes and from the discussion above, it is clear that the solution can only be found among symmetric cryptographic systems.

4.2.1 Reliable Communications

It is necessary to assume that the underlying network structure connecting the partaking nodes supports reliable delivery of messages on the broadcast as well as the private channels. The assumption has to be made since such considerations do not include long-term security related aspects and are therefore outside the scope of this thesis.

4.2.2 The Broadcast Channel

Gupta and Gopinath assume the existence of a reliable broadcast channel but there are two problems with this requirement: The term *broadcast* is defined such that it reaches all nodes in a network. This behaviour might not actually be wished for in the case of the G^2_{its} protocol. Often it is only necessary that a subset of nodes receive a certain message.

Take for example the share redistribution process during which each node i in $[n, m]$ broadcasts its witnesses to the nodes in the new access structure $[n', m']$. In a strict broadcast scenario, this message would also be sent to the nodes in the current access structure $[n, m]$, which is unnecessary. The described behaviour is more accurately covered by what is commonly referred to as *multicast* where a message is sent to a select group of receivers instead of the entire network.

Furthermore, the question of reliable multicast (or broadcast, for that matter) is still subject to ongoing research and is by no means satisfactorily answered for the general case. Therefore, most current networks are not able to meet this requirement. In order to still be able to achieve the desired specifications, reliable multicast has to be mimicked through *unicasts*. This means that the sender initiates a separate transmission of the same message to each receiving node. For messages from one sender to one receiver reliable delivery can be guaranteed. However, this approach cannot guarantee to the receivers by itself they all received the same message.

If indeed a broadcast channel is required many options are available. One of the most illustrative examples of such is a public (digital) bulletin board which we will use throughout the rest of the thesis for its comprehensibility. The main ideas

behind this medium are that it is accessible for all participants to post on it or read the messages published on it. Furthermore, it is assumed that messages on the bulletin board cannot be deleted or changed. Bulletin boards are usually used within e-voting protocols, where they allow voters to e.g. verify that their vote has been counted. Later in our protocol, for example, the commitment to the secret $g^k h^t$ will be published on the bulletin board by the client during the initial distribution. In all later redistribution steps, the nodes will use this value for their verification of shares. The indefinite availability of the client's initial commitment on the bulletin board makes it unnecessary for shareholders to keep track of the commitment throughout the long life-time of the secret.

4.2.3 Perfectly Secret Encryption

First, we want to give a formal definition of what it means for an encryption scheme $\mathcal{E}$ =(**Gen, Enc, Dec**), consisting of a key-generation algorithm **Gen**, an encryption algorithm **Enc** and a decryption algorithm **Dec**, to be perfectly secret:

Definition 4.1 (Perfect Secrecy)**.** An encryption scheme $\mathcal{E}$ =(**Gen, Enc, Dec**) with message space $\mathcal{M}$ is *perfectly secret* if for every probability distribution over $\mathcal{M}$, every message $m \in \mathcal{M}$, and every ciphertext $c \in \mathcal{C}$ for which $\Pr[C = c] \geq 0$:

$$\Pr[M = m | C = c] = \Pr[M = m].$$

Shannon showed that for perfect secrecy to be achieved the key space $\mathcal{K}$ must be at least as large as the message space $\mathcal{M}$ [27], i.e.,

$$|\mathcal{K}| \geq |\mathcal{M}|. \tag{9}$$

For schemes with fixed key length, this implies in particular that the key length must be greater or equal to the message length. This imposes an undeniable limitation with regard to practicality on such ciphers since a large amount of secret key material must be agreed on and administered in a secure manner. Nevertheless, as an essential component in achieving long-term confidentiality, this highest of all security levels cannot be dismissed so easily, as shown for example by Wolf in [31]. Probably the best-known example of a perfectly secret cipher is the one-time-pad (OTP), made popular by Vernam [29] in 1917:

Vernam OTP Let $n > 0$ be fixed. As before, let $\mathcal{K}$ denote the key space, $\mathcal{M}$ the message space and $\mathcal{C}$ the ciphertext space. Set $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^n$, i.e., the set of all bit strings of length n.

Then the Vernam OPT encryption scheme $\mathcal{E}$ =(**Gen, Enc, Dec**) is realised as follows:

- **Gen:** A key $k \in_R \mathcal{K}$ is chosen uniformly at random.
- **Enc:** The encryption c of a message $m \in \mathcal{M}$ with given key $k \in \mathcal{K}$ is accomplished by bitwise exclusive or (XOR, denoted by $\oplus$), i.e., $c := m \oplus k$.
- **Dec:** Given the ciphertext $c \in \mathcal{C}$ and key $k \in \mathcal{K}$, the original message m is retrieved by $m := k \oplus c$.
- **Validity:** It holds that $k \oplus c = k \oplus (k \oplus m) = m$.

Since $|\mathcal{K}| = |\mathcal{M}|$, the one-time pad is already optimal with regard to key length and we need to look no further in our considerations for a perfectly secret encryption scheme. This leaves us with the issue of key agreement and management which we will now discuss.

4.2.4 Information-theoretic Secure Key Agreement

The topic of key agreement protocols is an interesting one and has been studied extensively as it is in some sense the basis of private key cryptography: let Alice and Bob be the two parties who wish to communicate over an insecure channel using private key cryptography (e.g. OTPs) in the presence of an adversary Eve. Before either of them can send an encrypted message, they have to assure that they are both in the possession of the same private key k. We are of course interested in how Alice and Bob can agree on a shared private key in an information-theoretic secure manner. The standard approaches to this problem are the noisy channel model, the Bounded Storage and Limited Access model, as well as quantum key distribution. For an extensive treatment of these models, we refer to the survey on long-term confidentiality by Braun [5].

We will describe each of these briefly and analyse them according to their suitability for G^2_{its} VSR where the smooth execution of the protocol requires each pair of nodes to exchange a large amount of key material. Furthermore, we will consider other means of key distribution between two (or more) communication partners.

Noisy Channel Model Noise is a natural occurrence in every physical communication channel and induces errors on these channels. The fact that the different channels of Alice, Bob and Eve experience different noise levels, i.e., error probabilities, can then be used to create a secret key by public interaction between Alice and Bob from a given a random publicly broadcast bit string.

One of the first approaches was Wyner's Wire-Tap Channel [33], but this protocol required the adversary Eve to receive a degraded version of the transmitted signal -

an assumption which cannot be made realistically. Csiszar and Körner [10] relaxed this assumption and required Eve's channel to be noisier than that of Alice and Bob. Generally, a significant drawback of key agreement protocols based on noisy channels is that it is unlikely to have information about the error probability on the adversary's channel. Especially protocols which assume the inferiority of the adversary's channel fail to model sensible scenarios.

A more realistic approach was described by Maurer [19] who allowed the adversary's channel to be even less noisy than that of legitimate participants. This came at the cost of the further requirement that the legitimate participants have access to a secondary, noiseless and authenticated public channel. Let R denote the random publicly broadcast bit string. Since Alice, Bob and Eve have different noisy channels with different error probabilities they all receive a different version of R: Alice receives X, Bob receives Y and Eve receives Z.

The goal of Alice and Bob is now to to derive a secure key from their received signals X and Y. In order to be able to do so they engage in a protocol consisting of the following phases:

1. *Advantage Distillation*: Alice and Bob use the aforementioned secondary channel to share information about their received signals X and Y and thereby attain an advantage over Eve.

2. *Information Reconciliation*: This advantage is used by them to agree on a string S which is the same for Alice and Bob with high probability.

3. *Privacy Amplification*: This string is then modified into the secret key.

All of the above mentioned approaches assume a passive adversary. Active adversaries, as are present in our case, have been considered for example in [20], [24] and [34].

The Bounded Storage Model The Bounded Storage model was first introduced by Maurer [18]. As the name indicates, it rests on the assumption that the adversary Eve has limited storage capacity, say, s. Furthermore, it is assumed that Alice and Bob shared a short secret key k in advance and that there exists a very large source of randomness R which is publicly available and $|R| > s$, i.e., it can only be partially stored by Eve. Suitable examples for R would be a satellite broadcast, a deep space radio source or a pulsar. Alice and Bob use their pre-shared key k to determine which part r of R to access. They then can expand k to a much longer key $x = f(r, k)$ by applying a key expansion function f. During this, Eve is allowed to read all of

R and store some value $h(R)$ with $h(R) \leq s$. Naturally, each signal from R is lost permanently after transmission, unless it was stored.

There were quite few improvements made to the original proposal e.g. by [7], [2] and [1], but still the crucial assumption that the adversary's storage is significantly bounded has become somewhat unrealistic with ever dropping storage costs. Moreover, a truly practical realisation of R proves problematic. Not only must each participant have the adequate equipment to listen to signals coming from e.g. deep space radio sources but it also takes a considerable amount of time to receive the needed volume of signals.

The G^2_{its} protocol relies on the fact that each two participating node have exchanged secret keys and due to the usage of OTPs this amounts to a lot of secret key material. Furthermore, it is not reasonable to assume that each pair of participants in the protocol (esp. during the redistribution phase with the introduction of new nodes in access structure $[n', m']$) have a pre-shared secret k.

Limited Access Model First proposed by Rabin [23] the Limited Access model follows a similar idea to the Bounded Storage model. The scheme is based on the existence of a large network of so-called page server nodes. Alice and Bob have again pre-shared a short secret key k which will specify a subset of pages from a selection of page server nodes. Alice and Bob will then access these pages and download the respective bit strings. The crucial assumption of the protocol is that Eve is not able to monitor all page server nodes and all strings downloaded by Alice and Bob.

But as in the Bounded Storage model the assumptions about Eve's capabilities are not justifiable and therefore it is not suitable as a key agreement protocol for our scenario.

Quantum Key Distribution A completely different approach is given by quantum key distribution which relies on the transmission of qubits rather than classical bits. The well-known BB84 protocol by Bennet and Brassard [3] was the very first quantum cryptographic protocol. It achieves secret key agreement by sending polarized photons over fibre-optic cables from one legitimate participant to the other. Another approach, the E91 protocol [13], was described by Ekert in 1991 and is based on the possibility of quantum entanglement. The protection from an eavesdropping adversary is in both protocols given by the so-called *observer effect* which states that certain systems cannot be measured without altering the state of the system. Before measurement qubits exist in a *superposition* of their two base states, commonly denoted in Dirac notation by $|0\rangle$ and $|1\rangle$. Once they are measured they are well-defined in one of the two possible states. For example if Eve eavesdropped on

the transmission of polarized photons from Alice to Bob, the superposition of the quantum states would collapse and Bob could see that the information had been accessed before it reached him and was therefore likely compromised. Analogously, in entangled particle systems the measurement of one of the two particles also yields a change in the (nonlocal) opposite particle. Unfortunately, these very promising ideas are still impractical for widespread use since they require dedicated and expensive hardware in order to be able to generate and measure the needed qubits and therefore scale badly. Additionally, the transmission of e.g. polarized photons is only possible via fibre-optic cables between any two participants and there exist serious limits on the distances that can be bridged.

"Offline" Key Exchange As we have seen, existing information-theoretic secure key agreement protocols have their flaws concerning the underlying assumptions or practicability of the protocols. In our case we have to ensure information-theoretically secure key agreement between nodes which belong to the same company or at least collaborating companies. This implies, that it is feasible to achieve key exchange by e.g. trusted couriers. This enables the exchange of a sufficiently large amount of OTP keys to go through multiple iterations of the protocol on a storage medium such as a thumb drive or disk.

