Authenticated Share Renewal for Proactive Secret Sharing: Technical Report

Giulia Traverso, Lucas Schabhüser, Denis Butin, Johannes Buchmann

Technische Universität Darmstadt, Germany

Abstract. Long-term confidentiality can be realised through proactive secret sharing. Authenticity is often required in addition to confidentiality, and should be verifiable without reconstructing the initial data from the secret shares. This requires signing the secret shares themselves. Share renewal normally requires the data owner to become active in order to prolong authentication by signing the new shares. Since the long-term availability of the data owner is uncertain, it is desirable to enable the extension of authenticity without intervention of the data owner. In this paper, we provide a solution for a coarse-grained solution where authenticity can be checked only when reconstructing the data. We formalise for the first time a long-term authenticity scenario where authenticity can be verified throughout the storage of the data. We discuss attempts at instantiating such fine-grained solutions pointed out why they do not satisfy our scenario.

1 Introduction

Sensitive data, such as health records, must often be securely stored over long time periods, e.g. decades. In particular, data owners requires the data to remain confidential over lengthy periods. Conventional cryptography is not sufficient for this requirement, due to the unpredictability of cryptanalytic progress. *Secret sharing* [5, 15, 17] can provide information-theoretic confidentiality, which is not vulnerable to cryptanalytic progress. In secret sharing, the data owner distributes the data among shareholders. The initial data can only be reconstructed when a sufficient number of shares are available. An even stronger notion of confidentiality can be ensured through *proactive secret sharing* [11], where the secret shares are periodically renewed by the shareholders. This prevents an attacker from reconstructing the secret by slowly collecting shares over time, since the shares are then only valid until their next renewal.

In addition to confidentiality, a common requirement regarding the stored data is authenticity: an assurance linking the data to the data owner. Authenticity can be achieved by having the data owner sign the data once before it is distributed through secret sharing. The signature remains valid upon share renewal since the reconstructed data remains identical. A drawback of this approach is that authenticity can only be checked globally: if a shareholder misbehaves, authentication failure cannot be traced back to this specific shareholder.

A more fine-grained approach to authenticity is possible if the data owner signs each share. However, in proactive secret sharing, share renewal must be possible without intervention of the data owner, whose long-term availability can not be guaranteed. This setting raises two challenges. First, to provide a meaningful definition of authenticity after share renewal. Second, to construct a protocol for proactive secret sharing that ensures authenticity of the shares in this new sense.

Contributions

- We provide a solution to the problem of share renewal over authenticated data for coarse-grained authenticity verifiability. More precisely, we provide directions for how to perform proactive secret sharing when the shared document has also to be signed once it is reconstructed.
- We provide a framework and a formal security model for a solution providing fine-grained authenticity verifiability. More precisely, we formalise a number of notions for the scenario when it is required to have a finer-grained approach to authenticity so as to blame the shareholder(s) that compromised it. First, we define the notion of outsourced data in distributed storage systems based on secret sharing schemes. Second, the initial data can be reconstructed and the shares of the honest shareholders can be authenticated, as long as the number of dishonest shareholders does not exceed the threshold of the proactive secret sharing scheme. Third, we define a forgery as a (share, signature) pair, where the share is a valid share for the initial data and the signature is a valid signature over this share. Fourth, we require privacy in the sense that no information about previous shares can be obtained from new shares and from signatures over new shares.
- We survey the state of the art of possible candidates for the fine-grained solution discussed above. We argue why the state of the art is currently insufficient to provide such solution by displaying drawbacks and shortcomings of schemes that would utilize such primitives.

Outline The remainder of this paper is organised as follows. We start with preliminaries on secret sharing and proactive secret sharing (Section 2). Then, we present a coarse-grained solution for the problem of authenticated share renewal (Section 3). We introduce a framework and security model for a fine-grained solution for authenticated share renewal (Section 4). Next, we discuss our attempts at building an alternative solution for the fine-grained scenario and display their shortcomings (Section 5). Conclusions follow (Section 6).

2 Preliminaries

In this section, we first introduce secret sharing schemes as a means to build distributed storage systems for the outsourcing of data (Section 2.1). Then, we discuss proactive secret sharing for share renewal to prolong the confidentiality of distributed data (Section 2.2).

2.1 Secret Sharing and Distributed Storage Systems

Distributed storage systems allows for protected outsourcing of data. Shares of data are distributed to the storage servers owned by multiple commercial cloud service providers so as to achieve at the same time confidentiality and fast retrieval of the data. These two properties are enabled by secret sharing, the cryptographic primitive underlying distributed storage systems. Secret sharing schemes are used to share a message $d \in \mathbb{F}_q$ across a set $S = \{s_1, \ldots, s_n\}$ of n shareholders, where shareholders correspond to the storage servers in distributed storage systems. More precisely, a dealer generates shares $\sigma_1, \ldots, \sigma_n \in \mathbb{F}_q$ of message d and distributes each share $\sigma_i \in \mathbb{F}_q$ to the respective shareholder $s_i \in S$. Only specific subsets $A \subset S$ of shareholders can reconstruct the message provided that certain requirements are fulfilled. Instead, subsets $U \subset S$ not fulfilling such requirements cannot reconstruct the message and get no information about it. These subsets are called *authorized* and *unauthorized*, respectively. Denoting by $\mathcal{P}(S)$ the power set of S, the *access structure* $\Gamma \subset \mathcal{P}(S)$ determines both sets, i.e. $A \in \Gamma$ and $U \notin \Gamma$. More formally, secret sharing is a pair of algorithms (Share, Reconstruct). Algorithm Share takes as input a message $d \in \mathbb{F}_q$ and the unique ID $i \in \mathcal{I}$ of shareholder $s_i \in S$ and outputs its share $\sigma_i \in \mathbb{F}_q$, for $i = 1, \ldots, n$. Algorithm Reconstruct takes as input a subset $R \subset S$ and outputs the message d if $R \in \Gamma$ and $\perp$ otherwise. Adapting the definition provided in [1] to the purpose of this technical report, in the following we formalize the notions of correctness and perfect secrecy restricted to ideal secret sharing schemes, where the length of the message equals the length of the shares.

