A preliminary version of this paper appears in the proceedings of the *14th Annual Conference on Privacy, Security and Trust (PST 2016)*. This is the full version.

Efficient Proactive Secret Sharing

Jacqueline Brendel and Denise Demirel

TU Darmstadt, Germany

July 2017

Abstract

The secure storage of long-lived sensitive data is constantly growing in its relevance due to the ever increasing digitization of documents. One very important challenge of this research field is to provide confidentiality for the stored data even in the long term. The only known approach to achieve this, as required, for instance, for medical records, is to use proactive secret sharing. However, all currently known schemes suffer from being inefficient. They require information-theoretic secure communication channels between any two shareholders and between the client and each shareholder and come with a high communication complexity. Thus, this work addresses the scenario where only a subset of servers holding shares is connected via private channels. Furthermore, it is sufficient if there is only one private channel between the client and one shareholder. In addition to improving practicability the presented proactive secret sharing solution, called EPSS, performs data aggregation to provide an efficient solution with respect to the communication complexity. Nevertheless, it still provides unconditional confidentiality for the data at rest and towards external attackers eavesdropping the communication channels.

1 Introduction

Digital storage is growing in its relevance due to the ever increasing digitization of documents. This concerns all areas of our lives: from medical records and any kind of legal data to company and state secrets. In all these scenarios special precautions must be taken to ensure the basic protection goals of authenticity, integrity, and confidentiality for *the entire lifetime* of electronic data. Medical records, for instance, must be kept as long as the concerned individuals are alive. Thus, the archiving and protection period may easily extend to 100 years. This leads to new challenges, since over time hardware may fail and needs to be replaced. Furthermore, the security of complexity based cryptographic algorithms, such as encryption and signature schemes, will fade out as computational and cryptanalytical capabilities evolve. Thus, long-term secure solutions are needed that handle these changes while retaining the required protection goals for the data at any point in time for its entire life span.

The challenges of long-term authenticity and integrity are well-studied (cf., e.g., Vigil et al.'s survey [VBC+15] for a comprehensive overview of the approaches) and efficient solutions are available.

The only known approach to address long-term confidentiality is to use proactive secret sharing (PSS) as introduced by Herzberg et al. [HJKY95]. Using this technique, documents are split into shares that are distributed among a set of shareholders. To prevent that a mobile adversary is able to collect enough shares over time to reconstruct the data the shares are renewed periodically.

Nevertheless, the widespread adaptation of proactive secret sharing is still hindered by its impracticality. To renew shares each shareholder has to communicate in private with each other shareholder. This not only leads to a large amount of traffic, but also requires private channels between each two parties. Furthermore, state of the art proactive secret sharing schemes require a private channel between the client and each

shareholder. Classical encrypted channels are efficient, but do not provide long-term confidentiality. A passive adversary can simply store the eavesdropped communications and retrieve the shares (and thus the shared secret) once the underlying computational assumption that secured the channel is broken. With the key lengths selected today the data is secure for some decades only. Furthermore, even if large parameters are chosen there is no guarantee that the scheme does not break earlier than expected, e.g., due to the introduction of quantum computers. Thus, e.g., for medical records, information-theoretic secure channels are required that ensure long-term confidentiality for the data communicated.

Such channels can be built using Vernam's One Time Pad [Ver26] which is optimal with respect to the key material consumed per data transmission. However, to provide efficient solutions to exchange these keys is challenging. There exist numerous works on information-theoretic secure key agreement and exchange (e.g., [Mau93, CM97, Rab05]) but all come with considerable limitations to practical relevancy. The only efficient channels available are those providing merely computational confidentiality. This shows that with respect to current solutions there is a trade-off between confidentiality and practicability. The higher the intended level of confidentiality, the less practical the storage of data becomes.

Contribution. In this work we present a proactive secret sharing scheme called EPSS which slightly relaxes the level of confidentiality but achieves much better results concerning practicability. More precisely, we do not require point-to-point communication links between any two shareholders involved in the PSS (re)distribution and reconstruction process. Instead we cluster the shareholders into groups that can only communicate securely within the cluster itself and across clusters through single distinguished nodes, so-called *roots*. In addition, it is sufficient if there is at least one private channel between the client and one shareholder instead of a private channel between the client and all shareholders.

It is evident that existing PSS schemes cannot be transferred to this network structure without any further modifications. This has multiple reasons. Foremost, the root nodes learn all (sub-)shares that are routed through them and can cut off their cluster from the communication. Furthermore, the resolution of complaints becomes more difficult since also during this process no information may be revealed that allows to break confidentiality of the shared secret later on.

To address these challenges, all shares and sub-shares are encrypted under the public key of the receiver, which eliminates the immediate risk of exposing the secret to compromised root nodes. Even if the threshold number of shares is exposed to compromised colluding (root) nodes, the secret cannot be reconstructed as long as the encryption scheme is secure. Note that although the clustered scenario admits fewer communication channels, the same information needs to be transmitted in order to provide equivalent functionality. To reduce the load on the network, each root node aggregates data aimed for the same receiver. More precisely, root nodes on the sender side combine the sub-shares they receive to one partial sub-share per receiver node. These are then transmitted to the respective receiver root nodes which combine them further by computing the new share for each of their children. As this requires the root nodes to process encrypted (sub-)shares, we use an additively homomorphic encryption scheme to encrypt the shares sent over the network.