4.3 Summary

At this point, we want to briefly summarize our findings so far: We examined the long-term secure aspects of the verifiable secret sharing protocol by Gupta and Gopinath and found that some substantiations were necessary. We saw that the used Pedersen commitments in themselves are not sufficient for long-term secure integrity of the committed value and need to be prolonged regularly. The required private point-to-point communications are achieved by symmetric OTP encryption, which provide perfect secrecy and are optimal with respect to key length. The issue of key agreement or exchange is a complex one in the information-theoretic secure setting and a conclusive best choice cannot be made and is highly dependent on the concrete scenario. Due to our special environment, which will be described in detail in the following Chapter 5, the participants in the key agreement/ exchange process are servers belonging to the same company and thus we decided on the direct generation of keys and exchange in person via a qualified trusted courier, e.g. an employee. We are aware that this solution is not feasible for a lot of applications and one of the other presented methods might be worth further considerations. Commonly

used key agreement protocols today are based on public key cryptography and do not provide information-theoretic security which is necessary to guarantee the long-term secure storage of highly sensitive data. Nevertheless, public key signatures can be used to ensure authenticity of sent messages since it is only relevant for the short time interval in which these messages are processed. On a side note, we clarified the usage of the term 'broadcast' and mentioned the lack of reliable broadcast or multicast protocols leading to the emulation through unicasts. Nevertheless, we will keep the phrase 'broadcast' in the following in accordance with the G^2_{its} protocol.

5 The Enterprise Scenario

In the following we wish to present the setup in which the EPSS scheme (described in Chapter 6) is situated. In order to gain efficiency and a more realistic model for use case environments, we propose to relax the commonly made assumption that any two parties within the protocol must be able to communicate via a direct private communication link. This assumption, which is also found in the protocol by Gupta and Gopinath [16] (cf. 3.1.2), demands that e.g. during redistribution from an $[n, m]$ access structure to an $[n', m']$ access structure up to $\binom{n+n'}{2}$ private channels need to be established[2]. A PSS scheme wishing to achieve long-term security requires these channels to be information-theoretically secure. Obviously, as detailed in Chapter 4, this is not an option for most use cases as establishing information-theoretically secure channels is still costly and limited to special cases (e.g. QKD) or arduous (e.g. offline OTP key material exchange). It was found that nevertheless this requirement is most realistically achieved by securely exchanging OTP key material on e.g. hard disks.

5.1 Scenario Description

Specifically, we have in mind the application of PSS within a company with e.g. different locations or departments where it is infeasible for *all* shareholders to securely exchange secret key material (e.g. due to geographical distance) and thus, to establish a direct information-theoretically secure communication link. Therefore, we assume that there exist clusters of shareholders (e.g. all shareholders grouped by their company location) and secure channels are established only within clusters and across clusters.

This will in the following be referred to as the *enterprise scenario* as it is a natural structure found within enterprise networks. Enterprises above all have a vested interest in the long-term confidential storage of some of their data, be it a secret company recipe or signature keys and are therefore prime use cases for practical PSS schemes.

Remark 5.1. We wish to stress that the scenario presented here is not to be confused with use cases of what is called Hierarchical Secret Sharing (e.g. by Tassa [28]). In Hierarchical Secret Sharing, a hierarchy among the shareholders is introduced with respect to different degrees of *authorization*. Some shareholders receive

[2]This corresponds to the number of edges in the complete graph $K_{n+n'}$, i.e., in an undirected graph in which any two of the $n + n'$ vertices are connected by an edge.

in a sense 'more' reconstruction power than others such that fewer of these more powerful shareholders are needed to establish an authorized subset than of shareholders of a lower level. In our scenario, all shareholders enjoy equally potent shares and are grouped by their position within the actual network.

5.2 Terminology and General Assumptions

5.2.1 Shareholder Structure

Client and Shareholders The nodes which establish communications across clusters will be referred to as *roots*, while the other nodes are referred to as *child* nodes. Any node belongs to a single cluster. We assume that a client that wants to share a secret has the means to communicate securely, e.g. via an OTP-encrypted channel, with at least a single root node in the access structure designated for initial distribution. This root node is then responsible for distributing the received data to all other root nodes. However, for simplicity, we assume that the client has a direct communication link to *every* root node within the access structure. Furthermore, the client is not assumed to be a shareholder. Again, this is easy enough to implement in any given PSS scheme, but for sake of simplicity we demand the strict separation of these two kinds of nodes. To guarantee the smooth execution of the PSS protocols, it is assumed that sufficiently many OTPs were exchanged in advance between all the root nodes and the client, as well as between the root node and child nodes within a cluster. The same is true for the respective nodes before the redistribution process.

Notation In the following, let the client be denoted by C and let n be the total number of participating nodes with ids $1, 2, \ldots, n$ and let m define the threshold of the secret sharing scheme. To accommodate the additional parameter N which will denote the number of root nodes within an access structure, access structures will in the following be denoted by $[n, m, N], [n', m', N']$, and so forth. The root nodes are denoted in capital letters to distinguish their id's from the child nodes and the cluster of root node I will commonly be referred to as $\mathcal{N}_I$.

Parameter Choice As usual, the threshold m is derived from the assessment as to how many servers are estimated to be compromised within a single time period (maximally $m-1$). Following [16], we assume $n = 2m-1$ for the moment. Obviously, it is necessary to choose N such that $N << n$ to gain the maximal advantage in terms of communication complexity, as otherwise the number of secure communication

channels between root nodes increases and the partition is close to the original unclustered setup. Furthermore, it is advisable that clusters be of roughly the same size to avoid inbalances in the conceived worth (or vulnerability) of these clusters to potential adversaries.

5.2.2 Network

Authentication Following the assumptions made in many PSS schemes, we assume that all messages are authenticated by the sending party using a signature scheme that is considered secure at the time of message generation and transmission. It is not feasible to spoof the identity of a sender within the network.

Reliability The underlying network is considered reliable in the sense that all sent messages cannot be arbitrarily delayed and are eventually delivered (within a reasonable upper bound on message delay).

The Broadcast Channel Another means of communication between nodes participating in the PSS scheme, is the broadcast channel. It is - just like the private channels - assumed to be reliable. There are many ways to implement broadcast channels, but in the following we will use the example of a public bulletin board due to its nice illustrative properties. Any nodes will have read and write access to this bulletin board. Nodes cannot at any point be prohibited from reading or writing to the bulletin board and the bulletin board cannot be cleared. I.e., any message posted to the bulletin board will be available on the board indefinitely. The posts on the board are authenticated by signatures which implies that no node can post on behalf of another node.

5.2.3 Adversary

The assumed model of the adversary is the same as in most PSS schemes, namely the one of a mobile and active adversary as described previously in Section 3.1.2. We will not differentiate between adversary-induced malicious behaviour of nodes and regular failures (e.g. hardware failures). Both cases will be referred to as compromised, dishonest or malicious nodes. We assume all compromised nodes to act in collusion. Furthermore, we assume that an adversary can be removed from a compromised node by a reboot procedure.

5.3 A First Approach

5.3.1 Shortcomings

As a first naive approach one might try to simply use an existing PSS protocol such as the one by Gupta and Gopinath [16] on top of the newly created shareholder structure in which the shareholders are clustered into groups. As mentioned before, information-theoretically secure communications across clusters can only be established through a distinguished shareholder within each cluster. Messages are then no longer sent via private point-to-point communication channels between any two nodes but by directing them along the given communication paths within and between the clusters. This approach, however, leads to substantial problems:

- Root nodes learn all (sub-)shares that are routed through them during initial distribution, redistribution and reconstruction. This presents a serious threat to the confidentiality of the shared secret k.
- Root nodes can effectively cut off their cluster from all communications, endangering the availability of the secret k.
- Since the communication links involved in the (re)distribution of the secret are no longer point-to-point, the resolution of complaints becomes more complex. The complaint mechanism applied in the G^2_{its} does no longer suffice to guarantee a swift complaint handling while retaining confidentiality, integrity and availability of the secret k.

From this it is clear that several changes need to made to existing PSS schemes such as G^2_{its} protocol to accommodate the changes presented by the clustered scenario.

5.3.2 Key Ideas

In this section, we will first describe the key ideas that went into into the development of the EPSS scheme which ease the above mentioned shortcomings before presenting the detailed initial distribution, redistribution and reconstruction protocols in Chapter 6. A security analysis will follow in Chapter 7.

Separation of Roots and Shareholders The threat to confidentiality can be met by various measures: For once, root nodes should not be part of the access structure, i.e., they should not act as shareholders. This can first and foremost be justified by their inherent difference to simple shareholding nodes. The root node's primary

function in the clustered scenario is to enable secure communication flows across clusters and between shareholders and client. As such vital parts of the network infrastructure used by the PSS protocol, these nodes are granted stronger security and stability assertions and have access to more computational resources than simple shareholders. While certain root nodes may become expendable (because their cluster is entirely removed) and some root nodes may be replaced over time, no marked fluctuations are expected from one time period to the next in stark contrast to the interchangeability of the child nodes which have the sole purpose of storing shares and redistributing them. Therefore, it is reasonable to exclude root nodes from the dynamic changes in the access structures across share redistribution processes. This also goes in accordance with existing PSS protocols since they too abstract the shareholders and clients from the (necessary) rest of the network infrastructure such as e.g. switches. This new situation is depicted in Figur 1.

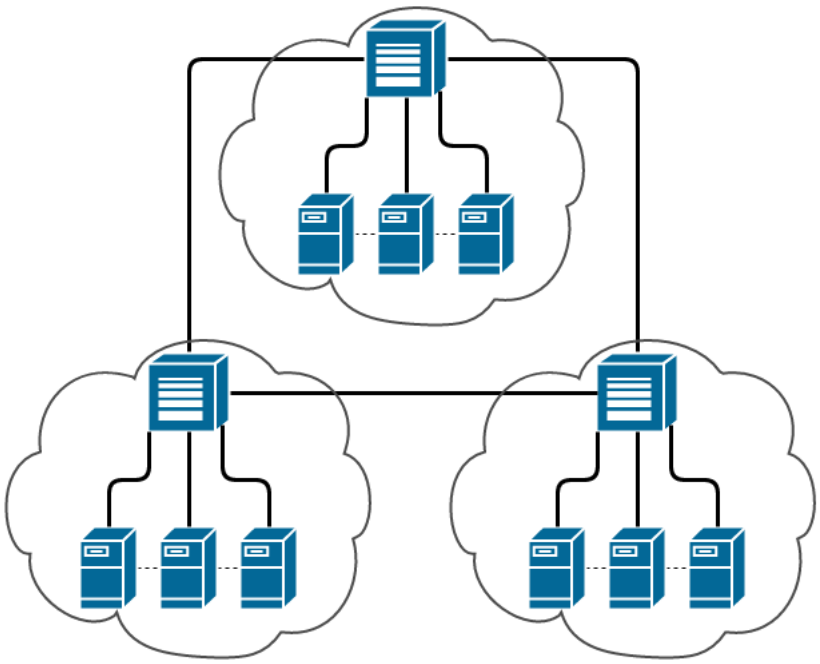

Figure 1: Clustered Scenario with Roots and Shareholders separated in Functionality.

Encryption of (Sub-)Shares The above presented measure alone, however, is not enough to strengthen the confidentiality within the new structure. A further step is to encrypt all (sub-)shares with the public key of the receiver, which eliminates the immediate risk of exposing the secret to compromised root nodes or during complaint distribution on the bulletin board. Even if the threshold number of shares would be

exposed to compromised colluding (root) nodes, the secret cannot be reconstructed as long as the underlying computational assumption of the encryption scheme is not broken. Since a signature scheme is already put into place it is not unnatural to necessitate the maintenance of a public key pair by all involved nodes. Note, that this encryption scheme does not replace the information-theoretically secure channels. The public-key-encrypted shares are still secured for transport with OTP-encryption. The advantage of the computationally secure encryption is to not expose (sub-)shares as plain text to root nodes which receive and transmit them.