Definition 1. *Given the message space $\mathbb{F}_q$ and the set $S = \{s_1, \ldots, s_n\}$ of shareholders, the pair of algorithms* (Share, Reconstruct) *is a secret sharing scheme realizing access structure $\Gamma \subset \mathcal{P}(S)$ if the following two requirements hold:*

1) Correctness*: if shares held by shareholders of an authorized set $A \in \Gamma$ are given as input to algorithm* Reconstruct*, then algorithm* Reconstruct *retrieves the message d shared during algorithm* Share*, for every message $d \in \mathbb{F}_q$.*
2) Perfect secrecy*: if shares held by shareholders of an unauthorized set $U \notin \Gamma$ are given as input to algorithm* Reconstruct*, then algorithm* Reconstruct *leaks no information about the message d shared during algorithm* Share*, for every message $d \in \mathbb{F}_q$.*

Linear threshold secret sharing schemes are amongst the most studied schemes because they are easily implemented for application scenarios such as distributed storage systems. The first (t, n)-threshold secret sharing scheme is due to Shamir [15] and it is based on interpolation of polynomials. More precisely, a message d is shared using a polynomial $f(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1}$ of degree $\deg(f(x)) = t - 1$, where $a_0 := d$ and coefficients $a_1, \ldots, a_{t-1} \in \mathbb{F}_q$ are chosen uniformly at random. Algorithm Share computes share $\sigma_i \in \mathbb{F}_q$ for shareholder $s_i \in S$ as a point on polynomial $f(x)$, i.e. $\sigma_i := f(i)$, where $i \in \mathcal{I}$ is the ID of shareholder s_i. Algorithm Reconstruct is based on Lagrange interpolation of polynomials. Thus, on the one hand, authorized subsets $A \subset S$ are composed

of t or more shareholders, that is $|A| \geq t$. When t or more points of polynomial $f(x)$ are collected, it is possible to correctly interpolate polynomial $f(x)$ and message d is retrieved as $f(0) = a_0$. On the other hand, unauthorized subsets $U \subset S$ are composed of $t-1$ or less shareholders, that is $|U| \leq t-1$. When only $t-1$ or less points are collected, polynomial $f(x)$ cannot be reconstructed and no information about message $d \in \mathbb{F}_q$ is leaked.

Shamir secret sharing scheme is not the only cryptographic primitive on which distributed storage systems are built on. Hierarchical secret sharing schemes [8], [5] address scenarios where the shareholders are not equal in their reconstruction capability, such as in companies. In the context of distributed storage systems, hierarchical secret sharing schemes can be used to grant higher quality shares to the most trustworthy storage servers in terms of availability and computational performance [18], [13], [20].

2.2 Proactive Secret Sharing and Long-Term Confidentiality

The long-term confidentiality protection of outsourced data is threatened in distributed storage systems by the so called mobile adversary. A mobile adversary is not bounded by the amount of storage servers it can break into over time. Given that the storage of data can last for decades, this means that a mobile adversary has enough time to eventually collect enough shares as to reconstruct the data by itself. To prevent this, proactive secret sharing [11] is used as a countermeasure. Proactive secret sharing is a protocol with which shareholders refresh their shares in a distributed fashion without requiring the intervention of the data owner. The shares are renewed in such a way that they are independent of the old shares and still reconstruct to the same document. Furthermore, no information about the old shares is leaked during proactive secret sharing. Thus, if this process is performed often enough, a mobile adversary doesn't have enough time to collect the required number of shares for data reconstruction. In the following, we provide a description of how proactive secret sharing is performed for Shamir secret sharing scheme according to the scheme proposed in [11]. This will be useful to understand the roadblocks of the fine-grained authenticity verification problem for share renewal over authenticated data in Section 5.

Each shareholder s_i holding share $\sigma_i = f(i)$ for a polynomial $f(x) = a_0 + a_1 x + \ldots + a_{t-1}x^{t-1}$ for document d computes a fresh share σ_i' by performing the following steps, for $i = 1, \ldots, n$.

- It selects a polynomial $f_i(x) = a_{0,i} + a_{1,i}x + \ldots + a_{t-1,i}x^{t-1}$, where $a_{0,i} := 0$ and coefficients $a_{1,i}, \ldots, a_{t-1,i} \in \mathbb{F}_q$ are chosen uniformly at random.
- It computes sub-share $\sigma_{j,i}. = f_i(j)$ and distributes it to shareholder $j \neq i$ though a private channel and keeps sub-share $\sigma_{i,i} := f_i(i)$ for itself.
- It collects al sub-shares $\sigma_{i,j}$ received by shareholder $j \neq i$ as well as sub-share $\sigma_{i,i}$ and computes the fresh share σ_i' as $\sigma_i' := \sigma_i + \sum_{j=1}^{n} \sigma_{i,j}$.

Proactive secret sharing can be performed similarly also for hierarchical secret sharing [18]. Thus, the matter of share renewal over authenticated data concerns

both distributed storage systems built over Shamir's secret sharing scheme and hierarchical secret sharing schemes. However, for the purpose of this technical report in the following we will only consider solutions based on Shamir's secret sharing scheme.

3 Authenticated Share Renewal with Coarse-Grained Authenticity Verifiability

The question of how to accomplish share renewal over authenticated data belongs, more in general, to the research problem of having a distributed storage system where the primitives involved can be renewed without the intervention of the data owner. In particular, for authenticated data, the two main primitives involved are secret sharing and digital signature schemes. In this technical report, we focus on how to renew the shares given that the signature does not change. We do not consider here the problem of prolonging the security or of renewing the signature scheme used to authenticate the data without the intervention of the data owner, which is still an open problem [4]. Instead, we consider authenticated share renewal within the time window in which the signature scheme used is secure.