Regarding communication complexity there is a slight increase with respect to the data being sent due to ciphertext expansion. However, the combination of this measure with data aggregation decreases the load on the network in comparison to a solution without either of the measures. Regarding confidentiality we distinguish between data in transit and data at rest. With respect to data in transit confidentiality towards the root nodes is only provided computationally due to the used encryption scheme. However, external attackers do not learn anything about the communicated data, not even in the long-term. Thus, this solution still provides a reasonable level of confidentiality with respect to the data in transit. Regarding data at rest long-term confidentiality is still provided without any restriction. To our knowledge, this is the first proactive secret sharing solution that provides a reasonable trade-off between long-term confidentiality

and efficiency bringing long-term secure storage solutions closer to practice.

Structure. This work is structured as follows. After related work in Section 2 we describe the general functionality of PSS schemes in Section 3. The model and the assumptions made for our protocol follow in Section 4. The protocol itself is presented in Section 5, followed by a performance and security analysis in Section 6. We conclude with directions for future work in Section 7.

2 Related Work

The first PSS scheme was proposed by Herzberg et al. [HJKY95] in 1995 and further refinements in the area were made by Desmedt and Jajoda [DJ97], Wong et al. [WWW02], and Gupta and Gopinath [GG06,GG07]. While these schemes all relied on a synchronous network, the works of Cachin et al. [CKLS02], Zhou et al. [ZSVR05], and Schultz et al. [SLL10] presented PSS protocols in the asynchronous network setting. Since most existing networks are asynchronous in nature (e.g., the Internet), these works set a further milestone in making PSS applicable in a real world context. However, all these approaches assume the existence of private channels between all shareholders and the client and each shareholder.

With regard to private channels, a somewhat promising approach is given by quantum key distribution ([BB84, Eke91]). Its security relies on the fundamental laws of quantum mechanics. Most recent developments in the field and the demonstration of the real world applicability within the Tokyo QKD Network [SHF+14] give hope that in the future these mechanisms will be able to first support business-to-business applications and will eventually be available to regular users. Although the Tokyo QKD Network provides methods for secret sharing, share renewal and especially how to perform this efficiently is not addressed. For now, the exchange of OTP key material has been known to happen mostly *out of band*, e.g., on disks via trusted couriers or in diplomatic pouches which is a cumbersome approach. Thus, in this work we show how to reduce the number of necessary private channels and thereby significantly reduce the amount of key material that needs to be exchanged.

3 Proactive Secret Sharing

In traditional (m, n)-threshold secret sharing schemes such as Shamir's [Sha79] a secret is distributed across n shareholders, such that any subset of m shares is sufficient to reconstruct it. However, no malicious subset of at most $m-1$ shareholders can recover any information about the distributed secret in an information-theoretic sense. To prevent malicious clients from distributing invalid shares so-called *verifiable* secret sharing schemes, e.g. [Fel87], [Ped92], have been developed. However, they cannot provide confidentiality for the secret's whole life span, because given enough time a mobile adversary might be able to collect enough shares to reconstruct the secret.

With the introduction of PSS this shortcoming was resolved by periodically renewing the shares. Some PSS schemes further allow to change the values of m and n from one period to the next, i.e. they support dynamic addition and removal of shareholders. Since these procedures not only renew already existing shares but may distribute new shares to different shareholders altogether, they are often referred to as share *redistributions* rather than *renewals*. More precisely, after the initial distribution of the shares, where the secret is divided into shares and distributed to the shareholders, the share redistribution takes place in periodic intervals. In the following, we briefly describe the general functionality of the redistribution mechanism that is also integrated in our EPSS solution and will be presented in more detail in Section 5.

Let $[n, m]$ be the set of current shareholders and $[n', m']$ the set of new shareholders to which the redistribution shall take place. Let $\mathcal{S}_{m,n}(\cdot)$ be the underlying (m, n)-threshold secret sharing procedure. Each node $i \in [n, m]$ applies the secret sharing procedure $\mathcal{S}_{m',n'}(\cdot)$ to their currently stored share s_i. This results

in a sub-sharing $\{\hat{s}_{ij}\}_{j=1}^{n'}$ and each i sends the sub-share $\hat{s}_{ij}$ to the respective receiver $j \in [n', m']$ using a private channel. The receiver nodes in $[n', m']$ then agree on an m-subset of sender nodes in $[n, m]$, say $\mathcal{B}_{\tilde{u}}$. After this, each j can compute their new share s'_j by combining the m sub-shares $\{\hat{s}_{ij}\}_{i \in \mathcal{B}_{\tilde{u}}}$. Once the new shares have been computed and stored, the old shareholders in $[n, m]$ securely erase their now obsolete shares. The redistribution process is complete. Note that by creating sub-shares of existing shares and recombining them, the original secret remains unchanged, but shares from different time periods cannot be combined to reconstruct the secret.