Data Aggregation Recall that the overall motivation for the clustered scenario was to present an efficient PSS solution. The main weakness of current PSS schemes is their high communication complexity, i.e., the amount of data sent by honest parties during an average run of the protocol. Although the clustered scenario offers fewer communication channels, the same information needs to be transmitted in order to guarantee the same functionality. To reduce load on the network, data should therefore be aggregated whenever possible before being further distributed. During the redistribution process for example, the data needs to be transmitted from the sending shareholders to the receiving shareholders via their respective root nodes. Similarly, the reconstruction has the shareholder's respective roots as a relay to the client. Therefore, the data aggregation will take place within root nodes. The root nodes on the sender side will combine the sub-shares they received from their children to one partial share per receiver node j. These will then be transmitted to the respective receiver nodes which will, finally, compute the new share for each of their children j.

Homomorphic Encryption Since root nodes process encrypted (sub-)shares, it is necessary that the public-key scheme allows for computations on the encrypted data in order to achieve the aggregation of data. As we will see in 6, an additively homomorphic encryption scheme, further denoted by $\mathcal{E}$, will be sufficient for the task. While there is a slight increase to the data being sent due to ciphertext expansion, the combination of this measure with the data aggregation will eventually decrease the load on the network as opposed to a solution within the clustered scenario without either of the measures. Since over the lifetime of the shared secret, many encryption schemes will have the assumed property, we treat the homomorphic public key encryption scheme as a black box and only require the additive homomorphic property as well as the security at the time of encryption and transmission.

Complaint Resolution As the traceability of complaints becomes more complex with each additional involved party that is potentially misbehaving, the information disclosed on the broadcast channel when trying to resolve these complaints increases and presents a further threat to the confidentiality of the secret. Furthermore, in the absence of direct point-to-point communication channels between any two nodes, the possibility of (honest) nodes being involuntarily cut off from all communications by dishonest nodes along the communication paths, is an additional risk that must be detected and handled. Solutions to both of these problems can be found in asynchronous proactive secret sharing schemes as briefly presented in 2.3.2. Asynchronous schemes have to deal with arbitrary long message delays and therefore must supply their PSS schemes with a mechanism to finish in the presence of unresponsive shareholder and are therefore a solid basis to build a complaint resolution extension upon.

6 The Enterprise PSS Scheme (EPSS)

In the following chapter, we will use the situation and ideas described in the previous Chapter 5 as a foundation for the introduction of the Enterprise PSS Scheme (EPSS).

6.1 The EPSS Scheme

After presenting the EPSS protocol for the initial distribution of the secret, a basic EPSS redistribution protocol which treats the case where all participants are honest is presented. This is done to show the general idea behind the modification to the redistribution process and to give a starting point for further considerations with malicious entities involved. With malicious entities present, the protocol execution will make it necessary to establish consensus among the honest participating nodes. The mechanisms to achieve consensus are in parts inspired by asynchronous PSS schemes (cf. 2.3.2) which rely on Byzantine agreement. It is well known that Byzantine agreement can only be achieved optimally in the presence of less than $\lfloor \frac{n}{3} \rfloor$ faults among n shareholders. Furthermore, these solutions often call for a dedicated node Coord which acts as a coordinator to the redistribution process. The coordinator will be selected among the root nodes in a deterministic round-robin fashion such that an adversary cannot influence the choice. We choose a root node to take the role of the coordinator as they are assumed to have both higher computation capabilities than shareholders and they are able to communicate securely across access structures. The coordinator can be chosen among the sender as well as the receiver roots, as no profound fluctuations among the root nodes is expected in the ordinary case. The participating nodes can usually request for a new coordinator if they suspect misbehaviour. The chapter will be closed with the EPSS reconstruction protocol as well as some small-scale considerations in case of a dishonest coordinator.

Notation Any m-subset $\mathcal{B}_u$ can be written as the unique union of pairwise disjoint sets $\mathcal{N}_1, \mathcal{N}_2 \ldots, \mathcal{N}_N$ where $\mathcal{N}_I$ describes the set of all nodes in $\mathcal{B}_u$ that belong to the cluster of root node I, $1 \leq I \leq N$. Some $\mathcal{N}_I$ may be the empty set. If in the following, the notation $\mathcal{N}_I \in \mathcal{B}_u$ is used, only non-empty sets $\mathcal{N}_I$ are considered.

6.1.1 EPSS Initial Distribution Protocol

Let the client C be trusted with the task of sharing the secret k. The client C takes the role of the coordinator and handles the initial distribution of the secret k.

Should the initial distribution process not be able to complete within a reasonable time frame or should any discrepancies be detected by the participating nodes which monitor the client that indicate misdemeanour, the system management will be alerted and further investigations are triggered. Therefore a dishonest client can be assumed to be always detected. Such discrepancies can include for example abortion of the process although the broadcast channel has seen sufficiently many finished messages.

1. The client C which wants to share its secret k, initializes the secret-sharing process by broadcasting an $\mathsf{init}_{\mathsf{Distr}}$ message which contains all chosen shareholders $i \in [n, m, N]$.

2. The client C then picks coefficients a_l and b_l, $l = 1, \ldots, m-1$ to form the polynomials $a(x) = k + \sum_{l=1}^{m-1} a_l x^l$ and $b(x) = t + \sum_{l=1}^{m-1} b_l x^l$. The shares $s_i := a(i)$ of k and $t_i := b(i)$ of t can now be computed.

3. The client C encrypts the shares s_i and t_i for node i using the homomorphic encryption resulting in the encrypted share pair $(\mathcal{E}_i(s_i), \mathcal{E}_i(t_i))$ which is then signed by the client.

4. Subsequently, the encrypted and signed share pairs for each of the N clusters are gathered and each collection is OTP-encrypted for the respective root node. The shares are thereby secured for transport and can be sent to their appropriate root node.

5. To share the secret in an information-theoretic secure way, the client uses generators g and h to compute the commitments

$$g^k h^t, g^{a_1} h^{b_1}, \ldots, g^{a_{m-1}} h^{b_{m-1}}$$

   which are then published in an authenticated manner on the bulletin board.

6. Upon receiving the share pairs, the root nodes verify the client's signature as well as the format of the message (i.e., the correct number of shares of the correct size are contained).

   - Those shares which carry a valid signature of the client and are well-formatted are then forwarded to the respective child node in an information-theoretic secure fashion.
   - The other ones are discarded and no further action is taken by the root nodes.

7. Upon receiving the share from its root nodes, each child i verifies the client C's signature on the share pair, decrypts them and checks that

$$g^{s_i} h^{t_i} \equiv g^k h^t \prod_{l=1}^{m'-1} \left(g^{a_l} h^{b_l}\right)^{i^l} \tag{10}$$

holds.

   - If the verifications are successful, the child i saves (s_i, t_i) as its share pair and signals the successful storage with a $\mathsf{finished}_{\mathsf{Sig_i}}$ message on the bulletin board. Shareholders i handle invalid signatures gracefully if and only if the verification equation (10) is satisfied, in that they accept the share pair nevertheless.
   - If the verification equation (10) fails, the child node i does not store the received share pair and broadcasts a $\mathsf{reject}_{\mathsf{Sig_i}}$ message to indicate the failure to store a valid share pair.

8. After a pre-defined time frame, the client C checks if there are at least $2m-1$ distinct and authentic $\mathsf{finished}$ message present.

   - If so, the client announces the successful initial distribution of the secret by an authenticated $\mathsf{done}_{\mathsf{Distr}}$ broadcast. All involved nodes securely delete any internal data used in the distribution process such that the only stored data are the distributed share pairs stored with the shareholders.
   - If less than $2m-1$ shareholders in $[n, m, N]$ reported the successful storage of the shares, the initial distribution failed. The client C then broadcasts an $\mathsf{abort}_{\mathsf{Distr}}$ message. Upon seeing an $\mathsf{abort}_{\mathsf{Distr}}$ message, all involved nodes securely delete any data used in the distribution. The client then re-initiates the initial distribution process.

Remark 6.1. For the initial distribution it is not wishful to carry out a complaint resolution protocol (as later described for the redistribution process) since this always leads inadvertently to the leakage of information about the secret. Since the process of initial distribution can be executed efficiently due to its shortness, it is reasonable to re-start the process with varying shareholders until it terminates successfully. Should the process not be able to complete within a reasonable time frame, further investigations into the client (and - if there exist indications -the involved root nodes) are necessary as indicated above. A rebooting procedure or the replacement of suspiciously behaving nodes (especially root nodes or the client) will ultimately resolve the issue.

6.1.2 Basic Redistribution Protocol

General Idea

The basic EPSS redistribution protocol works as follows: a distinguished root node - in the following referred to as Coord - will be chosen to direct the activities of the redistribution process. In particular, this node will be responsible for the selection of distributing nodes in the current access structure $[n, m, N]$ and will also determine whether the redistribution to the new access structure $[n', m', N']$ terminated successfully. After initiating the redistribution process, Coord will identify an m-subset $\mathcal{B}_{\tilde{u}}$ of $[n, m, N]$ such that the stored shares of members of $\mathcal{B}_{\tilde{u}}$ are consistent with the original secret. These so-called sender nodes will then create an encrypted sub-sharing of their stored shares and forward the result to their root nodes. Upon receiving the sub-sharings, the root nodes will aggregate the received data into encrypted *partial share pairs* for each $j \in [n', m', N']$ and transmit these to the respective receiver root nodes. Here, the data is further aggregated by computing the new (still encrypted) share pairs for each $j \in [n', m', N']$ from the received partial shares. Each j then stores its new share pair after decryption and verification and signals the successful storage by a broadcast finished message. The coordinator Coord then collects these messages and declares the redistribution to be completed successfully once m' distinct messages have been broadcast.

Basic EPSS Redistribution

1. A coordinator Coord is chosen among the root nodes.

2. The coordinator Coord initiates the redistribution process by sending an $\mathsf{init}_{\mathsf{Redist}}$ message over the broadcast medium.

3. All nodes $i \in [n, m, N]$ publish the commitment to their currently stored share pair (s_i, t_i) of the form $g^{s_i} h^{t_i}$ on the bulletin board.

4. The coordinator executes the algorithm $\mathcal{A}$ on $[n, m, N]$ to obtain a sequence of m-subsets $\{\mathcal{B}_u\}_{1 \le u \le L}$ where $L = \binom{n}{m}$. The coordinator then selects the first element in the sequence $\mathcal{B}_1$ to check if all of its members have stored shares which are consistent to the original secret. This can be seen by evaluating if the following equation holds:

$$g^k h^t \equiv \prod_{i \in \mathcal{B}_1} (g^{s_i} h^{t_i})^{b_i} \quad \text{where} \quad b_i = \prod_{\substack{l \in \mathcal{B}_u, \\ l \neq i}} \frac{l}{l - i} \tag{11}$$

(i) If equation (11) holds, the coordinator publishes a $\mathsf{select}_{\mathcal{B}_1}$ message on the bullet board.

(ii) If equation (11) does not hold, the coordinator chooses the second element $\mathcal{B}_2$ in the sequence generated by $\mathcal{A}$ and evaluates the verification for this set. This process repeats until the first $\mathcal{B}_{\tilde{u}}$ is found for which equation (11) holds.[3]

5. Let $\mathcal{B}_{\tilde{u}}$ denote the m-subset that was established in Step 4. All $i \in \mathcal{B}_{\tilde{u}}$ compute their sub-sharings according to the new access structure $[n', m', N']$. Each sub-share is homomorphically encrypted for the respective receiver node $j \in [n', m', N']$, i.e., $(\hat{s}_{ij}, \hat{t}_{ij})$ is encrypted to $(\mathcal{E}_j(\hat{s}_{ij}), \mathcal{E}_j(\hat{t}_{ij}))$. The ciphers are then signed individually and transmitted securely to the sender's root node via the OTP-encrypted channel.

6. Each sender node i publishes the commitments

$$g^{a'_{i1}} h^{b'_{i1}}, g^{a'_{i2}} h^{b'_{i2}}, \ldots, g^{a'_{i(m'-1)}} h^{b'_{i(m'-1)}}$$

to the coefficients of its chosen sharing polynomials on the bulletin board in an authenticated manner.

