Verifiable Secret Redistribution for Archive Systems

Theodore M. Wong   Chenxi Wang   Jeannette M. Wing
Carnegie Mellon University
tmwong+@cs.cmu.edu, chenxi@ece.cmu.edu, wing+@cs.cmu.edu

Abstract

We present a new verifiable secret redistribution *protocol for threshold sharing schemes that forms a key component of a proposed archival storage system. Our protocol supports redistribution from* (m,n) *to* (m',n') *threshold sharing schemes without requiring reconstruction of the original data. The design is motivated by archive systems for which the added security of threshold sharing of data must be accompanied by the flexibility of dynamic shareholder changes. Our protocol enables the dynamic addition or removal of shareholders, and also guards against mobile adversaries. We observe that existing protocols either cannot be extended readily to allow redistribution between different access structures, or have vulnerabilities that allow faulty old shareholders to distribute invalid shares to new shareholders. Our primary contribution is that in our protocol, new shareholders can verify the validity of their shares after redistribution between different access structures.*

1 Introduction

We are conducting research on the design and implementation of an archive system. The primary goal of an archive system is to preserve the long-term availability and confidentiality of data in the face of storage server failures and compromises. Another goal is to adapt to the addition or removal of servers. In this paper, we outline a design for an archive system that meets those goals, and present a protocol for secret redistribution that is a key component of the system.

We envision an archive system to store data that is infrequently accessed, but which must remain available and confidential for long periods of time. Examples of such data include medical records, corporate tax records, and classified government documents. For such data, we can trade off longer storage and retrieval latencies in return for stronger availability and confidentiality guarantees.

The archival nature of our system allows us to use relatively heavyweight schemes for distributing the data (hereafter referred to as the *secret*) to storage servers. In our system, we use a *threshold secret sharing scheme* [39] with an (m,n) *access structure* to create n shares of the secret for n servers (also called *shareholders*). We only require m (where $m \leq n$) shares to reconstruct the secret, and an adversary must compromise at least m shareholders to compromise the secret. Threshold schemes introduce a degree of fault-tolerance: we can reconstruct the secret even if $n - m$ shareholders fail. Of course, we assume that there is enough diversity among the servers such that common security flaws and failure modes can be ruled out.

We design our archive system to defend against *active* and *mobile* adversaries. An active (or Byzantine [31]) adversary corrupts data or state, and may alter or replay messages. To defend against active adversaries, secret sharing must be *verifiable* [14, 33]: the participants of the sharing protocol should be able to verify the correctness of protocol execution, since an active attacker can send ill-formed protocol messages. A mobile (or dynamic) adversary compromises servers progressively, and left unchecked will eventually compromise enough shareholders to compromise the secret. To counteract mobile adversaries, there exist *proactive secret sharing* (PSS) schemes [18, 25] to regenerate shares periodically at all shareholders. PSS schemes assume a system model of temporary compromise (i.e., compromised shareholders can be restored to a clean state by a reboot), and that the adversary compromises at most $m - 1$ shareholders simultaneously prior to regeneration.

Our archive system imposes one additional requirement: we must preserve a minimum level of fault-tolerance over the long term. When shareholders become unavailable (due to benign failures or denial-of-service attacks), we must produce new shares and incorporate new shareholders to store those shares. Similarly, when new shareholders join the system, we may want to include them in the sharing scheme to balance loads and maintain availability. Finally, we must allow the possibility of permanent compromise (i.e., the removal of shareholders), in contrast to the PSS model where only temporary compromise is considered.

Our design goals and the desire to defend against active and mobile adversaries require a general sharing scheme in which the secret can be redistributed dynamically to a new set of shareholders after the initial sharing. The new shareholders may form a new access structure, i.e., they may or may not overlap with the old shareholders, and they may or may not share the same threshold value.

A trivial, but insecure, solution for recovering from lost or compromised shareholders is simply to reconstruct the original secret at a central "recovery" server and distribute new shares. This solution suffers from an obvious weakness: an adversary that compromises the recovery server immediately gains the ability to see all secrets. We require a solution that does not involve the reconstruction of the secret, in which the work of redistribution is performed by the remaining (non-compromised) shareholders.