4 Model Description and Assumptions

In this section, we present the setup in which the EPSS scheme described in Section 5 is situated and the underlying assumptions. A PSS scheme aiming to achieve long-term security requires the communication channels between any two parties to be information-theoretically secure. Obviously, this is not an option for most use cases as establishing information-theoretically secure channels is either costly and subject to special provisions (e.g., QKD) or arduous (out of band exchange of OTP key material). Thus, to gain efficiency and a more realistic model we propose an approach where shareholders are partitioned into clusters. Secure channels are then established solely within the clusters and across clusters via distinguished nodes. These special nodes will be referred to as *roots* or *root nodes*, while the other nodes are referred to as *child nodes*. The client is connected with at least one root node via a private channel.

4.1 Protection Goals

The EPSS scheme targets the protection goals integrity, availability, and long-term confidentiality.

Integrity and Availability

In PSS schemes, integrity is ensured by providing a verifiable distribution, redistribution, and reconstruction process. This guarantees that even after performing these processes the stored secret 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. Note that in order to provide long-term integrity additional measures must be taken, see for instance [VBC+15], but this is out of scope for this work.

Long-term Confidentiality

For confidentiality we distinguish between data in transit and data at rest. To ensure long-term confidentiality for data at rest proactive secret sharing is used. This technique prevents that even a computationally unbounded and mobile adversary can learn the stored data. In our solution we determine a further refinement to the notion of confidentiality with respect to data in transit: On the one hand we have the traditional sense of long-term confidentiality which forbids an external unbounded adversary to learn the secret. This can be achieved using OTP to encrypt all communication channels. On the other hand, we require only computational confidentiality with respect to malicious nodes. This is provided by encrypting data that passes several nodes on its way from the sender to the receiver.

4.2 Notation

For the remainder of the discussion, let the client be denoted by C and let n be the total number of participating nodes with identifiers $1, 2, \ldots, n$. Let m define the threshold of the secret sharing scheme,

i.e., the number of shares needed to reconstruct the shared secret k. Furthermore, to accommodate the distinction between root and child nodes, N shall denote the number of root nodes among the participating nodes. Using this notion, the access structures will in the following be denoted by $[n, m, N]$. Obviously, N should be chosen such that $N << n$ to gain a significant advantage. Otherwise the number of secure communication channels between root nodes is close to the original unclustered setup. Furthermore, it is advisable that clusters are roughly of the same size. This avoids imbalances in the conceived worth or vulnerability to potential adversaries. The root nodes are denoted in capital letters to distinguish their identifiers from the child nodes and the cluster of root node I will commonly be referred to as $\mathcal{N}_I$.

4.3 Network Model and Assumptions

To prove the accomplishment of the above mentioned protection goals by EPSS, we require the following assumptions:

Separation of Roots and Shareholders

The functionality of a node involved in PSS may lead to additional requirements regarding the involved hard- and software, e.g., with respect to storage space, response time, monitoring, and measures to protect against attackers. Thus, we distinguish between two types of nodes within the protocol: root nodes and child nodes.

The primary function of root nodes is to enable secure communication flows across clusters and between child nodes and the client. Child nodes on the other hand have the sole purpose of storing shares and redistributing them. Since the root nodes are vital parts of the network infrastructure, these nodes must be granted stronger security and stability assertions and greater access to computational resources than simple child nodes. Furthermore, while certain root nodes may become expendable and/or may be replaced over time, a high fluctuation from one time period to the next is not expected.

This is in strong contrast to the interchangeability of the child nodes. This separation goes in accordance with all existing PSS protocols since these also distinguish between shareholders, clients, and other components of the network infrastructure, such as switches.

Asynchrony and Reliability

The network is assumed to be of an asynchronous nature. The underlying network is only considered reliable in the sense that all sent messages cannot be arbitrarily delayed. This means that repeated retransmission leads to eventual delivery. As pointed out by Schultz et al. [SLL10], this simply ensures termination of PSS protocols and has no implications beyond.

Private Channels

Minimally, there exist secure point-to-point channels from the child nodes to the root node within a cluster and between any two root nodes. Furthermore, we assume that a client that wants to share or reconstruct a secret has the means to communicate securely with at least a single root node. This root node is then responsible for distributing the received data to all other root nodes. Note that to simplify the protocol description we assume that the client has a direct communication link to *every* root node.

Broadcast

There exists a reliable broadcast channel to which all participating nodes have access. In particular, nodes cannot be prohibited from reading data from or sending data to the broadcast channel. There are many

ways to implement them, but in the following we will use the example of a public bulletin board [HL09] due to its illustrative properties. We assume that any message posted on the bulletin board will be available on the board until the protocol terminates.

Message Authentication

All messages on both point-to-point as well as broadcast channels, are assumed to be authenticated by a public key signature scheme $\mathcal{S}$.

4.4 Cryptographic Assumptions

Information-theoretically secure channels

All point-to-point channels are information-theoretically secure (e.g., by using OTP encryption).

Unforgeable signatures

The signature scheme $\mathcal{S}$ provides existential unforgeability under adaptive chosen message attacks at the time of signature generation and transmission.

Additively homomorphic encryption

The public key encryption scheme, in the following denoted by $\mathcal{E}$, is additively homomorphic and is considered (computationally) secure at time of encryption and decryption.

Unconditionally hiding commitments

The used commitment scheme is unconditionally hiding. In our protocol description we will use Pedersen commitments [Ped92].