7. Each root I collects all sub-sharings by $i \in \mathcal{N}_I$ and verifies the sender's signatures. It then aggregates the received data by computing the encrypted partial share pair for receiver node j given by:

$$\begin{aligned} \mathcal{E}_j(s_j^{\mathcal{N}_I}) &= \sum_{i \in \mathcal{N}_I} b_i \cdot \mathcal{E}_j(\hat{s}_{ij}) \\ \mathcal{E}_j(t_j^{\mathcal{N}_I}) &= \sum_{i \in \mathcal{N}_I} b_i \cdot \mathcal{E}_j(\hat{t}_{ij}) \end{aligned}$$

where

$$b_i = \prod_{\substack{l \in \mathcal{N}_I, \\ l \neq i}} \frac{l}{l - i}.$$

Each partial share pair $(\mathcal{E}_j(s_j^{\mathcal{N}_I}), \mathcal{E}_j(t_j^{\mathcal{N}_I}))$, $j = 1 \ldots n'$, is then signed and transmitted via the OTP-encrypted channel to the appropriate cluster root node J to which j belongs.

[3] Note, that such a $\mathcal{B}_{\tilde{u}}$ does always exist even in the presence of up to $m - 1$ dishonest nodes in $[n, m, N]$ (considered later in this chapter).

8. The receiving roots verify the sender root's signature and compute the new share pair $(\mathcal{E}_j(s'_j), \mathcal{E}_j(t'_j))$ for child node j by

$$\begin{aligned}\mathcal{E}_j(s'_j) &= \sum_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} \mathcal{E}_j(s_j^{\mathcal{N}_I}) \\ \mathcal{E}_j(t'_j) &= \sum_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} \mathcal{E}_j(t_j^{\mathcal{N}_I}).\end{aligned}$$

   After signing each share pair individually, the new shares are transmitted OTP-encrypted to the respective child node.

9. The child node verifies the signature of its root node on the received share pair. It then decrypts the share pair and verifies if

$$g^{s'_j} h^{t'_j} = \prod_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} \prod_{i \in \mathcal{N}_I} \left(g^{s_i} h^{t_i} \prod_{l=1}^{m'-1} (g^{a'_{il}} h^{b'_{il}})^{j^l} \right)^{b_i} \tag{12}$$

   holds. If this is the case, node $j \in [n', m', N']$ stores (s'_j, t'_j) as its share pair.

10. After the successful decryption, verification and storage of the new share pair, node j publishes a signed $\mathsf{finished}_{\mathsf{Sig_j}}$ message on the bulletin board. The coordinator Coord announces the successful termination of the redistribution process by broadcasting a $\mathsf{done}_{\mathsf{Redist}}$ message once he verified the existence of m' correctly signed, distinct $\mathsf{finished}$ messages which ensures that at least m' valid shares have been redistributed. Note, that m' messages already guarantee the success of the redistribution protocol since all nodes are assumed to be honest.

11. Upon receiving the $\mathsf{done}_{\mathsf{Redist}}$ message, all involved nodes securely delete any data used in the redistribution process such that the only stored data is the new shares within the new access structure.

Proof of Correctness of Eq. (12).

$$\begin{aligned} g^{s'_j} h^{t'_j} &= g^{\sum_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} s_j^{\mathcal{N}_I}} h^{\sum_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} t_j^{\mathcal{N}_I}} \\ &\equiv \prod_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} g^{s_j^{\mathcal{N}_I}} h^{t_j^{\mathcal{N}_I}} \\ &\equiv \prod_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} g^{\sum_{i \in \mathcal{N}_I} b_i \hat{s}_{ij}} h^{\sum_{i \in \mathcal{N}_I} b_i \hat{s}_{ij}} \\ &\equiv \prod_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} \prod_{i \in \mathcal{N}_I} (g^{\hat{s}_{ij}} h^{\hat{t}_{ij}})^{b_i} \\ &\equiv \prod_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} \prod_{i \in \mathcal{N}_I} \left(g^{s_i} h^{t_i} \prod_{l=1}^{m'-1} (g^{a'_{il}} h^{b'_{il}})^{j^l} \right)^{b_i} \end{aligned}$$

□

6.1.3 Extension to Basic Redistribution Protocol: Average Case Complaint Resolution

In the following section we will investigate what we consider to be the average case in the enterprise setting: we allow both old and new shareholders to contain compromised nodes whose maximal number is bounded by the respective threshold (i.e., at most $m-1$ compromised nodes in $[n, m, N]$ and $m'-1$ in $[n', m', N']$). On the level of root nodes, no compromise is to be expected. As mentioned beforehand, this is a reasonable assumption as it goes in accordance with previous PSS protocols which only consider compromise among the shareholders, but not within the network infrastructure (e.g. switches). Furthermore, enterprise networks are closely monitored, especially machines which are crucial to the smooth execution of operations which include the root nodes in the EPSS scheme. Nevertheless, the possibility of misbehaving root nodes (if only through e.g. hardware failures) exists and the according extension to the EPSS protocol will be treated in Section 6.1.4.

While we assumed that it is not possible for an attacker to spoof the identity of another node within the access structure, this can only be ensured by careful signature checks. Should a root node detect an invalid signature, the message is discarded. If after a certain amount of time, the root node has not received all messages necessary for the continuation of the protocol, an authenticated cannotResume message is broadcast, listing those nodes which were unresponsive. The coordinator Coord can then re-initiate the protocol taking into account the information gained from these messages about unresponsive nodes within the network.

Preparatory Measures

The redistribution is established along three private channels:

1. from the redistributing sender nodes to their root nodes,
2. from these root nodes to the root nodes of the receiving access structure, and finally,
3. from there to the receiver nodes $[n', m', N']$.

After each transmission, varying verifications are done to be able to detect discrepancies as early as possible. In the following we will describe which discrepancies can arise and how they are dealt with in the presence of dishonest shareholders.

Transmission 1: Sender Children to Sender Roots Malicious sender children in $\mathcal{B}_{\tilde{u}}$ can either send incorrect messages to the roots or refrain from participating in the protocol execution. Incorrect messages can have various forms. The message can carry an invalid signature and/or can be malformed in the sense that it does not contain n' encrypted sub-shares of the correct sizing. If one of these cases is detected by a root node I, it will discard the messages in question and wait for a pre-established amount of time. After the timeout, the root node will list all its children from which it has not yet received a correctly signed and formed message (thus including completely unresponsive nodes). This list is then published in an authenticated $\mathsf{cannotResume}_{\mathsf{Sig_J}}$ message on the bulletin board.

After collecting all cannotResume messages within a given time frame, the coordinator Coord re-initiates the protocol by choosing the first valid m-subset[4] which contains none of the accused child nodes. If no such messages are recorded, the protocol execution can continue unchanged. We wish to stress here, that a sender root node is not able to detect sub-sharings which are inconsistent with the sender node's stored (and previously verified) share pair.

Transmission 2: Sender Roots to Receiver Roots Since both parties are assumed to be honest and the channels are assumed to be reliable, a receiver root J will eventually receive all necessary partial shares. Should a message with an invalid signature be received, the message is simply discarded and no further action is taken.

Transmission 3: Receiver Roots to Receiver Children After computing the new encrypted share pairs, the receiver roots forward these values signed to their children. Should a child node receive a share pair which carries an invalid signature, it simply discards the share pair and waits for the correct share pair to arrive (which is guaranteed again by the reliability of the channels and the honesty of the root nodes). Each child node in $[n', m', N']$ then decrypts and verifies the received share pair. If some child node j detects a discrepancy in the verification equation (12), this event is tracked by node j broadcasting an authenticated $\mathsf{reject}_{\mathsf{Sig_j}}$ message. This message indicates to all other nodes and in particular the coordinator, that node j is not happy to store the received and decrypted share pair (s'_j, t'_j) because it confirmed that the value is inconsistent with the published commitments. Of course, the lodged complaint might also be unwarranted due to a dishonest node j.

[4] I.e., all of its members have stored shares which are consistent with the original secret.

The general strategy is then to collect the reporting of discrepancies. The resolution of complaints is only initiated by the coordinator through an $\mathsf{init}_{\mathsf{Resol}}$ message if less than $2m' - 1$ new shareholders were able to complete the storage of their new shares after a reasonable amount of time. If $2m' - 1$ distinct finished messages are recorded, the necessary lower bound of m' valid new shares is attained even in the presence of $m' - 1$ malicious receiver nodes which may report an unwarranted successful storage of new shares. Therefore, this is a reasonable approach with respect to efficiency, as the ultimate goal of the redistribution process is to establish a minimum of m' valid new shares and the process of identifying dishonest parties is communication intensive and potentially threatening to the confidentiality of the secret and should thus to be avoided whenever possible.

General Idea

In the following we will describe the resolution of complaints which is initiated by the coordinator Coord if it was not able to record $2m' - 1$ authenticated finished messages from different receiver nodes. The processing of the complaints is done by choosing a single complaint at random. As we will later see in the Security Analysis in Chapter 7, only a single recorded complaint should be resolved in the indicated fashion before a re-initiation of the redistribution protocol must be enforced. This is due to the fact, that a resolution of more complaints could lead to the exposure of the secret k. The goal of the resolution is to pinpoint the dishonest parties among the sender nodes in $\mathcal{B}_{\tilde{u}}$ and potentially dishonest receiver nodes, by gradually revealing information and using an existing honest majority among a set of nodes which will be further referred to as the Jury. The Jury can consist of any set of entities that has access to the broadcast channel and can guarantee an honest majority among its members. In our present case, the Jury can most easily be chosen as the set of either all receiver or sender root nodes. In principle, this eliminates the need for majority vote to find consensus among the members of the Jury in the average caes, as all root nodes are assumed to be honest and will therefore always reach the same conclusions. Nevertheless, we will describe the protocol with the indication as to where majority votes are necessary. Note, that it is also possible to choose the Jury as either the old or new shareholders since both contain an honest majority by assumption. It is nevertheless more reasonable to select a Jury of root nodes over one of shareholders since they are generally fewer and therefore less parties are involved in the process. Furthermore, as mentioned before, root nodes have in general more computation power at their disposal which additionally aids the execution speed (and therefore overall efficiency).

Extension: EPSS Complaint Resolution Protocol (Avg. Case)

1. The coordinator Coord initiates the complaint resolution protocol by broadcasting an $\mathsf{init}_{\mathsf{Resolv}}$ message.

2. Coord then selects one of the recorded complaints at random, say $\mathsf{reject}_{\mathsf{Sig_j}}$ by node j, and asks the complaining node to publish its received encrypted share pair $(\mathcal{E}_j(s'_j), \mathcal{E}_j(t'_j))_{\mathsf{Sig_J}}$ carrying the receiver root node J's signature. Furthermore, node j must provide the decrypted share pair (s'_j, t'_j) along with a *proof of correct decryption* on the bulletin board.

3. The Jury can now use this information to execute the following verification checks:

   (i) $(\mathcal{E}_j(s'_j), \mathcal{E}_j(t'_j))_{\mathsf{Sig_J}}$ carries a valid signature of node J.

   (ii) Node j correctly decrypted the share pair $(\mathcal{E}_j(s'_j), \mathcal{E}_j(t'_j))$ to (s'_j, t'_j).

   (iii) Equation (12) is **not** satisfied by (s'_j, t'_j).

   If any of the conditions fail, then node j was acting dishonestly (under the assumption that root nodes are honest). Node j is then marked as **VC** by any member of the Jury that cannot verify all three conditions.

4. A majority vote will now determine if the complaint by j was justified.

   - If a majority of the Jury marked j as **VC** the complaint is disregarded and the protocol is re-initiated excluding j.
   - If the complaint cannot be rejected the resolution protocol proceeds.