There exist secret redistribution protocols that do not require reconstruction of the original secret. In particular, Desmedt and Jajodia [13] and Frankel *et al* [15] proposed protocols for the redistribution of secrets between different access structures. However, both of these extensions have vulnerabilities that allow faulty shareholders to corrupt the redistribution process undetectably. After a corrupted run, shareholders will have shares that cannot be used to reconstruct the original secret. We discuss the vulnerabilities in depth in Section 4.1.

We present a new *verifiable secret redistribution* (VSR) protocol for the redistribution of secrets from an (m,n) access structure to an arbitrary (m',n') access structure. Our protocol is similar to that of Desmedt and Jajodia; however, we incorporate a verification capability to enable shareholders to verify the *validity* of the new shares (i.e., that their shares can be used to reconstruct the original secret). We stress that the ability to perform verification is essential for archive systems where having some compromised entities is the common case rather than the exception. We present two verification conditions and prove that they are sufficient to guarantee the correctness of the secret redistribution. We also prove a *security* property of our protocol: an adversary that obtains fewer that m old shares and fewer than m' new shares cannot compromise the secret.

The key points of our paper are:

- The long-term availability and confidentiality of secrets must be preserved in the face of server failures and compromises. Therefore, distributed archive systems must include capabilities to accommodate dynamic changes in the group of servers that implement the system.

- Existing redistribution protocols [13, 15] and PSS schemes either allow a faulty shareholder to corrupt redistribution undetectably (and leave shareholders with invalid shares) or prohibit changes to the set of shareholders that store shares.

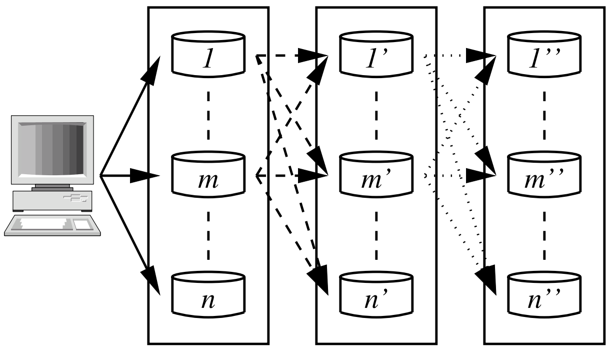

Figure 1. High-level operation of our archive system. A client distributes a file to the active group of servers (solid lines). When the servers detect changes in the active group membership, they redistribute the file to the new group (dashed lines). Servers may perform redistribution an arbitrary number of times (dotted lines) prior to reconstruction.

- Pinpoint identification and removal of faulty old shareholders is not immediately possible if redistribution is to occur between two disjoint sets of shareholders. In the worst case, $\binom{n}{m} - \binom{n-m+1}{m}$ restarts (for an (m,n) access structure) are required to eliminate faulty shareholders and complete the protocol.

2 Archive system architecture

We present a high-level view of our archive system architecture in Figure 1. It consists of two components: clients and a group of storage servers. Clients perform the initial distribution and final reconstruction of files (i.e., secrets), and are considered trusted entities.

The servers store shares and perform redistribution. We discuss the issue of faulty servers (i.e., shareholders) in depth in Section 4.1. Though the number of servers that implement the archive system may be large, we assume that the number of servers n that store shares for a particular file (or set of files) is small.

We require that the network provides private point-to-point links between the client and servers, and between all pairs of servers. We also require that the network supports reliable broadcast; if a server broadcasts a message M, then M is received at all other servers. In practice, networks do not usually support reliable broadcast, and we must use protocols built over point-to-point links to emulate broadcast. Since the number of servers n that store shares for a file is small, we expect that the overhead required by the (otherwise expensive) broadcast protocols will be small relative to the computational cost of the VSR protocol. In our prototype system, we use the broadcast primitives implemented in the Ensemble group communication toolkit [24].