In order to perform share renewal over authenticated data, the data owner performs the following steps. First, it authenticates the data it wants to outsource to a distributed storage system by using a secure signature scheme. Second, it runs algorithm Share described in Section 2 and generates shares of the authenticated data by using (t, n)-Shamir's threshold secret sharing scheme. It chooses the integers t and n depending on the distributed storage system to be built. Third, it distributes each share to each of the storage servers and then leaves the system. The shares are not any more simply shares of the data, but they are shares of the signed document. When proactive secret sharing is performed by the storage servers in a distributed fashion, the refreshed shares still reconstruct to the signed document. Thus, when the data owner wants to reconstruct the outsourced data, it can simply run algorithm Reconstruct described in Section 2 to reconstruct the authenticated data. Integrity of the outsourced data is verified if the retrieved document corresponds to the secret-shared one.

4 A Framework for Authenticated Share Renewal with Fine-Grained Authenticity Verifiability

The coarse-grained solution proposed in Section 3 has the drawback that integrity can be verified only when reconstructing the outsourced data and the storage server providing an incorrect share cannot be detected. In this section, we aim at constructing the framework for a fine-grained authenticity verification solution to the problem of share renewal over authenticated data. That is, a solution where integrity can be verified in a finer-grained manner, so that the storage servers providing inconsistent shares can be detected and, if necessary, kicked out of

the distributed storage system. In the following, Section 4.1 introduces CSRAS, a scheme describing the protocols and the actors involved for the fine-grained authenticity verifiability problem. Afterwards, Section 4.2 formalizes the security model for the CSRAS scheme.

4.1 Definition of a Scheme for Confidential Share Renewal with Fine-Grained Authenticity Verifiability

Before introducing the CSRAS scheme, we need to redefine the notion of authenticated data outsourced to a distributed storage system. In distributed storage systems, data shares are generated through secret sharing and each share is distributed to a different storage server. That is, the notion of authentication of outsourced data corresponds to the authentication of secret-shared data.

Definition 2. *We denote by d the data that is outsourced to a distributed storage system. The data d is secret-shared, i.e. shares $s_1, \ldots, s_n$ are computed through a secret sharing scheme. Each share is initially signed by a signature scheme with the data owner's private key and signatures $\sigma_1, \ldots, \sigma_n$ are computed for, respectively, shares $s_1, \ldots, s_n$. We refer to the set of pairs $(s_1, \sigma_1), \ldots, (s_n, \sigma_n)$ as* authenticated secret-shared data*.*

The rationale for this formalisation is that, by signing each share of the data, the shareholders (in this case the storage servers) can always check the validity of the origin of the shares. One might argue that the data itself is never authenticated and that in order for this to happen one should first sign the data and then perform secret sharing over the signed data. However, the whole document is never available or physically present somewhere throughout its long-term storage. Thus, for the shareholders to be able to check the origin of the data, they must reconstruct it first. Instead, what Def. 2 achieves is that at any point in time it is always possible to check the origin of the piece of information, i.e. the share, that is indeed stored.

A *Confidential Share Renewal Authenticity Scheme* (CSRAS) is composed of the following algorithms:

- $\mathsf{Setup}(n) \rightarrow \mathsf{ID}$. On input the number of shareholders n, this algorithm outputs the public set ID of shareholders' identities, i.e. $\mathsf{ID} = \{id_1, \ldots, id_n\}$.
- $\mathsf{KeyGen}(1^\lambda, \mathsf{ID}) \rightarrow ((\mathsf{sk_D}, \mathsf{pk_D}), (\mathsf{sk_1}, \mathsf{pk_1}), \ldots, (\mathsf{sk_n}, \mathsf{pk_n}))$. On input a security parameter λ and the public set of identities ID, this algorithm outputs the private/public key pair $(\mathsf{sk_D}, \mathsf{pk_D})$ for the data owner and the private/public key pairs $(\mathsf{sk_1}, \mathsf{pk_1}), \ldots, (\mathsf{sk_n}, \mathsf{pk_n})$ for the shareholders.
- $\mathsf{Outsource}(d, \mathsf{sk_D}, \mathsf{ID}) \rightarrow ((\mathsf{s_1}, \sigma_1), \ldots, (\mathsf{s_n}, \sigma_n), \tau, \theta)$. On input a data d, the private signing key $\mathsf{sk_D}$ of the data owner, and the set of shareholder identities ID, this algorithm outputs the authenticated secret-shared data, i.e. the pairs of signed shares $((s_1, \sigma_1), \ldots, (s_n, \sigma_n))$, a document identifier τ as well as a state identifier θ, initialised with the value 0. $\mathsf{Outsource}$ is run by the data owner.

– $\mathsf{Renew}((s_1, \sigma_1), \ldots, (s_n, \sigma_n), \mathsf{sk}_1, \ldots, \mathsf{sk}_n, \theta) \rightarrow ((\mathsf{s}'_1, \sigma'_1), \ldots, (\mathsf{s}'_n, \sigma'_n), \theta{+}1)$. On input an authenticated secret-shared data $(s_1, \sigma_1), \ldots, (s_n, \sigma_n)$ the private signing keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ of the shareholders, and a state identifier θ, this algorithm outputs renewed shares with corresponding fresh signatures $(s'_1, \sigma'_1), \ldots, (s'_n, \sigma'_n)$ and the incremented state identifier $\theta + 1$. Renew is run by the shareholders.
– $\mathsf{Verify}(s_i, \sigma, \mathsf{pk_D}, \mathsf{id_i}, \tau, \theta) \rightarrow \mathsf{b}$. On input a share s_i, a signature σ, the data owner's public verification key $\mathsf{pk_D}$, the identity of the shareholder owning s_i, a document identifier τ, and a state identifier θ, this algorithm outputs $b = 1$ (accept) or $b = 0$ (reject). Verify is run by a third party.