5. The coordinator now asks all involved root nodes I (i.e., those with children $i \in \mathcal{B}_{\tilde{u}}$), to reveal the encrypted sub-share pairs $(\mathcal{E}_j(\hat{s}_{ij}), \mathcal{E}_j(\hat{t}_{ij}))_{\mathsf{Sig_i}}$ for node j, which they received from their children i on the bulletin board .

6. Subsequently, node j is asked to decrypt the published sub-share pairs and to then verify their validity by checking if

$$g^{\hat{s}_{ij}} h^{\hat{t}_{ij}} \equiv g^{s_i} h^{t_i} \prod_{l=1}^{m'-1} \left(g^{a'_{il}} h^{b'_{il}}\right)^{j^l} \tag{13}$$

   holds. This allows an honest node j to identify dishonest child nodes without miss.

7. Node j then reports the dishonest nodes in an authenticated manner on the bulletin board along with the decrypted sub-shares and a proof of correct decryption. This ensures that a dishonest j cannot accuse honest sender nodes of cheating without being detected by the Jury.

8. Each Jury member then verifies the following conditions:

   (i) The bulletin board entry was published by node j.

   (ii) Each accused sub-share pair $(\mathcal{E}_j(\hat{s}_{ij}), \mathcal{E}_j(\hat{t}_{ij}))$ carries node i's valid signature.

   (iii) Node j correctly decrypted the sub-share pairs $(\mathcal{E}_j(\hat{s}_{ij}), \mathcal{E}_j(\hat{t}_{ij}))$ to $(\hat{s}_{ij}, \hat{t}_{ij})$.

   (iv) Equation (13) is **not** satisfied.

   If all conditions are fulfilled for a sub-share pair, the respective sender node is marked as **VC** by the member of the Jury which confirmed the verification. If any of the conditions (ii) to (iv) fail for any accused sub-share pair, j is marked as **VC**.

9. A majority vote among the Jury then determines the compromised nodes.

10. The coordinator Coord aborts the current instance of the redistribution protocol by broadcasting an $\mathsf{abort}_{\mathsf{Redist}}$ message. Upon receiving an $\mathsf{abort}_{\mathsf{Redist}}$ message, all involved nodes securely delete any data used in the redistribution and resolution process. The coordinator then re-initiates the process and excludes all identified compromised nodes.

Remark 6.2. It should be noted that not all cheating parties are revealed after an execution of the complaint resolution protocol. This is for once due to the fact that only a single complaint is resolved. Furthermore, a dishonest j can refrain from reporting sub-shares that failed the verification of equation (13).

6.1.4 Extension to Basic Redistribution Protocol: Worst Case Complaint Resolution

We now want to investigate the case which allows for dishonest participants both on the shareholder level (old and new) as well as on the root level. We see that apart from signature and format validations executed by root nodes, the main responsibility for the verification of the redistribution process lies with the receiving nodes in $[n', m', N']$.

General Idea

Now receiver nodes in $[n', m', N']$ must also take into account that their own root node is behaving arbitrarily badly. Since we want to avoid that dishonest receiver root nodes can force new shareholders to reject an otherwise valid share pair and to reveal it in the ensuing complaint resolution, receiver nodes handle invalid signatures by their roots gracefully. More specifically, this means that if a receiver obtains a share pair which verifies correctly, but carries an invalid signature, the receiver will store this correct share pair anyway. This also prohibits dishonest receiver children from discrediting an otherwise correct chain of share computation.

Again, in case of a non-terminated redistribution process, a single complaint will be dealt with during each redistribution process. The complaint which was chosen at random is resolved by tracing the computation of the refused share pair from the receiver to the sender in a step-by-step fashion as now there are also potentially dishonest root nodes involved. This process will be described in the following. But first a few additional assumptions need to made to accommodate the new situation:

Additional Assumptions

(i) Assume that $M-1$ is the number of maximally dishonest root nodes in an access structure $[n, m, N]$. Then the clustering is organized such that the sum of shareholders in any $M-1$-subset of clusters does not exceed the threshold m. I.e., any $M-1$-subset of clusters contains at most $m-1$ shareholders (analogously for $[n', m', N']$ etc.).. The threshold M is determined analogously to the threshold m, i.e., the estimated lower bound on which number of compromises an adversary is not able to achieve within a given time period.

(ii) The distributing and reconstructing set $\mathcal{B}_{\tilde{u}}$ is chosen such that the number of involved sending root nodes, i.e., I such that $\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}$, is not smaller or equal the number of maximally compromised root nodes $M-1$. In short: there exists at leasts one honest sender root node $\tilde{I}$ which is involved in the redistribution process.

Extension: EPSS Complaint Resolution Protocol (Worst Case)

1. The coordinator Coord initiates the complaint resolution protocol by broadcasting an $\mathsf{init}_{\mathsf{Resolv}}$ message.

2. Coord then selects a recorded complaint at random, say $\mathsf{reject}_{\mathsf{Sig}_j}$ by node j, and asks the complaining node to publish its received encrypted share pair

$(\mathcal{E}_j(s'_j), \mathcal{E}_j(t'_j))_{\mathsf{Sig_J}}$ which carries the receiver root node J's signature. Furthermore, node j must provide the decrypted share pair (s'_j, t'_j) along with a *proof of correct decryption* on the bulletin board.

3. The Jury can now use this information to execute the following verification checks:

   (i) $(\mathcal{E}_j(s'_j), \mathcal{E}_j(t'_j))_{\mathsf{Sig_J}}$ carries a valid signature of node J.

   (ii) Node j correctly decrypted the share pair $(\mathcal{E}_j(s'_j), \mathcal{E}_j(t'_j))$ to (s'_j, t'_j).

   (iii) Equation (12) is **not** satisfied by (s'_j, t'_j).

4. Each Jury member evaluates the three conditions, and a majority vote determines the outcome of the verification process. Depending on the outcome, the following actions are taken:

   - If neither of the conditions fail, the accusation by j is valid and the complaint resolution protocol resumes.
   - If conditions (i) and (ii) are correct, but condition (iii) fails or if condition (ii) fails (regardless of the outcome of condition (i) and (iii)), j lodged an unwarranted accusation. The complaint by j is disregarded and the coordinator Coord aborts the current instance of the redistribution protocol. It is re-initiated excluding node j.
   - If condition (i) fails, but (ii) and (iii) are correct, then at least either j or its root J was acting dishonestly. Both are marked as **VC** and the complaint is seen as resolved. The protocol is re-initiated as soon as J has been rebooted and excludes node j.

5. If the complaint by j was found to be valid, the coordinator asks its root node J to publish all encrypted partial share pairs $(\mathcal{E}_j(s_j^{\mathcal{N}_I}), \mathcal{E}_j(t_j^{\mathcal{N}_I}))_{\mathsf{Sig_I}}$ it used to compute the share pair that was sent to node j and the correctness of which is questioned.

6. The Jury executes the following verification checks:

   (i) Each encrypted partial share pair carries the correct signature of the respective sender root and is correctly formatted.

   (ii) Root node J combined the encrypted partial share pairs correctly to the new encrypted share pair for node j which was disclosed earlier.

7. Each Jury member evaluates the two conditions, and a majority vote determines the outcome of the verification process. Depending on the outcome, the following actions are taken:
   - If both conditions are seen as fulfilled by a majority of Jury members, the discrepancy must have its cause earlier in the chain and therefore the complaint resolution protocol resumes.
   - If condition (i) is revealed to fail for some I by a majority of Jury members, J is marked as **VC** because it should have detected the invalid signature or the incorrect format and aborted the protocol with a cannotResume message. The complaint is seen as resolved and the protocol is re-initiated as soon as J has been rebooted.
   - If condition (ii) is found to fail by a majority of Jury members, the root node J is marked as **VC** because it did not carry out the computations correctly. The complaint is seen as resolved and the protocol is re-initiated as soon as J has been rebooted.

8. Each sending root I with nodes in $\mathcal{B}_{\tilde{u}}$ is now asked by the coordinator to reveal the signed and encrypted sub-shares $(\mathcal{E}_j(\hat{s}_{ij}), \mathcal{E}_j(\hat{t}_{ij}))_{\mathsf{Sig_i}}$ it received from its children i for the computation of the partial shares $(\mathcal{E}_j(s_j^{\mathcal{N}_I}), \mathcal{E}_j(t_j^{\mathcal{N}_I}))$ of node j.

9. The Jury can now use this information to execute the following verification checks:
   (i) The sub-share pairs carry valid signatures of the respective sender nodes and are correctly formatted.
   (ii) Root node I combined the encrypted sub-share pairs correctly to the new encrypted partial share pair $(\mathcal{E}_j(s_j^{\mathcal{N}_I}), \mathcal{E}_j(t_j^{\mathcal{N}_I}))$ for node j.

10. Each Jury member evaluates the two conditions, and a majority vote determines the outcome of the verification process. Depending on the outcome, the following actions are taken:
    - If both conditions are seen as fulfilled by a majority of Jury members, the discrepancy must have its cause among the sending nodes in $\mathcal{B}_{\tilde{u}}$ and therefore the complaint resolution protocol resumes.
    - If condition (i) is revealed to fail for some i by a majority of Jury members, the respective root is marked as **VC** because it should have detected the invalid signature or the incorrect format and aborted the protocol

with a cannotResume message. The complaint is seen as resolved and the protocol is re-initiated as soon as all misbehaving roots have been rebooted.

- If condition (ii) is found to fail by a majority of Jury members, the respective root node is marked as **VC** because it did not carry out the computations correctly. The complaint is seen as resolved and the protocol is re-initiated as soon as all misbehaving roots have been rebooted.

11. If no fault has been found with the root nodes, node j is asked to decrypt the published sub-share pairs and to then verify their validity by checking if

$$g^{\hat{s}_{ij}} h^{\hat{t}_{ij}} \equiv g^{s_i} h^{t_i} \prod_{l=1}^{m'-1} \left(g^{a'_{il}} h^{b'_{il}}\right)^{j^l} \tag{14}$$

holds. This allows an honest node j to identify dishonest child nodes without miss.

12. Node j then reports the dishonest nodes in an authenticated manner on the bulletin board along with the decrypted sub-shares and a proof of correct decryption. This ensures that a dishonest j cannot accuse honest sender nodes of cheating without being detected by the Jury.

13. Each Jury member then verifies the following conditions:
    (i) The bulletin board entry was published by node j.
    (ii) Each accused sub-share pair $(\mathcal{E}_j(\hat{s}_{ij}), \mathcal{E}_j(\hat{t}_{ij}))$ carries node i's valid signature.
    (iii) Node j correctly decrypted the sub-share pairs $(\mathcal{E}_j(\hat{s}_{ij}), \mathcal{E}_j(\hat{t}_{ij}))$ to $(\hat{s}_{ij}, \hat{t}_{ij})$.
    (iv) Equation (14) is **not** satisfied.

    If all conditions are fulfilled for a sub-share pair, the respective sender node is marked as **VC** by the member of the Jury which confirmed the verification. If any of the conditions (ii) to (iv) fail for any accused sub-share pair, j is marked as **VC**.

14. A majority vote among the Jury then determines the compromised nodes.

15. The coordinator Coord aborts the current instance of the redistribution protocol by broadcasting an $\mathsf{abort}_{\mathsf{Redist}}$ message. Upon receiving an $\mathsf{abort}_{\mathsf{Redist}}$ message, all involved nodes securely delete any data used in the redistribution and resolution process. The coordinator then re-initiates the process by excluding all identified compromised nodes.

6.1.5 Malicious Coordinator

Of course it is possible that the coordinator Coord acts dishonestly during the protocol execution. The coordinator directs the activities of the group during the redistribution process by initiating the process and selecting the set of redistributing shareholders. Furthermore, Coord either terminates the protocol officially or can start the complaint resolution with subsequent aborts. Since the coordinator is making all of its choices due to publicly available information, a misbehaving coordinator can easily be identified. In particular, no sensitive information is shared with the coordinator that is not publicly available. Once participating nodes suspect a dishonest coordinator, they can request a coordinator change. A new coordinator will then be chosen in a deterministic round robin fashion such that an adversary cannot influence the choice of the coordinator. However, we will not describe in detail how the identification and subsequent change of a malicious coordinator is handled, for details we refer to the asynchronous PSS scheme by e.g. Schultz and Liskov[25].