The servers require mechanisms to keep track of the members of the active group of servers, and to determine when servers have joined or left (intentionally or through failure) the group. Upon learning of a change in group membership, the servers use the VSR protocol to redistribute shares of files to a new access structure based on the size of the group. We assume that the rate of change of membership is low compared to the rate at which clients contact the group for I/O operations. Since group membership protocols are distributed by definition, they come with additional assumptions about the failure behavior of participating servers (i.e., whether server failures are benign or Byzantine). As the focus in our prototype is primarily on the properties of the VSR protocols itself, we elect to treat the group membership protocol as a "black-box" service provided by the underlying network, and we use the membership protocol and gossip-style failure detector [40] implemented in Ensemble. The protocol in Ensemble tolerates only benign failures; other protocols exist that tolerate Byzantine failures [28, 37].

Clients require a mechanism to locate the active group of servers for I/O operations. Clients are not part of the group of servers (in contrast to peer-to-peer storage systems such as Intermemory [10] or OceanStore [30]), and thus cannot rely on the group membership protocol used by the servers. Also, most (relatively expensive) protocols are designed with the assumption that membership changes are infrequent; given our assumption that the rate at which membership changes is lower than the rate at which clients contact the group for I/O operations, having the client join the group temporarily for I/O would have a negative impact on performance. A simple approach would be for the client to contact a central directory that replies with the list of servers; the servers would update the directory after a change in group membership. Of course, the central directory is an obvious point of vulnerability in an otherwise decentralized architecture. A more robust approach is for the client to contact a replicated directory service that uses agreement protocols to ensure consistent and valid updates to the list of servers (such as in Farsite [1]). In our prototype, we adopt a third approach: since our test network is small, a client broadcasts a query over the network to locate the active set; a distinguished member of the group (the coordinator of the group membership protocol, for convenience), responds with the list of servers. Note that even if a faulty server responds with an invalid list, a non-faulty server will respond with a valid list, which alerts the client to the existence of a faulty server.

Clients also require a heuristic to select the threshold value m given n servers. We require m non-faulty servers, and we can tolerate at most $m-1$ faulty servers; we disuss faulty servers further in Section 4.1. Thus, we have the constraint that $m+m-1 \leq n$, or

$$m \leq \left\lfloor \frac{n+1}{2} \right\rfloor \tag{1}$$

In our prototype, we set $m = \left\lfloor \frac{n+1}{2} \right\rfloor$.

To store a file in the archive, a client locates the active group of n servers and selects the (m,n) access structure to use. It then distributes n shares of the file and a *witness* to the file (described in Section 4) to the servers.

When the servers detect that another server has joined or left the group, they use the VSR protocol to redistribute their shares of the file to the new group. The servers use the same heuristic as the clients to select the new threshold value m' given the size n' of the new group, i.e., $m' = \left\lfloor \frac{n'+1}{2} \right\rfloor$. Servers may redistribute the file an arbitrary number of times.

Finally, when a client needs to reconstruct the file, it locates the active group of servers, which may differ from the group to which it distributed shares initially. The client then retrieves at least m' shares and reconstructs the file.

3 Cryptographic building blocks

In this section, we outline the cryptographic protocols that form the building blocks for our VSR protocol. We first recap Shamir's threshold sharing scheme [39], and then summarize Desmedt and Jajodia's secret redistribution protocol [13] and Feldman's VSS scheme [14].

3.1 Shamir's threshold sharing scheme

Shamir's (m,n) threshold sharing scheme is based on polynomial interpolation [39]. Secrets k are in $\mathbb{Z}_p$, where p is prime and $p > n$, and shareholders i are in $\mathcal{P}$, where $|\mathcal{P}| = n$. Shares s_i of i are also in $\mathbb{Z}_p$. Authorized subsets $\mathcal{B}$ are in the access structure $\Gamma_{\mathcal{P}}^{(m,n)}$, where $|\mathcal{B}| = m$.