The algorithm Verify described above is actually performed by multiple third parties verifiers. The reason is that the shares that are checked are in clear so as that the validity of the corresponding signature can also be verified. We can assume that the data owner relies on multiple auditors such that none of them gets to verify more than $t - 1$ shares. This way, confidentiality of the outsourced data is preserved due to the information theoretic confidentiality of the underlying (t, n)-threshold secret sharing scheme.

4.2 Security Model

In the following, we formalise the security model for the CSRAS scheme presented in Section 4.1. More precisely, we formalise what a forgery is and what unforgeability means, and the notion of privacy.

We now define forgery by an adversary for the CSRAS scheme described above.

Definition 3. *A* forgery *for a CSRAS scheme is a tuple* $(s^*, \sigma^*, \mathsf{id}^*, \tau^*, \theta^*)$*, such that* $\mathsf{Verify}(\mathsf{s}^*, \sigma^*, \mathsf{pk}, \mathsf{id}^*, \tau^*, \theta^*) = 1$ *and one of the following conditions holds:*

– *The data owner corresponding to the public key* pk *never output a message identifier* τ^**, when calling* Outsource*.*
– *There exists an authorized set A, with* $\mathsf{id}^* \in \mathsf{A}$*, such that calling* Reconstruct *with input A during time period* θ *does not output data d identified by the data identifier* τ^**.*

We formalize the notion of unforgeability in the form of an experiment between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.

$\mathsf{AuthShare} - \mathsf{UF} - \mathsf{CMA}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{t})$**:**

Key Generation: $\mathcal{C}$ calls $\mathsf{Setup}(n)$ to produce a set of identities ID. It proceeds to call $\mathsf{KeyGen}(1^\lambda, \mathsf{ID})$ to output key pairs $(\mathsf{sk_D}, \mathsf{pk_D}), (\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_n, \mathsf{pk}_n)$ for the data owner and shareholders. It gives $\mathsf{ID}, \mathsf{pk_D}, \mathsf{pk}_1, \ldots, \mathsf{pk}_n$ to the adversary $\mathcal{A}$. It sets $\theta = 0$. It initializes an empty list $L = \emptyset$.

For each time period $\theta = 0, \ldots poly(\lambda)$ the following steps occur:

Corruption: $\mathcal{A}$ adaptively chooses a subset $B \subset \mathsf{ID}$ of size $|B| \leq t$. $\mathcal{C}$ gives $\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$ for all $\mathsf{id} \in \mathsf{B}$. Then $\mathcal{C}$ gives all tuples $(s_{\mathsf{id}}, \sigma_{\mathsf{id}}, \tau)$ to $\mathcal{A}$ for which $\tau \in L$.

Queries: $\mathcal{A}$ adaptively submits queries for different data d. The challenger $\mathcal{C}$ proceeds as follows. It calls $\mathsf{Outsource}(d, \mathsf{sk}_{\mathsf{D}}, \mathsf{ID}\ )$ to obtain $((s_1, \sigma_1), \ldots, (s_n, \sigma_n), \tau, \theta)$. It sends $(\tau, s_{\mathsf{id}}, \sigma_{\mathsf{id}})$ to $\mathcal{A}$ for all $\mathsf{id} \in \mathsf{B}$ and sets $L \leftarrow L \cup \{\tau\}$.

Renewal: $\mathcal{C}$ sets $\theta = \theta + 1$ and calls $\mathsf{Renew}((s_1, \sigma_1), \ldots, (s_n, \sigma_n), \mathsf{sk}_1, \ldots, \mathsf{sk}_{\mathsf{n}}, \theta)$ for all tuples of shares and signatures associated to $\tau \in L$.

Forgery: $\mathcal{A}$ outputs a tuple $(s^*, \sigma^*, \mathsf{id}^*, \tau^*, \theta^*)$. The experiment outputs $\mathtt{1}$, if $(s^*, \sigma^*, \mathsf{id}^*, \tau^*, \theta^*)$ is a forgery according to Def. 3 and else outputs $\mathtt{0}$.

Definition 4 (Unforgeability). *A CSRAS scheme $\mathcal{S}$ is* (t, n) unforgeable *if for any PPT adversary $\mathcal{A}$, $Pr[\mathsf{AuthShare} - \mathsf{UF} - \mathsf{CMA}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{t}) = 1] = \mathsf{negl}(\lambda)$, where $\mathsf{negl}(\lambda)$ denotes any function negligible in the security parameter λ.*

We formalize the notion of privacy in the form of an experiment between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.

$\mathsf{AuthShare} - \mathsf{Privacy}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{t})$**:**

Key Generation: $\mathcal{C}$ calls $\mathsf{Setup}(n)$ to produce a set of identities ID. It proceeds to call $\mathsf{KeyGen}(1^{\lambda}, \mathsf{ID})$ to output key pairs $(\mathsf{sk}_{\mathsf{D}}, \mathsf{pk}_{\mathsf{D}}), (\mathsf{sk}_1, \mathsf{pk}_1), \ldots, (\mathsf{sk}_{\mathsf{n}}, \mathsf{pk}_{\mathsf{n}})$ for the data owner and shareholders. It gives $\mathsf{ID}, \mathsf{pk}_{\mathsf{D}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_{\mathsf{n}}$ to the adversary $\mathcal{A}$. It sets $\theta = 0$. It chooses $b \xleftarrow{\$} \{0, 1\}$.