Each participating node maintains a public key pair for the above mentioned encryption and signature operations, respectively. We assume that the public keys are known to all other parties within the network.

4.5 Adversary Model

The underlying threat model is that of a mobile and active adversary. More precisely, the adversary can dynamically attack nodes within the network by moving from node to node and can cause arbitrary malicious behaviour of the compromised nodes. In case of a compromise, all (secret) information stored within the server becomes available to the adversary. In particular, this implies that the adversary can spoof and read messages. The adversary is assumed to be in full control over the network. This means that the adversary has the ability to alter and inject messages. Additionally, the adversary can schedule when (and if) messages are delivered.

We will not differentiate between adversary-induced malicious behaviour of nodes and such caused by 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.

4.6 Further Assumptions

The threshold m in the secret sharing scheme is chosen such that the above described adversary can never corrupt more than $m-1$ nodes within the network at any given time. From this, the number of nodes n is determined by the relation $m = \left\lfloor \frac{n}{3} \right\rfloor$ which is optimal in the Byzantine setting [LSP82, GIKR01]. Let

$M-1$ be the number of maximally expected dishonest root nodes in the access structure $[n, m, N]$. Then we assume that the clustering is organized such that any $M-1$-subset of clusters contains at most $m-1$ child nodes (analogously for $[n', m', N']$). During share renewal, the redistributing set $\mathcal{B}_{\tilde{u}}$ is chosen such that there exists at least one honest sender root node $\tilde{I}$ such that $\mathcal{N}_{\tilde{I}} \in \mathcal{B}_{\tilde{u}}$.[1]

In addition, we assume the secure deletion of data by honest nodes is guaranteed.

At a certain point in the protocol it becomes necessary to compute a sequence of $m-$subsets of the current shareholders. To achieve this, an algorithm $\mathcal{A}([n, m, N])$ is fixed during the initialization of the scheme. $\mathcal{A}$ will output a sequence $\{\mathcal{B}_u\}_{1 \leq u \leq L}$ of subsets of $[n, m, N]$ containing m elements each where $L = \binom{n}{m}$. The algorithm $\mathcal{A}$ is known to all participating nodes.

5 The Efficient PSS Scheme: EPSS

In this section we present our improved PSS scheme called EPSS. Compared to standard PSS most improvements affect the redistribution process. In the following, we consider the worst case setting, i.e., with the maximally supported number of compromised nodes both on the child and root node level.

As mentioned before, the main novelty in EPSS is the introduction of *clusters*. We no longer require private communication channels between any two nodes participating in the protocol as in previous PSS solutions. Communication flows between shareholders are established via the so-called root nodes (of which each cluster has one). The root nodes are interconnected and do not only route data, but aggregate it whenever possible. To preserve confidentiality, root nodes only operate on encrypted (sub-)shares. We recall that whenever share-related data is in transit over the direct communication channels, the data is additionally secured by OTP-encryption. To facilitate readability, this fact will not be made explicit in the following descriptions.

5.1 EPSS Initial Distribution

The initial distribution of the secret k to an access structure $[n, m, N]$ is initiated by the client C broadcasting an $\mathsf{init}_{\mathsf{Distr}}$ message announcing all designated child nodes $i \in [n, m, N]$. C then computes the shares by applying the standard (m, n)-threshold secret sharing procedure. The shares are then encrypted under the public key of the respective receiver node. The distribution of shares occurs via the root nodes. In order to support verifiability, the client commits to both the original secret k as well as to all distributed shares. If a child node finds its share pair[2] to be valid, it stores it and a signed $\mathsf{finished}$ message is broadcast. The initial distribution was successful if after a pre-defined time frame at least $2m-1$ distinct, authenticated $\mathsf{finished}$ messages are recorded. In this case, C broadcasts a $\mathsf{done}_{\mathsf{Distr}}$ message. All involved nodes securely delete any internal data used in the distribution process such that the only stored data are the distributed share pairs hold by the child nodes. If less than $2m-1$ child nodes in $[n, m, N]$ reported the successful storage of the shares, the initial distribution failed. C then broadcasts an $\mathsf{abort}_{\mathsf{Distr}}$ message to indicate the failure. Subsequently, all involved nodes securely delete *any* data used in the distribution. C re-initiates the process with a differing set of shareholders.

More precisely, if a client C wants to store a secret k using EPPS, the initial distribution is performed as follows:

EPSS Initial Distribution

[1] This is achieved if the number of involved sender root nodes is not smaller or equal the number of maximally compromised root nodes $M-1$.

[2] Since the verifiability in EPSS is implemented using Pedersen commitments a commitment to a share s_i is computed as $g^{s_i} h^{t_i} \bmod p$ [Ped92], where p is a prime number, g, h are two generators of group $\mathbb{Z}_p^*$, and t_i is the share of some randomly chosen value t. It follows that the share consists of a share pair of the form (s_i, t_i).

1. C initializes the secret-sharing process of k by broadcasting an $\mathsf{init}_{\mathsf{Distr}}$ message which contains all designated child nodes $i \in [n, m, N]$.

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

3. C encrypts the shares s_i and t_i under the public key of node i using the additively homomorphic encryption scheme $\mathcal{E}$ resulting in the encrypted share pair $(\mathcal{E}_i(s_i), \mathcal{E}_i(t_i))$ and signs the pair.