To distribute k to the access structure $\Gamma_{\mathcal{P}}^{(m,n)}$, we select an $m-1$ degree polynomial $a(x)$ with constant term k and random coefficients $a_1 \ldots a_{m-1} \in \mathbb{Z}_p$, and use $a(x)$ to generate s_i for each i:

$$s_i = k + a_1 i + \ldots + a_{m-1} i^{m-1} \tag{2}$$

To reconstruct k, we retrieve m coordinate pairs (i, s_i) of $i \in \mathcal{B}$, and use Lagrange interpolation:

$$k = \sum_{i \in \mathcal{B}} b_i s_i \quad \text{where} \quad b_i = \prod_{j \in \mathcal{B}, j \neq i} \frac{j}{(j-i)} \tag{3}$$

Desmedt and Jajodia's Secret Redistribution protocol for Shamir's scheme:
To redistribute a secret $k \in \mathbb{Z}_p$ from a $\Gamma_{\mathcal{P}}^{(m,n)}$ to a $\Gamma_{\mathcal{P}'}^{(m',n')}$ access structure, using the authorized subset $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(m,n)}$:

1. For each $i \in \mathcal{B}$, use the polynomial $a'_i(j) = s_i + a'_{i1}j + \ldots + a'_{i(m'-1)}j^{m'-1}$ to compute the subshares $\hat{s}_{ij}$ of s_i, and send $\hat{s}_{ij}$ to the corresponding $j \in \mathcal{P}'$.
2. For each $j \in \mathcal{P}'$, generate a new share s'_j by Lagrange interpolation:

$$s'_j = \sum_{i \in \mathcal{B}} b_i \hat{s}_{ij} \quad \text{where} \quad b_i = \prod_{x \in \mathcal{B}, x \neq i} \frac{x}{(x-i)}$$

b_i are constant for each $i \in \mathcal{B}$, are independent of the choice of $a'_i(x)$, and may be precomputed.

Figure 2. Protocol for the redistribution of shares of a secret from a $\Gamma_{\mathcal{P}}^{(m,n)}$ to a $\Gamma_{\mathcal{P}'}^{(m',n')}$ access structure [13], for Shamir's threshold sharing scheme [39].

3.2 Desmedt and Jajodia's secret redistribution protocol

Desmedt and Jajodia present a protocol for the redistribution of shares of secrets from threshold sharing schemes without requiring the intermediate reconstruction of the secret [13]. We specialize their protocol for use with Shamir's threshold sharing scheme [39], as shown in Figure 2. Suppose we have distributed a secret k to the access structure $\Gamma_{\mathcal{P}}^{(m,n)}$, and wish to redistribute k to the access structure $\Gamma_{\mathcal{P}'}^{(m',n')}$. To achieve this, we select an authorized subset $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(m,n)}$. Each shareholder $i \in \mathcal{B}$ uses Shamir's scheme to distribute *subshares* $\hat{s}_{ij}$ of its share s_i to $\Gamma_{\mathcal{P}'}^{(m',n')}$. Each shareholder $j \in \mathcal{P}'$ receives $\hat{s}_{ij}$ from each i, and generates a new share s'_j by Lagrange interpolation:

$$s'_j = \sum_{i \in \mathcal{B}} b_i \hat{s}_{ij} \quad \text{where} \quad b_i = \prod_{x \in \mathcal{B}, x \neq i} \frac{x}{(x-i)} \tag{4}$$

3.3 Feldman's VSS scheme

Feldman presents a VSS scheme that can be used by shareholders of a secret to verify the validity of their shares [14]. We specialize the VSS scheme for use with Shamir's threshold sharing scheme [39], as shown in Figure 3. Herzberg *et al* present a similar treatment [26].

The application of Feldman's VSS scheme to Shamir's scheme takes advantage of the homomorphic properties of exponentiation and the assumption that the computation of discrete logs in a finite field is intractable. Suppose we have fields $\mathbb{Z}_p$ and $\mathbb{Z}_r$, such that p and r are prime and $r = pq+1$ (where q is a non-negative integer), and suppose we have an element $g \in \mathbb{Z}_r$ of order p. Then, suppose we use Shamir's scheme with polynomial $a(x)$ to distribute a secret $k \in \mathbb{Z}_p$ to the access structure $\Gamma_{\mathcal{P}}^{(m,n)}$. In addition to sending the shares $s_i \in \mathbb{Z}_p$ to shareholders $i \in \mathcal{P}$, we broadcast witnesses to k and the coefficients a_1 ... a_{m-1} of $a(x)$ of the form g^k and g^a_1 ... $g^{a_{m-1}}$. Each i may then verify that s_i is a valid share of k:

$$g^{s_i} \equiv g^k (g^{a_1})^i \ldots (g^{a_{m-1}})^{i^{m-1}} \tag{5}$$

which is the exponentiation of $a(x)$ (Equation (2)). Since we have assumed that the computation of discrete logs is intractable, no-one can learn k or a_1 ... a_{m-1} from the broadcast of the witnesses.

4 The VSR protocol

We present our verifiable secret redistribution protocol for secrets distributed with Shamir's threshold sharing scheme [39]. The protocol takes as input shares of a secret distributed to the access structure $\Gamma_{\mathcal{P}}^{(m,n)}$, and outputs shares of the secret distributed to the access structure $\Gamma_{\mathcal{P}'}^{(m',n')}$. We assume that the computation of discrete logs in a finite field is intractable, and that there exist reliable broadcast channels among all participants and private channels between every pair of participants. We also assume that there are at least m non-faulty old shareholders, that there are at most $m-1$ faulty old shareholders, and that there are n' non-faulty new shareholders.

The initial distribution of a secret (INITIAL in Figure 4) proceeds as in Feldman's VSS scheme [14]. The dealer of secret $k \in \mathbb{Z}_p$ distributes shares $s_i \in \mathbb{Z}_p$ to each shareholder $i \in \mathcal{P}$, using the polynomial $a(i)$ (INITIAL step 1). The dealer also broadcasts g^k and g^{a_1} ... $g^{a_{m-1}}$, which each i uses in Equation (5) to verify the validity of s_i (INITIAL steps 2 and 3). If Equation (5) holds, i stores s_i and g^k (INITIAL step 4).

Redistribution of the secret (REDIST in Figure 4) proceeds as follows. Each i in an authorized subset $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(m,n)}$ uses Shamir's scheme (with the polynomial $a'_i(j)$) to distribute subshares $\hat{s}_{ij} \in \mathbb{Z}_p$ of its share s_i to $\Gamma_{\mathcal{P}'}^{(m',n')}$ (REDIST step 1). Each shareholder $j \in \mathcal{P}'$ receives $\hat{s}_{ij}$ from each i, and generates a new share s'_j (Equation (4), which is

Feldman's Verifiable Secret Sharing scheme for Shamir's scheme:
To distribute a secret $k \in \mathbb{Z}_p$ to the access structure $\Gamma_{\mathcal{P}}^{(m,n)}$:

1. Use the polynomial $a(i) = k + a_1 i + \ldots + a_{m-1} i^{m-1}$ to compute the shares s_i of k, and send s_i to the corresponding $i \in \mathcal{P}$ over private channels.
2. Use g to compute $g^k, g^{a_1} \ldots g^{a_{m-1}}$, and send them to all $i \in \mathcal{P}$ over the broadcast channel.
3. For each $i \in \mathcal{P}$, verify that:

$$g^{s_i} \equiv g^k \prod_{l=1}^{m-1} (g^{a_l})^{i^l}$$

If the condition holds, i broadcasts a "commit" message. Otherwise, i broadcasts an "abort" message.

Figure 3. Feldman's VSS scheme [14] for Shamir's threshold sharing scheme [39].

REDIST step 4). We may redistribute k an arbitrary number of times before we reconstruct it. This redistribution phase is the same as in Desmedt and Jajodia's protocol [13].