Chosen Messages: $\mathcal{A}$ chooses d_0, d_1 and gives them to $\mathcal{C}$. $\mathcal{C}$ runs $((s_1, \sigma_1), \ldots, (s_n, \sigma_n), \tau, \theta) \leftarrow \mathsf{Outsource}(d_b, \mathsf{sk}_{\mathsf{D}}, \mathsf{ID})$

For each time period $\theta = 0, \ldots, poly(\lambda)$, the following steps occur:

Corruption: $\mathcal{A}$ adaptively chooses a subset $B \subset \mathsf{ID}$ of size $|B| \leq t$. $\mathcal{C}$ gives $\mathsf{sk}_{\mathsf{id}}$ to $\mathcal{A}$ for all $\mathsf{id} \in \mathsf{B}$. Then $\mathcal{C}$ gives $(s_{\mathsf{id}}, \sigma_{\mathsf{id}}, \tau)$ to $\mathcal{A}$.

Renewal: $\mathcal{C}$ sets $\theta = \theta + 1$ and calls $\mathsf{Renew}((s_1, \sigma_1), \ldots, (s_n, \sigma_n), \mathsf{sk}_1, \ldots, \mathsf{sk}_{\mathsf{n}}, \theta)$ for all for all tuples of shares and signatures associated to $\tau \in L$.

Forgery: $\mathcal{A}$ outputs a bit $b^* \in \{0, 1\}$. The experiment outputs $\mathtt{1}$, if $b = b^*$ and else outputs $\mathtt{0}$.

Definition 5 (Hiding). *A CSRAS scheme $\mathcal{S}$ is perfectly* (t, n) hiding *if for any adversary $\mathcal{A}$, $Pr[\mathsf{AuthShare} - \mathsf{Privacy}_{\mathcal{A}}(\lambda, \mathsf{n}, \mathsf{t}) = 1] = \frac{1}{2}$*

5 Attempts at Building an Alternative Solution

In this section, we discuss our attempts to build an alternative solution for the CSRAS defined in Section 4.1. In Section 5.1, Section 5.2, Section 5.3, and Section 5.4 our attempts with respect to, respectively, homomorphic signature schemes, aggregate signature schemes, multi-key homomorphic authenticator schemes, and threshold signature schemes are discussed. In Section 5.5, we discuss our attempt with respect to publicly verifiable secret sharing is presented.

5.1 Homomorphic Signature Schemes

In the following, we discuss *homomorphic signature schemes* [12] and show how they do not provide a solution for our fine-grain authenticity verification problem. More precisely, first we discuss why using homomorphic signature schemes is not sufficient. Then, we propose a modification that mitigates the drawbacks of homomorphic signature schemes but that still does not meet our privacy requirements.

Homomorphic signature schemes allow for computations over authenticated data. This property makes them interesting for many applications, such as smart grids, electronic voting and, of course, distributed storage systems [19]. More precisely, homomorphic signature schemes allow anyone to generate the signature of a message, which is the result of a computation over messages previously signed by the data owner. The current state of the art homomorphic signature schemes allow to perform both linear operations and multiplications over authenticated data. We refer to [19] for an extensive survey on this topic. In the following, we provide a formal definition of homomorphic signature schemes.

Definition 6. [10] *A* homomorphic signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*

- $\mathsf{Setup}(1^\lambda)$. *It takes as input a security parameter* λ *in unary. It outputs a secret key* sk *and the respective public key* pk.
- $\mathsf{Sign}(\mathsf{sk}, \tau, \mathsf{m}, \mathsf{i})$. *It takes as input a secret key* sk, *a tag* $\tau \in \{0,1\}^\lambda$, *a message* m, *and an index* i. *It outputs a signature* σ, *computed using the secret key* sk, *which is the signature for the* i*-th message* m *of the data set tagged by* τ.
- $\mathsf{Evaluate}(\mathsf{pk}, \tau, \mathsf{f}, \overrightarrow{\sigma})$. *It takes as input a public key* pk, *a tag* $\tau \in \{0,1\}^\lambda$, *a function* f *and a tuple of signatures* $\overrightarrow{\sigma}$. *It outputs a signature* σ', *output of a function* f *over the (tuple of) signatures* $\overrightarrow{\sigma}$. *Such tuple* $\overrightarrow{\sigma}$ *corresponds to the signatures on the messages within the data set labeled by tag* $\tau \in \{0,1\}^\lambda$.
- $\mathsf{Verify}(\mathsf{pk}, \tau, \mathsf{m}, \sigma, \mathsf{f})$. *It takes as input a public key* pk, *a tag* $\tau \in \{0,1\}^\lambda$, *a message* m, *a signature* σ, *and a function* f. *It outputs* 1 *if* σ *is a valid signature for the message* m. *Such message* m *is the output of the function* f *over the data set tagged by* τ, *whose messages are signed using the public key* pk. *It outputs* 0 *otherwise.*

Homomorphic signature schemes were the first primitive we looked into for an instantiation of the CSRAS scheme presented in Section 4.1. Because of the fact that such schemes allow to perform computations on signed shares, they are suitable for performing the reconstruction algorithm of secret sharing schemes. This property matches Definition 2 of outsourced data: each share is signed by the data owner by its secret key before being distributed to the shareholders. When the data has to be retrieved, linearly homomorphic signature schemes allow for Lagrange interpolation as the function to be evaluated and the authenticated data can be computed. However, homomorphic signature schemes allow computations on messages that have been authenticated with the same key only. As discussed in Section 2.2, during proactive secret sharing sub-shares are

computed independently by each shareholder and distributed to each other. These sub-shares have to be authenticated by the shareholder that had computed them with its own secret signing key. When the refreshed shares are computed, the old signed share and the sub-shares cannot be added together because they have been signed with different keys. Thus, it is not possible to obtain authenticated refreshed shares.