4. The encrypted and signed share pairs for each of the N clusters are gathered and each collection is OTP-encrypted using the key of the respective root node.

5. To ensure the consistency of the shares generated and sent C commits to these values. More precisely, it 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 on the bulletin board in an authenticated manner.

6. Upon receiving the share pairs, the root nodes verify C's signature as well as the format of the message (i.e., whether the message contains a correct number of shares each of correct size). Those shares which carry a valid signature of C and are well-formatted are then forwarded to the respective child node in an information-theoretic secure fashion. Invalidly signed shares are discarded and no further action is taken by the root nodes.

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

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

   holds.

   If the verification is 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. Each child i handles invalid signatures gracefully by means of accepting shares with invalid signatures as long as the verification equation (1) is satisfied. If the verification equation (1) 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.

8. After a pre-defined time frame, C checks if there are at least $2m-1$ distinct and authentic $\mathsf{finished}$ messages present. If so, C announces the successful initial distribution of secret k 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 saved by the child nodes.

   If less than $2m-1$ child nodes in $[n, m, N]$ reported the successful storage of the shares, the initial distribution failed. 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. C then re-initiates the initial distribution process.

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 on its part, the system management will be alerted and further investigations are triggered. Such discrepancies can include for example abortion of the process although the broadcast channel has seen sufficiently many $\mathsf{finished}$ messages.

Remark. Here it is not appropriate to carry out complaint resolution (as later described for the redistribution process), since this leaks information about the secret. Note that the process of initial distribution can be executed efficiently due to its shortness. Thus, it is reasonable to re-start the process with varying child nodes until it terminates successfully. Should the process not be able to complete within a reasonable time frame, further investigations by the system management regarding the client (and - if there exist indications - the involved root nodes) are necessary. A rebooting procedure or the replacement of suspiciously behaving nodes will ultimately resolve the issue.

5.2 EPSS Redistribution

Here, we first provide an intuition of the EPSS redistribution process followed by the full version of the protocol. The redistribution is directed by a distinguished node Coord, the *coordinator*. In particular, this node will be responsible for the selection of distributing child nodes in the current access structure $[n, m, N]$ and will also determine whether the redistribution to the child nodes of the new access structure $[n', m', N']$ terminated successfully. The introduction of a coordinator becomes necessary since at some point the redistribution protocol requires to establish consensus among the honest participating nodes. The mechanisms to achieve this consensus via a coordinator are in parts inspired by Schultz et al.'s asynchronous PSS scheme MPSS [SLL10] which relies on the BFT protocol [CL02]. As in [SLL10], Coord will be selected in a deterministic round-robin fashion such that an adversary cannot influence the choice. The participating nodes can request for a new coordinator if they suspect misbehaviour. 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 Schultz et al. [SLL10].

After initiating the redistribution process by a broadcast message, Coord identifies 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 child nodes will then create an encrypted sub-sharing of their stored shares and forward the result to their root nodes. Upon receiving these sub-sharings, the root nodes will aggregate the received data into a single encrypted *partial share pair* for each child node j of the new access structure $[n', m', N']$ and transmit these to the respective receiver root nodes. Here, the data is further aggregated by computing the new (but still encrypted) share pair for each $j \in [n', m', N']$ from the received partial shares. Each child node j then stores its new share pair after decryption and verification and signals the successful storage by a broadcast finished message. Coord then collects these messages and declares the redistribution to be completed successfully once $2m' - 1$ distinct finished messages have been broadcast.

If this is not the case, a resolution of complaints identified during the redistribution protocol is applied. We recall that the worst case described here allows for malicious nodes on both the child and the root level (bounded by the respective thresholds). As elaborated in the extended version [Bre16], only a single rejection may be resolved to ensure long-term confidentiality of the secret. Afterwards, a re-initiation of the redistribution protocol is enforced. The resolution allows to trace the path of the rejected share pair from the receiver to the sender by publishing corresponding information on the bulletin board. Then, a set of entities, further referred to as the Jury, identifies the cheating party. 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, e.g., the set of receiver or sender root nodes.

At the end of the complaint resolution protocol, indications to misbehaving nodes in the structure are established. Coord then re-initiates the redistribution process by excluding all identified compromised nodes.

EPSS Redistribution

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

2. 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 a Pedersen commitment to their share pair (s_i, t_i) of the form $g^{s_i} h^{t_i}$ mod p on the bulletin board [3], where p is a prime number and g, h are two generators of group $\mathbb{Z}_p^*$.

4. Coord 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 $L = \binom{n}{m}$. It then selects the first set $\mathcal{B}_{\tilde{u}}$ where all members hold shares which are consistent with the original secret. This is done by evaluating the following equation:

$$g^k h^t \equiv \prod_{i \in \mathcal{B}_{\tilde{u}}} (g^{s_i} h^{t_i})^{b_i} \quad \text{where} \quad b_i = \prod_{\substack{l \in \mathcal{B}_u, \\ l \neq i}} \frac{l}{l - i} \tag{2}$$

   Coord publishes a $\mathsf{select}_{\mathcal{B}_{\tilde{u}}}$ message on the bullet board.[4]

5. All sender child nodes $i \in \mathcal{B}_{\tilde{u}}$ compute their sub-sharings according to the new access structure $[n', m', N']$. Each sub-share is homomorphically encrypted for the respective receiver child 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 respective root node of i.