6.1.6 EPSS Reconstruction Protocol

If the reconstruction of the original secret k from the access structure $[n^*, m^*, N^*]$ is wished by the client the following steps need to be taken:

1. The client C initiates the reconstruction process by sending an authenticated $\mathsf{init}_{\mathsf{Reconst}}$ message over the broadcast medium.

2. All nodes $i \in [n^*, m^*, N^*]$ publish the commitment to their currently stored share pair (s_i, t_i) of the form $g^{s_i} h^{t_i}$ on the bulletin board.

3. The client executes the algorithm $\mathcal{A}$ on $[n^*, m^*, N^*]$ to obtain a sequence of m^*-subsets $\{\mathcal{B}_u\}_{1 \leq u \leq L}$ where $L = \binom{n^*}{m^*}$. The client then selects the first element in the sequence $\mathcal{B}_1$ to check if all of its members have stored shares which are consistent to the original secret. This can be seen by evaluating if the following equation holds:

$$g^k h^t \equiv \prod_{i \in \mathcal{B}_1} (g^{s_i} h^{t_i})^{b_i} \quad \text{where} \quad b_i = \prod_{\substack{l \in \mathcal{B}_u, \\ l \neq i}} \frac{l}{l - i} \tag{15}$$

   (i) If equation (15) holds, the client publishes a $\mathsf{select}_{\mathcal{B}_1}$ message on the bullet board.

(ii) If equation (15) does not hold, the client chooses the second element $\mathcal{B}_2$ in the sequence generated by $\mathcal{A}$ and evaluates the verification for this set. This process repeats until the first $\mathcal{B}_{\tilde{u}}$ is found for which equation (15) holds.[5]

4. Let $\mathcal{B}_{\tilde{u}}$ denote the m^*-subset that was established in Step 3. All $i \in \mathcal{B}_{\tilde{u}}$ then encrypt their stored share pair with the public key of the client and sign. Then the signed cipher $(\mathcal{E}_C(s_i), \mathcal{E}_C(t_i))_{\mathsf{Sig_i}}$ is sent OTP-encrypted to the respective root node.

5. Each root I collects all shares by $i \in \mathcal{N}_I$ and verifies the sender's signatures. It then aggregates the received data by computing the encrypted partial secret pair for the client given by:

$$\begin{aligned} \mathcal{E}_C(s^{\mathcal{N}_I}) &= \sum_{i \in \mathcal{N}_I} b_i \cdot \mathcal{E}_C(s_i) \\ \mathcal{E}_C(t^{\mathcal{N}_I}) &= \sum_{i \in \mathcal{N}_I} b_i \cdot \mathcal{E}_C(t_i) \end{aligned}$$

where

$$b_i = \prod_{\substack{l \in \mathcal{N}_I, \\ l \neq i}} \frac{l}{l-i}.$$

Each partial secret pair $(\mathcal{E}_C(s^{\mathcal{N}_I}), \mathcal{E}_C(t^{\mathcal{N}_I}))$, is then signed and transmitted via the OTP-encrypted channel to the client C.

6. The client verifies the root's signatures and decrypts the received partial secrets. It then checks if the secret computed from these partial secrets is in accordance with the commitment to the stored secret. I.e., for $\tilde{k} := \sum_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} s^{\mathcal{N}_I}$ and $\tilde{t} := \sum_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} t^{\mathcal{N}_I}$ the following must hold:

$$g^k h^t \equiv g^{\tilde{k}} h^{\tilde{t}} \tag{16}$$

- If eq. (16) holds, the client C accepts $\tilde{k}$ as the reconstructed secret and broadcasts a $\mathsf{done}_{\mathsf{Reconst}}$ message. Upon receiving a $\mathsf{done}_{\mathsf{Reconst}}$ message, all involved nodes securely delete any internal data used in the reconstruction process such that the only stored data are the share pairs stored with the shareholders.

[5] Note, that such a $\mathcal{B}_{\tilde{u}}$ does always exist even in the presence of up to $m^* - 1$ dishonest nodes in $[n^*, m^*, N^*]$.

- Otherwise, the client C aborts the current instance of the reconstruction protocol by broadcasting an $\mathsf{abort}_{\mathsf{Reconst}}$ message. Upon receiving an $\mathsf{abort}_{\mathsf{Reconst}}$ message, all involved nodes securely delete any temporary data used in the reconstruction such that the only stored data are the share pairs stored with the shareholders. The client then re-initiates the process.

Remark 6.3. As with the initial distribution, if the reconstruction fails to be successful within a pre-defined time frame, further investigations into the client and the involved root nodes are necessary. A reboot of suspiciously behaving nodes (especially root nodes or the client) will ultimately resolve the issue.

7 Security Analysis of EPSS

The goal of proactive secret sharing schemes is to maintain the *integrity* and *confidentiality* of the secret k for long periods of time or even indefinitely. Furthermore, given that the (re)distribution is carried out successfully, the secret should be reconstructible at any point in time (*availability*). In this chapter, we will clarify the notion of integrity, availability and confidentiality and it will be presented under which assumptions and to what extent the EPSS protocol can provide security assurances relative to an active and mobile adversary (cf. Section 3.1.2 for a recollection of the definition).

7.1 General Assumptions and Definition

Additionally to the assumptions made in Section 5.2, we recall the following general requirements:

- The secure deletion of data by honest nodes is ensured.
- A dishonest coordinator Coord is always detected and replaced in the reinitiated redistribution process (cf. Section 6.1.5).
- All dishonest nodes act in collusion under active and mobile adversaries.
- The partially homomorphic encryption scheme $\mathcal{E}$ is considered secure at time of encryption and transmission.

During the analysis of the initial distribution, the following definition will be needed:

Definition 7.1 (Semi-honest Client)**.** *A* semi-honest *client C that is ordered to secret share k or reconstruct the secret, is a client that will not*

- *share a secret $s \neq k$, or*
- *expose the secret k or any of the computed or received shares.*

However, the client may try to distribute invalid shares to shareholders.

7.2 Integrity and Availability

Integrity assurances are an important aspect of long-term storage of any data. In PSS, integrity assurances ensure that even after many runs of the redistribution protocol, the secret to which the renewed shares correspond remains the same. Closely related to this is the notion of *availability* which asserts that the client (or any other authorized party) cannot be prevented from reconstructing the secret.

We recall that the secret k was set to be the constant factor of a polynomial $a(x)$ of degree $m-1$. The shares correspond to unique points on this polynomial and Lagrangian interpolation guarantees that any m of the shares uniquely reconstructed the original polynomial $a(x)$ and thus, the secret k.

The integrity and availability of the secret are therefore endangered as soon as less than m valid shares remain within the current access structure, since this means that there are no longer sufficiently many consistent shares available to reconstruct the original secret. Availability is further endangered if it is possible to prevent the client from receiving at least m valid shares of the secret during the reconstruction process.

7.2.1 Initial Distribution

Theorem 7.2. *The EPSS Initial Distribution protocol assures the integrity and availability of the secret k in the presence of a semi-honest client C and at most $m-1$ dishonest shareholders.*

Proof (Sketch). The secret k is first distributed during the initial distribution by a *semi-honest* client C. In the integrity and availability context, this means that the client would not enter an entirely different secret $s \neq k$ into the secret sharing process, as this could not be detected and integrity and availability assurances for the original secret k would be lost. The client may nonetheless try to distribute invalid shares (i.e., shares that are inconsistent with the secret k).

The root nodes receive their children's shares from the client. Honest root nodes will only forward correct shares to their children. Note, that the root nodes cannot verify the actual validity of the shares they received from the client, but honest roots will not forward ill-formatted, invalidly signed or altered shares to their children. Dishonest root nodes, however, could try to distribute invalid shares. This will not go undetected as they are not able to forge the client's signature and the shares would fail the verification with the published commitments and be henceforth rejected by honest shareholders.

In conclusion, once the protocol terminated successfully, at least m nodes hold shares corresponding to the secret k since at most $m-1$ out of n shareholders are dishonest and may store invalid shares (while reporting to have finished successfully). Therefore, enough valid shares are present to guarantee correct reconstructability.
□

Remark 7.3. Note, that there is no assumption about the honesty of root nodes here. Arbitrary malicious behaviour or unresponsiveness will lead to the repeated abortion of the protocol. As mentioned earlier, if the protocol is not able to terminate successfully within a reasonable time frame, investigations to the causes will be taken up, leading to the detection of the misbehaving root nodes. After rebooting them, i.e., after starting with a set of fresh root nodes, the initial distribution process will be re-initiated and then terminate eventually.

7.2.2 Redistribution

Theorem 7.4. *The EPSS Redistribution protocol assures the integrity and availability of the secret k in the presence of fewer than threshold dishonest shareholders in $[n, m, N]$ and $[n', m', N']$, respectively.*

Proof (Sketch). The protocol does only terminate successfully if at least m' valid new shares are generated and stored. Furthermore, no more than $m'-1$ invalid sub-shares can be stored as it is impossible for dishonest parties to convince honest receivers to store invalid shares. Any such attempts will be detected and the respective shares will be rejected. In between share redistributions the upper bound on the number of compromised current shareholders assures that there remain a threshold number of valid shares for redistribution and reconstruction, therefore assuring the availability and integrity of the secret k.
□

7.2.3 Reconstruction

Theorem 7.5. *The EPSS Reconstruction protocol assures the integrity and availability of the secret k in the presence of fewer than threshold dishonest shareholders in the current access structure, say $[n^*, m^*, M^*]$.*

Proof (Sketch). Finally, during reconstruction, the client requests the shares from m^* members of $[n^*, m^*, M^*]$ whose stored shares were proven to be consistent with the secret k. The root nodes of each cluster $\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}$ are then responsible for combining their children's shares to a partial secret pair $(\mathcal{E}_C(s^{\mathcal{N}_I}), \mathcal{E}_C(t^{\mathcal{N}_I}))$ and

forwarding it to the client. Root nodes under adversarial control cannot coerce an honest client to accept invalid partial secrets. Should the reconstruction fail repeatedly, all involved parties will be investigated and if necessary rebooted or replaced. If only up to $m^* - 1$ invalid shares are present among all shareholders during reconstruction, the secret remains reconstructible as there are always at least m^* valid ones left to find an authorized subset $\mathcal{B}_{\tilde{u}}$. A dishonest client which e.g. reports the successful reconstruction of the secret in the case of failure, or claims to have failed although the reconstruction was successful, does not impact the integrity and availability of the secret for other reconstruction or redistribution processes.

□

7.3 Confidentiality

The protection goal confidentiality requires that no information about the secret is revealed. In order to break the confidentiality of the scheme, i.e., to reconstruct the original secret k, an adversary must learn the shares of at least m nodes within a single time period, i.e., in between share redistributions.

Since proactive secret sharing schemes ordinarily handle long-lived secrets, a further refinement to the notion of confidentiality needs to be introduced. On the one hand we have the traditional sense of confidentiality which forbids an adversary to learn the secret - but often only under certain computational assumptions and therefore within a limited time frame. Due to the homomorphically encrypted (sub-)shares, it is also potentially desirable for an adversary to collect at least m homomorphically encrypted shares from within the same time period by compromising receiver root nodes during the redistribution process (as opposed to learning the shares directly by compromising shareholders). Once the computational problem underlying the homomorphic encryption scheme is solvable, the adversary can then decrypt the stored shares and reconstruct the secret. As we are concerned with long-lived secrets, this is a practicable and reasonable approach for an adversary. Thus, there is the protection goal of long-term confidentiality on the other hand.