A possible modification of homomorphic signature schemes when used together with proactive secret sharing allows to compute authenticated refreshed shares as follows. The data owner computes the shares and sign them by using a homomorphic signature scheme. It distributes both the shares and the signatures to the designated shareholders. When proactive secret sharing is run, each shareholder computes the sub-shares and signs each of them by using the aforementioned homomorphic signature scheme. Then, it distributes the sub-shares and the signed sub-shares to the respective shareholders. Each shareholder computes its refreshed share as usual, i.e. by adding the sub-shares received and then one it computed for itself to the share distributed originally by the data owner. The signature of the refreshed share is obtained by appending the signatures of the sub-shares to the signature of the share distributed by the data owner. This means that the size of the signature of the refreshed share is $n + 1$ times larger than the size of the original signature. This size does not increase the next time proactive secret sharing is run. More precisely, given that the set of the shareholders remains unchanged, the homomorphic property of the scheme used to add the signed sub-shares from the same shareholder. i.e. the sub-shares signed with the same secret key. However, this approach does not provide the desired privacy requirements. The reason is that, in order for a third party auditor to check authenticity of the refreshed shares, it has to check each of the $n + 1$ signatures with the corresponding share or sub-share. This means that the (sum of) sub-shares from the same shareholder and the share distributed by the data owner must be stored individually in order to allow the verification procedure. This makes the approach vulnerable to a mobile adversary because, despite the fact that proactive secret sharing is run, it can always access to the original shares in case it breaks into the storage servers. By breaking into enough storage servers, it will be able to reconstruct the data by itself. Since this approach cannot be used in a framework involving proactive secret sharing and, thus, cannot be used to instantiate the CSRAS, we did not fully analyze its properties with respect to authenticity and integrity.

5.2 Aggregate Signature Schemes

Aggregate signature schemes [3] address the multi-user scenario as they combine multiple signatures from different users combines into a single one. More precisely, given n authenticated messages signed by, respectively, n secret-public key pairs, then aggregate signature schemes compute a single signature for all the messages. Moreover, once the aggregate signature is computed, it is always possible to aggregate a further signature on another message without starting the process from scratch.

Definition 7. [3] *An* aggregate signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*

- $\mathsf{Setup}(1^\lambda)$. *It takes as input a security parameter* λ *in unary. It outputs a secret-public key pair* $(\mathsf{sk_i}, \mathsf{pk_i})$ *for each user* i.
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{m}, \mathsf{i})$. *It takes as input a secret key* sk*, a message* m*, and an index* i*. It outputs a signature* σ*, which is the signature for the* i*-th message* m*, by means of the* i*-th secret key* sk.
- $\mathsf{Verify}(\overrightarrow{\mathsf{pk}}, \mathsf{m}, \sigma)$. *It takes as input the public keys' string* $\overrightarrow{\mathsf{pk}}$*, a message* m*, and a signature* σ*. It outputs* 1 *if* σ *is a valid signature for the message* m*, signed using the public keys* $\overrightarrow{\mathsf{pk}}$*. It outputs* 0 *otherwise.*
- $\mathsf{Aggregate}_\sigma(\overrightarrow{\mathsf{pk}}, \overrightarrow{m}, \overrightarrow{\sigma})$. *It takes as input a public keys' string* $\overrightarrow{\mathsf{pk}}$*, a messages's string* $\overrightarrow{m}$*, and a signatures' string* $\overrightarrow{\sigma}$*. It outputs a signature* σ_{agg}*, which is the aggregate signature of the signatures in* $\overrightarrow{\sigma}$ *of the messages in* $\overrightarrow{m}$*, signed using the public keys in* $\overrightarrow{\mathsf{pk}}$*, respectively.*

We have looked into aggregate signature schemes because they support different messages signed by different users. This suits our finer-grained scenario for authenticated share renewal because of the following reason. During proactive secret sharing, each shareholder computes a sub-share for itself and a sub-share for each of the other $n-1$ shareholders involved in the protocol. In order to have integrity of the refreshed shares, one has first to get integrity from the sub-shares that make up such refreshed share. Using aggregate signature schemes, each shareholder can authenticate the sub-share they computed before distributing them to the designated shareholders. Each shareholder authenticated by means of its private signing key. Then the shareholders, after receiving all the signed sub-shares, can aggregate all the signatures into a single one. However, one more step for proactive secret sharing to be carried out is that these sub-shares and the old share (signed by the data owner) have to be summed in order to get the refreshed share. Aggregate signature schemes do not support operations on messages and thus a signature that is the result of operations among other signatures cannot be obtained.

5.3 Multi-Key Homomorphic Authenticators

Multi-key homomorphic authenticators [9] are homomorphic authenticators that also allow for computations over messages signed with different secret keys. They have been introduced mainly to fit scenarios such as ubiquitous computing and sensor network, where large datasets consist of data provided by different users. In the following, we provide a formal definition of multi-key homomorphic authenticators.

Definition 8. [9] *A* multi-key homomorphic authenticator scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*
$\mathsf{Setup}(1^\lambda)$. *It takes as input a security parameter* λ *in unary. It outputs a key triple* $(\mathsf{sk}, \mathsf{ek}, \mathsf{vk})$*, where* sk *is a secret authentication key,* ek *is a public evaluation key, and* vk *is a verification key which could be either public or private.*