6. Each sender node i publishes the commitments $g^{a'_{i1}} h^{b'_{i1}}, g^{a'_{i2}} h^{b'_{i2}}, \ldots, g^{a'_{i(m'-1)}} h^{b'_{i(m'-1)}}$ on the bulletin board in an authenticated manner.

7. Each root I collects all sub-sharings received from its child nodes $i \in \mathcal{N}_I$ and verifies their signatures. It then aggregates the received data by computing the encrypted partial share pair $(\mathcal{E}_j(s_j^{\mathcal{N}_I}), \mathcal{E}_j(t_j^{\mathcal{N}_I}))$ for receiver node j given by:

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

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

   Each partial share pair is then signed and transmitted to the cluster root node J to which j belongs.

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

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

   Then, it signs the sum of share pairs and transmits them to j.

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

$$g^{s'_j} h^{t'_j} = \prod_{\mathcal{N}_I \in \mathcal{B}_{\tilde{u}}} \prod_{i \in \mathcal{N}_I} \Big(g^{s_i} h^{t_i} \prod_{l=1}^{m'-1} (g^{a'_{il}} h^{b'_{il}})^{j^l} \Big)^{b_i} \tag{3}$$

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

[3] In the following, all operations are to be understood modulo p.

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

10. After the successful decryption, verification, and storage of the new share pair, j publishes a signed $\mathsf{finished}_{\mathsf{Sig_j}}$ message on the bulletin board. Otherwise, j records a complaint via $\mathsf{reject}_{\mathsf{Sig_j}}$ on the bulletin board. Coord announces the successful termination of the redistribution process by broadcasting a $\mathsf{done}_{\mathsf{Redist}}$ message once it verified the existence of $2m' - 1$ correctly signed, distinct finished messages. This ensures that at least m' valid shares have been redistributed.

11. Upon receiving the $\mathsf{done}_{\mathsf{Redist}}$ message, all nodes delete all data except the renewed shares held by the new access structure.

EPSS Complaint Resolution Protocol

1. Coord initiates the complaint resolution by broadcasting an $\mathsf{init}_{\mathsf{Resolve}}$ 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 $(\mathcal{E}_j(s'_j), \mathcal{E}_j(t'_j))_{\mathsf{Sig_J}}$. Furthermore, node j must provide the decrypted share pair 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 (3) 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 process. Depending on the outcome, the following actions are taken: If neither of the conditions fails, the accusation by j is valid and the complaint resolution protocol resumes. If (i) and (ii) are correct, but (iii) fails or if (ii) fails (regardless of (i) and (iii)), j lodged an unwarranted accusation. The complaint by j is disregarded and Coord aborts the redistribution protocol and re-initiates it excluding node j. If (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** (valid complaint) and the complaint is seen as resolved. The protocol is re-initiated as soon as J has been rebooted and node j has been excluded.

5. If the complaint by j was valid, Coord asks 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.

6. The Jury executes the following verifications: (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 result. 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 occurred earlier. The resolution protocol resumes. If only (i) is found to fail for some I, J is marked as **VC**. The complaint is seen as resolved and the protocol is re-initiated as soon as J has been rebooted. If only (ii) is found to fail, the root J is marked as **VC**. The complaint is seen as resolved and the protocol is re-initiated as soon as J has been rebooted.

8. Coord asks each sending root I with nodes in $\mathcal{B}_{\tilde{u}}$ to reveal the signed and encrypted sub-shares it received from its children i for the computation of the partial shares destined for node j.

9. The Jury can now use this information for 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 both conditions. Again, a majority vote determines the outcome of the process. 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 child nodes in $\mathcal{B}_{\tilde{u}}$ and the resolution resumes. If only (i) is found to fail for some i, the respective root is marked as **VC**. The complaint is seen as resolved and the protocol is re-initiated as soon as all misbehaving roots have been rebooted. If only (ii) is found to fail,the respective root node is marked as **VC.** The complaint is seen as resolved and the protocol is re-initiated as soon as all misbehaving roots have been rebooted.

11. Node j is then asked to decrypt the published sub-share pairs and to verify their validity by the equation $g^{\hat{s}_{ij}} h^{\hat{t}_{ij}} \equiv g^{s_i} h^{t_i} \prod_{l=1}^{m'-1} (g^{a'_{il}} h^{b'_{il}})^{j^l}$. This allows an honest j to identify dishonest child nodes.

12. Node j then reports the dishonest nodes on the bulletin board along with the decrypted sub-shares and a *proof of correct decryption*. Thereby a dishonest j cannot accuse honest senders of cheating without being detected.

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) The equation from Step 11 is **not** satisfied.

    If all conditions are fulfilled for a sub-share pair, the concerned sender node is marked as **VC** by the respective member of the Jury. 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. Coord aborts 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 except the shares hold by the old child nodes. Coord then re-initiates the process by excluding all identified compromised nodes.