For the new shareholders to verify that their shares of the secret are valid after redistribution, we require that two conditions, SHARES-VALID and SUBSHARES-VALID, hold. When all $i \in \mathcal{B}$ redistribute s_i to each $j \in \mathcal{P}'$, all s_j are valid shares of k if:

SHARES-VALID:
$k = \sum_{i \in \mathcal{B}} b_i s_i$

SUBSHARES-VALID:
$\forall i \in \mathcal{B}; \mathcal{B}' \in \Gamma_{\mathcal{P}'}^{(m',n')} : s_i = \sum_{j \in \mathcal{B}'} b'_j \hat{s}_{ij}$

We define a NEW-SHARES-VALID condition, which will hold if new shareholders have valid shares of the secret. We prove in Section 4.3 that NEW-SHARES-VALID holds if SHARES-VALID and SUBSHARES-VALID hold. The definition of NEW-SHARES-VALID follows from Equation (3) for a secret distributed to $\Gamma_{\mathcal{P}'}^{(m',n')}$:

NEW-SHARES-VALID:
$\forall \mathcal{B}' \in \Gamma_{\mathcal{P}'}^{(m',n')} : k = \sum_{j \in \mathcal{B}'} b'_j s'_j$

We use Feldman's VSS scheme to verify that SUBSHARES-VALID holds. The distribution of $\hat{s}_{ij}$ from s_i (REDIST step 1) is just an application of Shamir's scheme. Thus, each $i \in \mathcal{B}$ broadcasts witnesses to its share and the coefficients of $a'_i(j)$ (g^{s_i} and $g^{a_{i1}}$ ... $g^{a_{i(m-1)}}$), which each j uses to verify the validity of $\hat{s}_{ij}$ (REDIST step 2).

The key insight embodied in our VSR protocol is that the naïve extension of Desmedt and Jajodia's protocol with Feldman's scheme does not in itself allow the new shareholders to verify that NEW-SHARES-VALID holds. The difficulty arises because Feldman's scheme only verifies that SUBSHARES-VALID holds, which in the absence of SHARES-VALID is insufficient to verify that NEW-SHARES-VALID holds. Although Desmedt and Jajodia observe that the linear properties of their protocol and the properties of g^x ensure that each j generates valid shares [13], they implicitly assume that each $i \in \mathcal{B}$ distributes subshares of valid s_i. The VSS scheme only allows $i \in \mathcal{B}$ shareholder to prove that it distributed valid $\hat{s}_{ij}$ of some value. However, i may have distributed "subshares" of some random value instead of subshares of s_i. Thus, we require a sub-protocol for i to prove that it distributed $\hat{s}_{ij}$ of s_i.

The same flaw can be found in the proactive RSA scheme proposed by Frankel *et al* [15]. Their protocol uses a poly-to-sum redistribution from a polynomial sharing scheme to an additive sharing scheme, and a sum-to-poly redistribution from the additive scheme back to a polynomial scheme. They suggest that changes in the threshold and number of shareholders can be accommodated in the poly-to-sum redistribution. Unfortunately, their verification checks hold only if one retains the same set of shareholders. If distribution to new shareholders is required, their verification conditions ensures SUBSHARE-VALID, but SHARES-VALID condition may not hold because their "proper secret" check relies on a witness value ($g^{s_i} L^2$ in their paper) computed from information distributed in the preceding round. A faulty shareholder can thus distribute spurious information to new shareholders and ultimately cause them to accept an invalid witness value.

To allow the new shareholders to verify that SHARES-VALID holds, which together with SUBSHARES-VALID verifies that NEW-SHARES-VALID holds, the old shareholders in our protocol broadcast a witness to the secret. Each $i \in \mathcal{B}$ must therefore store g^k (received during INITIAL) and later broadcast it to all $j \in \mathcal{P}'$. Recall that each j receives s_i from each i to verify that SUBSHARES-VALID holds. Once each j receives g^k, it verifies that s_i is a valid share of k:

$$g^k = \prod_{i \in \mathcal{B}} g^{b_i s_i} \tag{6}$$

Equation (6) follows from Equation (3) and the homomorphic properties of exponentiation. Since we have assumed that the computation of discrete logs is intractable, no-one

Verifiable Secret Redistribution protocol for Shamir's sharing scheme:

INITIAL: To distribute a secret $k \in \mathbb{Z}_p$ to the access structure $\Gamma_{\mathcal{P}}^{(m,n)}$:

1. Use the polynomial $a(i) = k + a_1 i + \ldots + a_{m-1} i^{m-1}$ to compute the shares s_i of k, and send s_i to the corresponding $i \in \mathcal{P}$ over private channels.
2. Use generator g to compute $g^k, g^{a_1} \ldots g^{a_{m-1}}$, and send them to all $i \in \mathcal{P}$ over the broadcast channel.
3. For each $i \in \mathcal{P}$, verify that:

$$g^{s_i} \equiv g^k \prod_{l=1}^{m-1} (g^{a_l})^{i^l}$$

   If the condition holds, i broadcasts a "commit" message. Otherwise, i broadcasts an "abort" message.
4. If all $i \in \mathcal{P}$ agree to commit, each i stores s_i and g^k. Otherwise, they abort the protocol.

REDIST: To redistribute $k \in \mathbb{Z}_p$ from $\Gamma_{\mathcal{P}}^{(m,n)}$ to the access structure $\Gamma_{\mathcal{P}'}^{(m',n')}$, using the authorized subset $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(m,n)}$:

1. For each $i \in \mathcal{B}$, use the polynomial $a'_i(j) = s_i + a'_{i1} j + \ldots + a'_{i(m'-1)} j^{m'-1}$ to compute the subshares $\hat{s}_{ij}$ of s_i, and send $\hat{s}_{ij}$ to the corresponding $j \in \mathcal{P}'$ over private channels.
2. For each $i \in \mathcal{B}$, use g to compute $g^{s_i}, g^{a'_{i1}} \ldots g^{a'_{i(m'-1)}}$, and send them to all $j \in \mathcal{P}'$ over the broadcast channel.
3. For each $j \in \mathcal{P}'$, verify that:

$$\forall i \in \mathcal{B} : g^{\hat{s}_{ij}} \equiv g^{s_i} \prod_{l=1}^{m'-1} (g^{a'_{il}})^{j^l}$$

   and:

$$g^k \equiv \prod_{i \in \mathcal{B}} (g^{s_i})^{b_i} \quad \text{where} \quad b_i = \prod_{l \in \mathcal{B}, l \neq i} \frac{l}{(l-i)}$$

   If the conditions hold, j broadcasts a "commit" message. Otherwise, j broadcasts an "abort" message.
4. If all $j \in \mathcal{P}'$ agree to commit, each j generates a new share s'_j:

$$s'_j = \sum_{i \in \mathcal{B}} b_i \hat{s}_{ij} \quad \text{where} \quad b_i = \prod_{l \in \mathcal{B}, l \neq i} \frac{l}{(l-i)}$$

   and stores s'_j and g^k. Otherwise, they abort the protocol.

Figure 4. Protocol for the verifiable redistribution of shares of a secret from a $\Gamma_{\mathcal{P}}^{(m,n)}$ to a $\Gamma_{\mathcal{P}'}^{(m',n')}$ access structure, for Shamir's threshold sharing scheme [39].

can learn k from the broadcast of g^k.

We have developed a generalized VSR protocol for linear threshold sharing schemes. The details of the generalized protocol appear in our technical report [42].

4.1 Discussion about faulty shareholders

During redistribution from a $\Gamma_{\mathcal{P}}^{(m,n)}$ to a $\Gamma_{\mathcal{P}'}^{(m',n')}$ access structure with our VSR protocol, we assume that at least m old shareholders in $\mathcal{P}$ are non-faulty (otherwise we will have an insufficient number of shares to reconstruct the secret), and that at most $m-1$ old shareholders in $\mathcal{P}$ may be faulty (since m faulty shareholders could collude to reconstruct the original secret). We also assume that all n' of the shareholders in $\mathcal{P}'$ are non-faulty. We denote faulty shareholders, and the values they distribute, with over-bars. A non-faulty shareholder $i \in \mathcal{P}$ distributes valid subshares $\hat{s}_{ij}$ of its share s_i to all shareholders $j \in \mathcal{P}'$ and broadcasts g^k corresponding to secret $k \in \mathbb{Z}_p$. A faulty shareholder $\bar{i} \in \mathcal{P}$ may distribute invalid subshares $\overline{\hat{s}_{\bar{i}j}}$ or broadcast $\overline{g^k}$ not corresponding to k.