$\mathsf{Sign}(\mathsf{sk}, \tau, \mathsf{m}, \mathsf{i})$. *It takes as input a secret key* sk*, a tag* $\tau \in \{0,1\}^\lambda$*, a message* m*, and an index* i*. It outputs an authenticator* σ*.*
$\mathsf{Evaluate}(\mathsf{f}, (\overrightarrow{\sigma}, \overrightarrow{\mathsf{ek}}))$. *It takes as input a function* f *and a tuple of authenticators* $\overrightarrow{\sigma}$ *and of evaluation keys* $\overrightarrow{\mathsf{ek}}$*. It outputs a signature* σ'*, output of a function* f *over the (tuple of) signatures* $\overrightarrow{\sigma}$*.*
$\mathsf{Verify}(\mathcal{P}, \tau, \{vk_i\}_{i\in\mathcal{P}}, m, \sigma)$. *It takes as input a labeled program* $\mathcal{P} = (f, \overrightarrow{i})$*, where* f *is a function and* $\overrightarrow{i}$ *is a tuple of indexes, a tag* $\tau \in \{0,1\}^\lambda$*, a set of verification keys* $\{vk_i\}_{i\in\mathcal{P}}$*, a message* m*, an authenticator* σ*. It outputs* 1 *if* σ *is a valid signature for the message* m*. It outputs* 0 *otherwise.*

We have investigated multi-key homomorphic authenticators for their capability of providing computations on data in a multi-user scenario, such as the one of distributed storage systems where each storage server is regarded as a different user. This is a key feature for our fine-grained authenticated share renewal scenario, because signed sub-shares from different shareholders can be summed to the signed old share from the data owner to compute the refreshed share and these operations can be mirrored on the signatures too. However, to the best of our knowledge, the state of the art multi-key homomorphic authenticators do not provide a security level that is sufficient for our scenario. More precisely, they are not resilient against collusions among parties, which might occur in our scenario as this is a key feature of threshold secret sharing schemes.

5.4 Threshold Signature Schemes

Threshold signature schemes [2] is one of the primitives of the so-called (t, n)-threshold cryptography approach. In particular, the idea behind threshold signature schemes is to distribute to n parties secret a secret key and a signature on a message to compute so that the single point of failure is removed. Given the threshold t, this means that any subset of at least t parties can use their share of the secret key to compute a valid signature for a certain message that can be verified under the public key corresponding to the distributed secret signing key. The security notion of threshold signature schemes is that a polynomial-time adversary that corrupts up to $t-1$ parties cannot get any information about the secret signing key and, thus, cannot compute a forge. Moreover, threshold signature schemes allow for proactiveness: this means that the shares of the secret signing key can be periodically refreshed without the intervention of the dealer. Thus the signature scheme can cope with a mobile adversary.

Definition 9. *Given a signature scheme* $\Sigma = (\mathsf{Setup}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Verify}_\Sigma)$*, the corresponding* threshold signature scheme *is a tuple of the following probabilistic, polynomial-time algorithms:*
$\mathsf{ThSetup}(1^\lambda)$. *It takes as input a security parameter* λ *in unary. It outputs a public key* pk *and shares* $x_1, \ldots, x_n$ *reconstructing through a* (t, n) *secret sharing scheme to the secret key* sk *corresponding to* pk*. This algorithm is run by the* n *parties.*

$\mathsf{ThSign}(\mathsf{sk}_{|A|}, \mathsf{m})$. *It takes as input the shares of a subset* $|A| \geq t$ *of parties, denoted by* $\mathsf{sk}_{|A|}$*, and a message* m*. This algorithm outputs a signature* σ *for the message* m*. It is run by the subset* $|A|$ *of parties.*
$\mathsf{ThVerify}(\mathsf{pk}, \mathsf{m}, \sigma)$. *It takes as input a public key* pk*, a message* m*, and a signature* σ*. This algorithm outputs* 1 *if* σ *is a valid signature for message* m *and* 0 *otherwise. This algorithm corresponds to* Verify_{Σ}.

Another primitive we investigated for the instantiation of CSRAS was threshold signature schemes because they allow for subsets of shareholders to correctly sign a message. This property is needed in our fine-grained authenticated share renewal scenario because the underlying secret sharing primitive allows for any subset of at least t shareholders to successfully carry out proactive secret sharing. More precisely, only t out of the n shareholders need to be "active"during share renewal by computing and distributing the sub-shares. The other (at most) $n-t$ "passive"shareholders only need to add the sub-shares received to the old share received by the data owner and store the result of this sum in place of the previous share. Since proactive secret sharing is expensive in terms of the amount of private channel needed to exchange the sub-shares, letting an authorized subset of shareholders carry it out decreases the complexity of this protocol. Accordingly, the signature scheme used to sign the sub-shares should provide this feature and, therefore, threshold signature schemes are a good candidate with this respect. However, this property holds only when a single message is signed by the shareholders. This constraint makes threshold signature schemes unsuitable for the CSRAS, because each shareholder signs multiple sub-shares.

5.5 Publicly Verifiable Secret Sharing

In the following, we discuss *publicly verifiable secret sharing* [16] and show why this primitive is not suitable for our authenticated share renewal scenario. First, we discuss solutions for publicly verifiable secret sharing and argue why they are not suitable. Second, we propose a modified approach based on publicly verifiable secret sharing and again argue why it does not meet our requirements.

Like *verifiable secret sharing* [6], publicly verifiable secret sharing is a cryptographic primitive to check the integrity of the shares. In addition, publicly verifiable secret sharing allows the integrity of the shares to be checked by any third party and not only by the designated shareholder. This is achieved through encryption of the shares, so that they are not available in clear to the third party. The encryption of the shares can be done through public-key encryption [16], [14], where the data owner encrypts the shares with the public key of the designated shareholder. Otherwise, encryption of the shares can be done through one-time pad [7]. The latter approach, together with integrity, ensures the privacy property required also in our fine-grained authenticity verifiability scenario. However, this approach does not provide authenticity. The approaches using public-key encryption do not provide unconditional privacy. The reason is that the encrypted shares are published so that *any* third party can verify them. The problem with

that is that a third party with enough computational power can decrypt as many shares as to reconstruct the secret, breaking confidentiality.