Remark. Since we want to avoid that dishonest receiver root nodes can force their child nodes to reject an otherwise valid share, receivers handle invalid signatures by their roots gracefully. More precisely, if they obtain a share pair which verifies correctly, but carries an invalid signature, they nevertheless accept and store the share pair. While we assumed that an attacker cannot spoof identities, a root should detect and discard invalid signatures. Furthermore, if after a certain amount of time the root has not received all expected messages, an authenticated cannotResume message is broadcast, listing those nodes which appear to be unresponsive. Coord can then re-initiate the protocol, taking into account the information gained from these messages.

5.3 EPSS Reconstruction

To reconstruct the original secret k from the current access structure, say $[n^*, m^*, N^*]$, the client C requests commitments from all current shareholders to their stored share. C then picks m^* shareholders with valid shares that then send their for C encrypted shares to their roots. These aggregate the shares and forward the data to the client. C reconstructs the secret and verifies this value with the original commitment to k. If this step fails, the client re-initiates the reconstruction process with varying shareholders. As for the initial distribution, if the reconstruction fails to be successful within a reasonable time frame, further investigations regarding the client and the access structure are necessary.

More precisely, to reconstruct the original secret k from the current access structure, say $[n^*, m^*, N^*]$, the client C performs the following steps:

EPSS Reconstruction

1. 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), i.e., $g^{s_i} h^{t_i}$, on the bulletin board.

3. C 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 $L = \binom{n^*}{m^*}$. It then selects the first element $\mathcal{B}_{\tilde{u}}$ where all of its members hold shares which are consistent with the original secret, i.e.,

$$g^k h^t \equiv \prod_{i \in \mathcal{B}_{\tilde{u}}} (g^{s_i} h^{t_i})^{b_i} \quad \text{where} \quad b_i = \prod_{\substack{l \in \mathcal{B}_u, \\ l \neq i}} \frac{l}{l - i} \tag{4}$$

   C publishes a $\mathsf{select}_{\mathcal{B}_{\tilde{u}}}$ message on the bullet board.[5]

4. All $i \in \mathcal{B}_{\tilde{u}}$ then encrypt their stored share pair with the public key of C 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 sent by their children $i \in \mathcal{N}_I$ and verifies the sender's signatures. It then aggregates the received data by computing the encrypted partial secret pair for C given by:

$$\mathcal{E}_C(s^{\mathcal{N}_I}) = \sum_{i \in \mathcal{N}_I} b_i \cdot \mathcal{E}_C(s_i), \quad \mathcal{E}_C(t^{\mathcal{N}_I}) = \sum_{i \in \mathcal{N}_I} b_i \cdot \mathcal{E}_C(t_i)$$

   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 C.

6. C 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. More precisely, 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:

$$g^k h^t \equiv g^{\tilde{k}} h^{\tilde{t}} \tag{5}$$

   - If Equation (5) holds, 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 except the share pairs stored by the child nodes.
   - Otherwise, 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 data except the share pairs stored by the child nodes. C then re-initiates the process.

Like for the initial distribution, if the reconstruction fails to be successful within a pre-defined time frame, further investigations regarding the client and the involved root nodes are necessary. A reboot or replacement of suspiciously behaving nodes will ultimately resolve this issue.

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

6 Communication and Security Analysis

In the following we first show how the communication complexity to a given access structure can be determined. Then we compare this estimation with a simple solution that does not use encryption nor data aggregation. Then, we summarize the security properties provided by EPSS.

6.1 Communication Anaylsis

When examining communication complexity we concentrate on the redistribution process and the case in which all participants are honest as dishonest parties can generate an arbitrary amount of traffic. Furthermore, we do not consider the case where all shareholders can be connected via private channels, since the aim of our solution is to prevent those channels due to their impracticality. Note that a private channel requires that the sender and the receiver exchange random data over another channel of the same size as the data sent over the channel. Because of this, we only estimate the amount of data sent over the remaining private channels. Assume, the redistribution takes place between $[n, m, N]$ and $[n', m', N']$, where $n = 3m - 1$ and $n' = 3m' - 1$, respectively.

Furthermore, let $\tilde{n}$ be the number of sender root nodes involved in the redistribution process. Then, root nodes on the sender side aggregate $m \cdot n'$ received sub-shares into $\tilde{n} \cdot n'$ partial shares. The receiving root nodes further combine these to n' shares. Consequently, $(m + \tilde{n} + 1) \cdot n'$ (sub-)share and partial share pairs are sent in total. Note that encrypted sub-shares, partial shares, and shares are of the same size x, e.g., 2048bit using up to date encryption schemes. It follows that $(m+\tilde{n}+1) \cdot x \cdot n'$ bits are communicated. Since $N << n$ and $n = 3m - 1$, also $(m + \tilde{n} + 1) << n$ such that an upper bound regarding the size of the data communicated can be given by $n \cdot x$.

Using a solution without aggregation the $m \cdot n'$ shares are not aggregated but simply forwarded by the root nodes on the sender and receiver side. This would lead to a total of $(m+m+m) \cdot n'$ (sub-)shares. Note that the size of the shares might be smaller than the size of the ciphertext, i.e. x, which might balance out this drawback. But this comes at the price that no privacy towards the root nodes is provided.

6.2 Security Assurances

For compactness, we only provide here the most important theorems concerning the long-term security of the EPSS redistribution protocol covering the worst case scenario and their proof sketches. For the complete results, we refer the interested reader to the thesis version [Bre16].