We want to see if the EPSS protocol can guarantee long-term confidentiality of the secret in the presence of an active and mobile adversary. To do this, we must show, that throughout the protocol the adversary can obtain only less than the threshold number of shares. We assume that all dishonest nodes act in collusion. Thus, if shareholders are compromised, their shares are compromised too and known to the adversary. Since we assumed that only fewer than threshold number of shareholders are compromised within a single time period, the adversary will - in the worst case - only have to gain knowledge of a single additional valid share from the same time period to break the confidentiality of the scheme.

We will see under which assumptions long-term confidentiality and short term confidentiality are assured. Due to the special construction of our access structures with root nodes and child nodes, the breaking of long-term confidentiality becomes easier to achieve for the adversary, as it is no longer necessary to compromise the threshold number of shareholders in total if certain root nodes can be compromised during protocol execution (i.e., either during the initial distribution, redistribution or reconstruction). Short-term confidentiality is not impacted by this since only

encrypted shares are passed through and processed by root nodes.

Additional Assumptions Additional to the assumptions made in Section 7.1, we would like to recall the following assumptions that were presented in the description of the EPSS protocol in Chapter 6:

- Assume that if $M-1$ is the number of maximally dishonest root nodes in $[n, m, N]$. Then the clustering is organized such that the sum of shareholders in any $M-1$-subset of clusters does not exceed the threshold m. I.e., any $M-1$-subset of clusters contains at most $m-1$ shareholders (analogously for $[n', m', N']$ etc.).
- The distributing and reconstructing set $\mathcal{B}_{\tilde{u}}$ is chosen such that the number of involved sending root nodes, i.e., I such that $\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}$, is not smaller or equal the number of maximally compromised root nodes M. In short: there exists at leasts one honest sender root node $\tilde{I}$ which is involved in the redistribution process.[6]

7.3.1 Initial Distribution

Theorem 7.6. *Let the client C be semi-honest. Assume that at most $m-1$ shareholders in $[n, m, N]$ are dishonest and assume the root nodes to be honest. Then long-term confidentiality is assured during the initial distribution process.*

Proof (Sketch). Since no complaint resolution is executed during the initial distribution, no critical data is exposed on the broadcast channel. The only data available to the adversary are the maximally $m-1$ shares of the possibly dishonest shareholders. Since this does not exceed the threshold m, the secret cannot be reconstructed. It is clear that if only a single root node were dishonest whose cluster contains at least one honest shareholder, the adversary could gain knowledge of the threshold number of shares to reconstruct the secret as the missing share would be known in encrypted form. Therefore long-term security is only guaranteed under the assumption of honest root nodes during the time of initial distribution protocol execution. □

Corollary 7.7. *Let the client C be semi-honest. Assume that at most $m-1$ shareholders in $[n, m, N]$ are dishonest. Then short-term confidentiality is assured during the initial distribution process.*

[6] Otherwise all shares would be known to the adversary in encrypted form and therefore the secret could be reconstructed once the computational assumption gets broken.

Proof (Sketch). Short-term confidentiality is endangered by an adversary learning the threshold number of shares unencryptedly. Since only the shareholders have access to the decrypted shares (apart from the client), the short-term confidentiality of the secret is guaranteed even in the presence of dishonest root nodes. Even if all malicious shareholders collude, they only have at most $m-1$ shares at their disposal and can therefore not reconstruct the secret k.

□

7.3.2 Redistribution

In between share redistributions (i.e., in the time intervals after a successful redistribution and before the next initialisation of the redistribution process), no threats to confidentiality are posed since the current shareholders in say $[n, m, N]$, contain at most $m-1$ dishonest shareholders and no knowledge of further shares can be gained by the adversary. A critical point for the assurance of confidentiality occurs during the redistribution process. For once, (sub)-shares are routed in their encrypted form through root nodes. On the other during complaint resolution exposes shares and sub-shares (occasionally unencrypted) on the broadcast channel in order to identify the misbehaving parties. In the following we will state under which conditions, the EPSS redistribution protocol guarantees long-term confidentiality of the secret k. At this point we wish to stress that knowledge gained in previous time periods is not advantageous to an adversary after a redistribution took place. In particular, Wong et al.[32] proved, that a combination of shares from different time periods (i.e., at least one redistribution process was executed in between) cannot be used to reconstruct the secret.

Theorem 7.8. *Let at most $m-1$ shareholders in $\mathcal{B}_{\tilde{u}}$ and up to $m'-2$ shareholders in $[n', m', N']$ be dishonest. Furthermore, assume that any cluster with 2 honest receiver nodes is under the control of an honest receiver root node. Then long-term confidentiality of the secret k is assured during redistribution.*

Proof (Sketch).

- Assume there are $\mathbf{m'-1}$ **dishonest receiver nodes in** $[n', m', N']$. Then, under the assumption that all receiver roots are honest and up to $m-1$ redistributing shareholders are compromised, the long-term confidentiality of the secret can be broken as follows: The adversary is already in the possession of $m'-1$ (unencrypted) shares. To break the confidentiality of the secret, another share needs to be gained. The goal is then to reveal the encrypted share of an honest receiver $\tilde{j}$ by the execution of the complaint resolution extension. Since

only one complaint is resolved per redistribution instance, no dishonest node $j \in [n', m', N']$ will lodge a complaint to avoid that its complaint is chosen for resolution, as this would not yield the wished-for knowledge gain to the adversary.

Let $\bar{i} \in \mathcal{B}_{\tilde{u}}$ be dishonest and let $\bar{i}$ distribute invalid sub-shares for all $j \in [n', m', N'] \setminus \{\text{dishonest receiver shareholders}\}$. Then the dishonest receiving shareholders obtain valid shares while all other new shareholders will reject their received share pair. Since fewer than $2m' - 1$ nodes were able to finish the protocol, the complaint resolution is initiated and one of the complaints, say of node $\tilde{j}$ is chosen for resolution.

Node $\tilde{j}$ will henceforth reveal its received share pair in encrypted and unencrypted form along with a proof of correct decryption. The Jury will find the complaint to be valid, as the share pair will not satisfy the verification equation. The encrypted partial shares revealed by its root node $\tilde{J}$ will show that $\tilde{J}$ has computed the share pair values correctly and therefore the mistake must have happened earlier. Next, all sender roots $\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}$ will reveal the sub-shares $(\mathcal{E}_{\tilde{j}}(\hat{s}_{i\tilde{j}}), \mathcal{E}_{\tilde{j}}(\hat{s}_{i\tilde{j}}))$ on the broadcast channel. Node $\tilde{j}$ will then identify $\bar{i}$ as dishonest and the redistribution protocol will be initiated excluding $\bar{i}$ from the selection process of $\mathcal{B}_{\tilde{u}}$.

We see the adversary has gained knowledge of the encrypted sub-shares of honest $i \in \mathcal{B}_{\tilde{u}}$ which it was missing to compute the encrypted share $s'_{\tilde{j}}$ of $\tilde{j}$. The adversary now holds $m' - 1$ unencrypted share plus one encrypted valid share. As soon as the computational assumption is broken, the adversary can decrypt the missing share and then reconstruct the secret k. Therefore, long-term confidentiality of the secret k is not ensured.

- Now assume that there exists a **dishonest receiver root** $\bar{J}$ with at least two child nodes $\tilde{j}_1$ and $\tilde{j}_2$ in its cluster. Then the long-term confidentiality of the secret k is broken in the presence of $m' - 2$ dishonest receiver nodes. Since all dishonest nodes are assumed to act in collusion, all nodes $j \in [n', m', N']$ can receive valid shares and the redistribution process will terminate successfully. Since $\bar{j}$ computes the encrypted valid shares of $\tilde{j}_1$ and $\tilde{j}_2$, it can decrypt these two values once the computational assumption is broken. Combined with the $m' - 2$ invalid shares it already has at its disposal, the secret can be reconstructed.

□

Theorem 7.9. *Let the total number of (encrypted) shares known to dishonest receiver roots not exceed $m' - 2$.*[7] *Let the number of dishonest redistributing shareholders $i \in \mathcal{B}_{\tilde{u}}$ not exceed $m - 1$. Then the long-term confidentiality of the secret k is assured during redistribution.*

Proof (Sketch).

Assume that the **number of encrypted shares known to dishonest receiver root nodes is $\mathbf{m' - 1}$**. Then the long-term confidentiality of the secret is broken since e.g. a dishonest sender $\bar{i} \in \mathcal{B}_{\tilde{u}}$ could manipulate enough new shares such that the complaint resolution for the share of an honest node $\tilde{j}$ kicks in. The ensuing resolution lets the adversary obtain the missing encrypted valid share of $\tilde{j}$. Cf. the proof of Theorem 7.8 for details.

□

Corollary 7.10. *Let the total number of (encrypted) shares known to dishonest receiver roots not exceed $m' - 1$. Then - if both all $i \in \mathcal{B}_{\tilde{u}}$ and all involved sending root nodes $\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}$ are honest - the long-term confidentiality of the scheme is assured during redistribution.*

Proof (Sketch). The proof of Theorem 7.9 showed that dishonest receiver roots can only gain additional knowledge if the complaint resolution can be invoked for an honest node $\tilde{j}$ whose share is not yet known to the adversary. This can only be achieved by either dishonest sending shareholders or dishonest sending roots. Since both of these parties are assumed to be honest in this case, the adversary can not learn the threshold number of shares to break the long-term confidentiality of the secret.

□

Theorem 7.11. *Assume that at most $m - 1$ shareholders in $[n, m, N]$ and at most $m' - 2$ in $[n', m', N']$ are dishonest. Then the short-term confidentiality of the secret k is assured during redistribution.*

Proof (Sketch). Under the made assumptions, the adversary has knowledge of $m - 1$ and $m' - 2$ shares in the plain text. It must be shown that the adversary cannot gain knowledge of an additional **unencrypted** share of either the old access structure or two unencrypted shares of the new access structure. Since all data going through the root nodes is encrypted for the respective receiver node in $[n', m', N']$,

[7] Receiver roots know all encrypted shares within their cluster. Furthermore, more shares can be available to the roots through up to $m' - 1$ compromised shareholders in $[n', m', N']$.

this information is of no use to a computationally bounded adversary. Neither is a forced complaint resolution of a rejected share by an honest $\tilde{j}$ helpful as only maximally one complaint is resolved. Therefore, the adversary cannot gain the missing shares for immediate reconstruction of the secret k.

□

7.3.3 Reconstruction

Theorem 7.12. *Let the client C be semi-honest. Given honest root nodes and at most m^*-1 dishonest shareholders in the reconstructing access structure $[n^*, m^*, N^*]$, long-term confidentiality of the secret is assured during reconstruction.*

Proof (Sketch). Since no complaint resolution is executed, no critical data is exposed on the broadcast channel. During the reconstruction of the secret, root nodes learn the encrypted shares of their children i if they are contained in the set $\mathcal{B}_{\tilde{u}}$. Since we assume that (at most) $m^* - 1$ shareholders may reveal their shares due to malicious behaviour it is clear that all root nodes must be honest to assure the long-term confidentiality of the secret k.

□

Corollary 7.13. *Let the client C be semi-honest. Given at most $m^* - 1$ dishonest shareholders in the reconstructing access structure $[n^*, m^*, N^*]$, short-term confidentiality of the secret is assured during reconstruction.*

Proof. The argumentation is analogous to the initial distribution case in Corollary 7.7. During reconstruction, root nodes only learn encrypted share pairs of a subset of the shareholders, while the dishonest shareholders cannot obtain sufficiently many unencrypted shares to reconstruct the secret. Therefore, short-term confidentiality is assured during the reconstruction of the secret k.

□

8 Communication Analysis

In the following we wish to conduct a small analysis concerning the communication complexity of the previously presented PSS schemes G^2_{its} VSR[16] and EPSS during the redistribution process. Furthermore, the communication complexity of simply applying G^2_{its} VSR on top of the clustered structure introduced in Chapter 5 will be determined. This is necessary as it is in general not possible to directly compare G^2_{its} VSR with the EPSS scheme, since they differ in the crucial assumption about which channels are secure to use for transmission of shares.