A possible way to modify the publicly verifiable approaches discussed in [16] and [14] to meet our privacy requirements follows. The data owner encrypts the shares by using the public-key of the respective shareholders and distributes the encrypted shares and the shares in clear to the shareholders through private channels. The encrypted shares are not published. Instead of allowing any third party to check the integrity of the shares, the verification procedure is outsourced to multiple third party auditors. These third party auditors receive at most $t-1$ shares to verify, given that the reconstructing threshold is t. During the verification phase, shareholders outsource through private channels their encrypted shares to the designated third party verifiers. Furthermore, it is assumed each shareholder outsources its share to one third party auditor only and that each of the third party auditor does not verify more than $t-1$ shares. This way the requirements for integrity and privacy of our authenticated share renewal scenario are fulfilled. However, publicly verifiable secret sharing does not provide authenticity as no means to link the shares to the data owner that generated them is provided. Thus, publicly verifiable secret sharing is not suitable for instantiating the CSRAS.

6 Conclusion

In this technical report, we investigated the problem of how to perform share renewal over data that are authenticated. We first provided a coarse-grained solution for the scenario where it is sufficient that authenticity of the outsourced document is checked after its reconstruction. Afterwards, we discussed how to provide means to check authenticity in a fine-grained manner throughout the storage of the document and defined so called CSRAS, a fine-grained solution framework for such scenario. We provided security definitions and discussed the features needed for the concrete instantiation of the CSRAS, which are currently not fulfilled by any signature scheme available in the literature. We discussed our attempts at providing such a solution and pointed out their shortcomings.

Acknowledgments. This work was in part funded by the European Commission through grant agreement no 644962 (PRISMACLOUD). Furthermore, it received funding from the DFG as part of project S6 within the CRC 1119 CROSSING.

References

1. Beimel, A.: Secret-Sharing Schemes: A Survey. In: IWCC 2011. LNCS, vol. 6639, pp. 11–46. Springer (2011)
2. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: International Workshop on Public Key Cryptography. pp. 31–46. Springer (2003)

3. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2656, pp. 416–432. Springer (2003), https://doi.org/10.1007/3-540-39200-9_26
4. Braun, J., Buchmann, J.A., Demirel, D., Geihs, M., Fujiwara, M., Moriai, S., Sasaki, M., Waseda, A.: LINCOS: A storage system providing long-term integrity, authenticity, and confidentiality. In: AsiaCCS. pp. 461–468. ACM (2017)
5. Brickell, E.F.: Some Ideal Secret Sharing Schemes. In: EUROCRYPT '89. LNCS, vol. 434, pp. 468–475. Springer (1989)
6. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: FOCS 1985
7. Demirel, D., Krenn, S., Lorünser, T., Traverso, G.: Efficient and privacy preserving third party auditing for a distributed storage system. In: 11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016. pp. 88–97. IEEE Computer Society (2016), https://doi.org/10.1109/ARES.2016.88
8. Farràs, O., Padró, C.: Ideal hierarchical secret sharing schemes. In: Micciancio, D. (ed.) Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings. Lecture Notes in Computer Science, vol. 5978, pp. 219–236. Springer (2010), https://doi.org/10.1007/978-3-642-11799-2_14
9. Fiore, D., Mitrokotsa, A., Nizzardo, L., Pagnin, E.: Multi-key homomorphic authenticators. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10032, pp. 499–530 (2016), https://doi.org/10.1007/978-3-662-53890-6_17
10. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. In: Fischlin, M., Buchmann, J.A., Manulis, M. (eds.) Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7293, pp. 697–714. Springer (2012), https://doi.org/10.1007/978-3-642-30057-8_41
11. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In: CRYPTO '95. LNCS, vol. 963, pp. 339–352. Springer (1995)
12. Johnson, R., Molnar, D., Song, D.X., Wagner, D.A.: Homomorphic signature schemes. In: Preneel, B. (ed.) Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2271, pp. 244–262. Springer (2002), https://doi.org/10.1007/3-540-45760-7_17
13. Nojoumian, M., Stinson, D.R.: Social secret sharing in cloud computing using a new trust function. In: Cuppens-Boulahia, N., Fong, P., García-Alfaro, J., Marsh, S., Steghöfer, J. (eds.) Tenth Annual International Conference on Privacy, Security and Trust, PST 2012, Paris, France, July 16-18, 2012. pp. 161–167. IEEE Computer Society (2012), https://doi.org/10.1109/PST.2012.6297936
14. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99,

19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 148–164. Springer (1999), https://doi.org/10.1007/3-540-48405-1_10
15. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)
16. Stadler, M.: Publicly verifiable secret sharing. In: Maurer, U.M. (ed.) Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding. Lecture Notes in Computer Science, vol. 1070, pp. 190–199. Springer (1996), https://doi.org/10.1007/3-540-68339-9_17
17. Tassa, T.: Hierarchical Threshold Secret Sharing. J. Cryptology 20(2), 237–264 (2007)
18. Traverso, G., Demirel, D., Buchmann, J.A.: Dynamic and verifiable hierarchical secret sharing. In: Nascimento, A.C.A., Barreto, P.S.L.M. (eds.) Information Theoretic Security - 9th International Conference, ICITS 2016, Tacoma, WA, USA, August 9-12, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10015, pp. 24–43 (2016), https://doi.org/10.1007/978-3-319-49175-2_2
19. Traverso, G., Demirel, D., Buchmann, J.A.: Homomorphic Signature Schemes - A Survey. Springer Briefs in Computer Science, Springer (2016), https://doi.org/10.1007/978-3-319-32115-8
20. Traverso, G., Demirel, D., Habib, S.M., Buchmann, J.A.: As^3: Adaptive social secret sharing for distributed storage systems. In: 14th Annual Conference on Privacy, Security and Trust, PST 2016, Auckland, New Zealand, December 12-14, 2016. pp. 528–535. IEEE (2016), https://doi.org/10.1109/PST.2016.7907011