Theorem 1 (Integrity and Availability)**.** *The EPSS Redistribution protocol assures the integrity and availability of the secret k in the presence of fewer than threshold dishonest child nodes 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. □

Theorem 2 (Long-term Confidentiality)**.** *Let at most $m - 1$ child nodes in $\mathcal{B}_{\tilde{u}}$ and up to $m' - 2$ child nodes in $[n', m', N']$ be dishonest. Furthermore, assume that any cluster with two 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 $m'-1$ 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.

Define $\mathcal{S}$ to be the set of all dishonest receiver shareholders. Let $\bar{i} \in \mathcal{B}_{\tilde{u}}$ be dishonest and let $\bar{i}$ distribute invalid sub-shares for all $j \in [n', m', N'] \setminus \mathcal{S}$. 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 3 (Computational Confidentiality)**.** *Assume that at most $m-1$ child nodes in $[n, m, N]$ and at most $m'-2$ in $[n', m', N']$ are dishonest. Then the computational confidentiality of the secret k is assured during redistribution (even in the presence of arbitrarily many compromised root nodes).*

(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 $[n', m', N']$, 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 Conclusions

In this work we presented a novel proactive secret sharing scheme EPSS which improves the state of the art with respect to practicability and efficiency. The number of necessary private point-to-point communication links was significantly reduced while maintaining long-term confidentiality towards external attackers. Towards internal attackers confidentiality is provided computationally. Confidentiality with respect to the stored data can still be guaranteed unconditionally.

For future work we plan to combine this approach with long-term secure archiving, thereby ensuring also long-term integrity and long-term authenticity for the stored data.

Acknowledgments

We thank the anonymous reviewers of PST 2016 for their helpful comments. This work has been co-funded by the DFG as part of project Long-Term Secure Archiving within the CRC 1119 CROSSING, as part of project D.2 within the RTG 2050 Privacy and Trust for Mobile Users, and received funding from the European Union's Horizon 2020 program under Grant Agreement No 644962.

References

[BB84] 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.

[Bre16] J. Brendel. Efficient Proactive Secret Sharing. unpublished, Jan 2016. unpublished thesis.

[CKLS02] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl. Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems. In *CCS '02*, pages 88–97. ACM, 2002.

[CL02] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

[CM97] C. Cachin and U. Maurer. Unconditional security against memory-bounded adversaries. In *CRYPTO '97*, volume 1294 of *LNCS*, pages 292–306. Springer, 1997.

[DJ97] Y. Desmedt and S. Jajodia. Redistributing Secret Shares to New Access Structures and Its Applications. Technical Report ISSE-TR-97-01, 1997.

[Eke91] A. K. Ekert. Quantum cryptography based on bell's theorem. *Phys. Rev. Lett.*, 67:661–663, Aug 1991.

[Fel87] 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. IEEE, 1987.

[GG06] V. H. Gupta and K. Gopinath. An Extended Verifiable Secret Redistribution Protocol for Archival Systems. In *ARES*, pages 100–107, 2006.

[GG07] V. H. Gupta and K. Gopinath. G^2_{its} VSR: An Information Theoretical Secure Verifiable Secret Redistribution Protocol for Long-term Archival Storage. In *Security in Storage Workshop, 2007. SISW '07. Fourth International IEEE*, pages 22–33. IEEE, 2007.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 580–589, New York, NY, USA, 2001. ACM.

[HJKY95] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *CRYPTO*, volume 963 of *LNCS*, pages 339–352. Springer, 1995.

[HL09] James Heather and David Lundin. Formal Aspects in Security and Trust. chapter The Append-Only Web Bulletin Board, pages 242–256. Springer-Verlag, Berlin, Heidelberg, 2009.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[Mau93] U. Maurer. Secret key agreement by public discussion from common information. *IEEE Information Theory*, 39(3):733–742, 1993.

[Ped92] T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO '91*, pages 129–140. Springer, 1992.

[Rab05] M.O. Rabin. Provably unbreakable hyper-encryption in the limited access model. In *IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security*, pages 34–37, 2005.

[Sha79] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[SHF+14] K. Shimizu, T. Honjo, M. Fujiwara, T. Ito, K. Tamaki, S. Miki, T. Yamashita, H. Terai, Z. Wang, and M. Sasaki. Performance of Long-Distance Quantum Key Distribution Over 90-km Optical Links Installed in a Field Environment of Tokyo Metropolitan Area. *Journal of Lightwave Technology*, 32(1):141–151, 2014.

[SLL10] D. Schultz, B. Liskov, and M. Liskov. MPSS: Mobile Proactive Secret Sharing. *ACM Trans. Inf. Syst. Secur.*, 13(4):34:1–34:32, 2010.

[VBC+15] 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, 2015.

[Ver26] G.S. Vernam. Cipher Printing Telegraph Systems For Secret Wire and Radio Telegraphic Communications. *American Institute of Electrical Engineers*, XLV:295–301, 1926.

[WWW02] T.M. Wong, C Wang, and J.M. Wing. Verifiable Secret Redistribution for Archive Systems. *Security in Storage Workshop*, 0:94, 2002.

[ZSVR05] 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, 2005.