The analysis will examine the case in which all participants are honest as dishonest parties can generate an arbitrary amount of traffic. In particular, we will focus exclusively on traffic sent on private channels as the broadcast channels are mainly used to resolve complaints and we are primarily interested in the amount of traffic generated by the transmission of sub-shares from old to new shareholders such that they can compute their new shares. Furthermore, signature sizes are omitted as they can be seen as a constant factor in the analysis conducted below.

8.1 Parameters

We look at the redistribution from shares of a secret k from an $[n, m]$ access structure to an $[n', m']$ access structure in the case of the original G^2_{its} VSR scheme. In the clustered scenario, the redistribution takes place between $[n, m, N]$ and $[n', m', N']$. Although it is reasonable to choose $n = 3m - 1$ in the EPSS case due to the asynchronity assumption involved, we take $n = 2m - 1$ to enable a more meaningful comparison with the protocol by Gupta and Gopinath.

We recall, that the original secret k was an element of $\mathbb{Z}_p$. As a result of the properties of Shamir's secret sharing, all shares and (sub)shares are also in $\mathbb{Z}_p$. Let $\mathcal{E} : \mathbb{Z}_p \to \mathbb{Z}^*_{p_1}$ be the homomorphic encryption scheme used throughout EPSS to protect the confidentiality of (sub-)shares that are passed to and through root nodes. Due to ciphertext expansion caused by probabilistic encryption schemes, it holds that $p_1 > p$ necessarily. For example, for the state-of-the-art additive homomorphic Paillier encryption scheme[21] it holds that $p_1 = p^2$. We remind that messages are further encrypted with OTP to secure the (sub-)shares for transport and that the length of an OTP-cipher is equal to the plain text length.

8.2 Comparison

8.2.1 G^2_{its} VSR

In the protocol by Gupta and Gopinath, the assumption is made that there are private point-to-point channels between any two shareholders. We assume that these channels are secured by OTP encryption to guarantee long-term confidentiality. Therefore, during redistribution, each old shareholder $i \in [n, m]$ can send its sub-share pair $(\hat{s}_{ij}, \hat{t}_{ij}) \in (\mathbb{Z}_p^*)^2$ directly to every $j \in [n', m']$. Therefore, $\mathbf{n} \cdot \mathbf{n'}$ sub-share pairs in $(\mathbb{Z}_p^*)^2$ are sent during redistributon.

8.2.2 G^2_{its} VSR in Enterprise Scenario

Next, we want to see how G^2_{its} VSR performs if the sub-shares cannot be sent directly from old shareholders in $[n, m, N]$ to new shareholder in $[n', m', N']$ but have to be routed through a root node on each side. Since only the communication channels from shareholder to root and in between roots are secured by OTP-encryption, it now becomes necessary for the old shareholders to encrypt their shares with a public key encryption scheme, say $\mathcal{E}$ for simplicity. Note that here, any public key encryption scheme would be sufficient, as the additive homomorphic property is not needed due to the missing data aggregation which is performed in EPSS.

Each $i \in [n, m, N]$ sends n' encrypted sub-share pairs $(\mathcal{E}_j(\hat{s}_{ij}), \mathcal{E}_j(\hat{t}_{ij})) \in (\mathbb{Z}_{p_1}^*)^2$ to their root nodes which then distribute the sub-shares to the according receiver roots. The receiver roots then send n sub-share pairs to each of their children j. Therefore, in total $\mathbf{3} \cdot \mathbf{n} \cdot \mathbf{n'}$ sub-share pairs in $(\mathbb{Z}_{p_1}^*)^2$ are sent during redistribution.

8.2.3 EPSS

Let $\tilde{n}$ be the number of sender root nodes which have children $i \in \mathcal{B}_{\tilde{u}}$, i.e., the number of sender root nodes involved in the redistribution process. In the EPSS redistribution protocol, root nodes on the sender side aggregate $m \cdot n'$ received sub-shares into $\tilde{n} \cdot n'$ partial shares and the receiving root nodes further combine these to n' shares. Note, that encrypted sub-shares, partial shares and shares are of the same size. Consequently there are $(\mathbf{m} + \tilde{\mathbf{n}} + \mathbf{1}) \cdot \mathbf{n'}$ (sub-)share and partial share pairs in $(\mathbb{Z}_{p_1}^*)^2$ sent in total.

8.3 Result

Unsurprisingly, G^2_{its} VSR in its original setting outperforms EPSS, but this is meaningless since the assumption of private point-to-point channels between any two

participating nodes made in G^2_{its} VSR is stronger than the one made in EPSS. When looking at the redistribution of sub-shares within the clustered scenario, the presented EPSS is more efficient than the simple routing of sub-shares along the given communication path. We recall that $\tilde{n} \leq N << n$. If one takes n to be chosen such that $n = 2m - 1$, then $m \approx \frac{n}{2}$, and if say $\tilde{n} \approx \frac{n}{5}$ then $m + \tilde{n} + 1 \approx 1.7n < 3n$.

9 Conclusion

9.1 Summary

In this thesis, a proactive secret sharing protocol named EPSS was designed which works under the condition that there need not exist secure point-to-point channels between any two participants in the proactive secret sharing process. We found that both this situation as well as the need for proactive data storage can be found especially within companies. Therefore the presented EPSS scheme was modelled to this specific environment. We gave a brief indication that EPSS outperforms the PSS protocol by Gupta and Gopinath[16] under the given modified assumptions. Furthermore, a small comparison of long-term secure primitives such as encrypted channels and commitment schemes, which are commonly used within PSS schemes, with respect to their applicability was presented.

9.2 Future Work

Efficiency improvement is key in long-term confidentiality research, especially for proactive secret sharing schemes. There still remains a vast body of work to be done to make proactive secret sharing schemes efficient and useable for the wide public. The EPSS scheme can generally be used in the asynchronous network setting, but the details and intricacies of this were left out in this first version which aimed at presenting the general idea.

References

[1] Y. Aumann, Y.Z. Ding, and M.O. Rabin. Everlasting security in the bounded storage model. *Information Theory, IEEE Transactions on*, 48(6):1668–1680, Jun 2002. Cited on page 26.

[2] Y. Aumann and M.O. Rabin. Information theoretically secure communication in the limited storage space model. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 65–79. Springer Berlin Heidelberg, 1999. Cited on page 26.

[3] C. H. Bennett and G. Brassard. Quantum Cryptography: Public Key Distribution and Coin Tossing. In *Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing*, pages 175–179, New York, 1984. IEEE Press. Cited on page 26.

[4] G.R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, Monval, NJ, USA, 1979. AFIPS Press. Cited on page 9.

[5] J. Braun, J. Buchmann, C. Mullan, and A. Wiesmaier. Long term confidentiality: a survey. *Designs, Codes and Cryptography*, 71(3):459–478, 2014. Cited on page 24.

[6] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 88–97, New York, NY, USA, 2002. ACM. Cited on pages 13 and 14.

[7] C. Cachin and U. Maurer. Unconditional security against memory-bounded adversaries. In Jr. Kaliski, BurtonS., editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer Berlin Heidelberg, 1997. Cited on page 26.

[8] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, November 2002. Cited on page 14.

[9] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 383–395, Oct 1985. Cited on page 10.

[10] I. Csiszar and J. Körner. Broadcast channels with confidential messages. *Information Theory, IEEE Transactions on*, 24(3):339–348, May 1978. Cited on page 25.

[11] D. Demirel and J. Lancrenon. How to securely prolong the computational bindingness of pedersen commitments. Cryptology ePrint Archive, Report 2015/584, 2015. Cited on page 21.

[12] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Technical Report ISSE-TR-97-01, Department of Information and Software Engineering, George Mason University, 1997. Cited on pages 13 and 14.

[13] A. K. Ekert. Quantum cryptography based on bell's theorem. *Phys. Rev. Lett.*, 67:661–663, Aug 1991. Cited on page 26.

[14] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87, pages 427–438, Washington, DC, USA, 1987. IEEE Computer Society. Cited on pages 10, 11, and 12.

[15] V. H. Gupta and K. Gopinath. An extended verifiable secret redistribution protocol for archival systems. In *ARES*, pages 100–107. IEEE Computer Society, 2006. Cited on page 13.

[16] V. H. Gupta and K. Gopinath. g^2_{its} vsr: An information theoretical secure verifiable secret redistribution protocol for long-term archival storage. *Security in Storage Workshop, International IEEE*, 0:22–33, 2007. Cited on pages 8, 13, 14, 16, 21, 29, 30, 32, 64, and 67.

[17] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352. Springer, 1995. Cited on pages 12 and 14.

[18] U. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992. Cited on page 25.

[19] U. Maurer. Secret key agreement by public discussion from common information. *Information Theory, IEEE Transactions on*, 39(3):733–742, May 1993. Cited on page 25.

[20] U. Maurer, R. Renner, and S. Wolf. Unbreakable keys from random noise. In P. Tuyls, B. Skoric, and T. Kevenaar, editors, *Security with Noisy Data*, pages 21–44. Springer-Verlag, 2007. Cited on page 25.

[21] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin Heidelberg, 1999. Cited on page 64.

[22] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, 1992. Springer-Verlag. Cited on pages 10, 11, 12, and 21.

[23] M.O. Rabin. Provably unbreakable hyper-encryption in the limited access model. In *Theory and Practice in Information-Theoretic Security, 2005. IEEE Information Theory Workshop on*, pages 34–37, Oct 2005. Cited on page 26.

[24] R. Renner and S. Wolf. Simple and tight bounds for information reconciliation and privacy amplification. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 199–216. Springer Berlin Heidelberg, 2005. Cited on page 25.

[25] D. Schultz, B. Liskov, and M. Liskov. Mpss: Mobile proactive secret sharing. *ACM Trans. Inf. Syst. Secur.*, 13(4):34:1–34:32, December 2010. Cited on pages 13, 14, and 51.

[26] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. Cited on pages 7 and 9.

[27] C.E. Shannon. Communication theory and secrecy systems. *Bell Systems Technical Journal*, 28-4:656–715, Oct 1949. Cited on page 23.

[28] T. Tassa. Hierarchical threshold secret sharing. In Moni Naor, editor, *Theory of Cryptography*, volume 2951 of *Lecture Notes in Computer Science*, pages 473–490. Springer Berlin Heidelberg, 2004. Cited on page 29.

[29] G.S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *American Institute of Electrical Engineers, Transactions of the*, XLV:295–301, Jan 1926. Cited on page 23.

[30] M. Vigil, J. Buchmann, D. Cabarcas, C. Weinert, and A. Wiesmaier. Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving. *Comput. Secur.*, 50(C):16–32, May 2015. Cited on page 7.

[31] S. Wolf. Unconditional security in cryptography. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 217–250, London, UK, 1999. Springer-Verlag. Cited on page 23.

[32] T.M. Wong, C Wang, and J.M. Wing. Verifiable secret redistribution for archive systems. *Security in Storage Workshop, International IEEE*, 0:94, 2002. Cited on pages 13, 14, and 60.

[33] A.D. Wyner. The wire-tap channel. *Bell System Technical Journal, The*, 54(8):1355–1387, Oct 1975. Cited on page 24.

[34] V. Yakovlev, V. Korzhik, and G. Morales-Luna. Key distribution protocols based on noisy channels in presence of an active adversary: Conventional and new versions with parameter optimization. *Information Theory, IEEE Transactions on*, 54(6):2535–2549, June 2008. Cited on page 25.

[35] L. Zhou, F. B. Schneider, and R. Van Renesse. Apss: Proactive secret sharing in asynchronous systems. *ACM Trans. Inf. Syst. Secur.*, 8(3):259–286, August 2005. Cited on page 13.