In order to check that the verification conditions hold, we require that certain information be made available to the new shareholders. In our VSR protocol, this information is witnesses g^k, g^{s_i}, and $g^{a_{i1}}$ ... $g^{a_{i(m-1)}}$. In the PSS scheme of Frankel *et al.* [15], this information is the value $g^{s_i L^2}$ and g^d. In the absence of a trusted information repository, the new members must rely on the old shareholders to deliver this information. It is this process that proves to be problematic for the pinpoint identification of faulty shareholders.

Consider redistribution from $\Gamma_{\mathcal{P}}^{(m,n)}$ to $\Gamma_{\mathcal{P}'}^{(m',n')}$. Assume that we start with a random authorized subset $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(m,n)}$, and recall that $|\mathcal{B}| = m$. It is possible that some subset of the old shareholders in $\underline{\mathcal{B}}$ (at most $m-1$) are faulty, and will attempt to broadcast $\overline{g^k}$ and $\overline{\hat{s}_{\bar{i}j}}$. If the faulty shareholders conspire to broadcast the same $\overline{g^k}$, the new shareholders will detect the discrepancy in the m broadcast values, but cannot pinpoint the faulty shareholders. The new shareholders cannot use majority voting since the majority of old shareholders in $\mathcal{B}$ may be faulty.

Since at most $m-1$ shareholders may be faulty, any randomly selected authorized subset of m old shareholders must contain at least one non-faulty shareholder. If the new shareholders detect discrepancies in the witnesses broadcast by the old shareholders, they can restart the redistribution protocol with another authorized subset until all values are consistent and all verification conditions hold. For $\Gamma_{\mathcal{P}}^{(m,n)}$, the number of times we must restart the redistribution protocol is bounded in the worst case by

$$\binom{n}{m} - \binom{n-m+1}{m} = \sum_{i=1}^{m-1} \binom{m-1}{i} \binom{n-m+1}{m-i} \quad (7)$$

which is the number of sets of size m containing at least one faulty shareholder, given $m-1$ faulty shareholders.

The assumption that all n' shareholders in $\mathcal{P}'$ are non-faulty is reasonable if we view the purpose of our VSR protocol as one of detecting faulty behavior by shareholders in $\mathcal{P}$. This is analogous to one of the assumptions underlying Feldman's VSS scheme [14] in which the shareholders are implicitly trusted to store valid shares (and reject invalid shares) of a secret.

4.2 Computational cost

The computational cost for each new shareholder of verification in our VSR protocol (REDIST Step 3 in Figure 4) is $O(mm')$ multiplications and $O(mm')$ exponentiations, exclusive of the cost of computing the witnesses. Consider redistribution from a $\Gamma_{\mathcal{P}}^{(m,n)}$ to a $\Gamma_{\mathcal{P}'}^{(m',n')}$ access structure. Each new shareholder $j \in \mathcal{P}'$ performs $m-1$ multiplications ($\mathcal{B} \in \Gamma_{\mathcal{P}}^{(m,n)}$; $|\mathcal{B}| = m$) and m exponentiations to verify that SHARES-VALID holds (Equation (6)), for a total cost of $O(m)$; we do not include the (small) cost of computing the powers of i. Each j also performs $m'-1$ multiplications ($\mathcal{B}' \in \Gamma_{\mathcal{P}'}$; $|\mathcal{B}'| = m'$) and $m'-1$ exponentiations for m old shareholders $i \in \mathcal{B}$ to verify that SUBSHARES-VALID holds (Equation (5)), for a total cost of $O(mm')$. Thus, the total cost for each j to verify that both conditions hold is $O(mm')$ multiplications and $O(mm')$ exponentiations, exclusive of the cost of computing the witnesses. In the worst case, the number of times we must restart the redistribution protocol is bounded by Equation (7).

4.3 Proof of correctness

We prove that NEW-SHARES-VALID holds after redistribution if SHARES-VALID and SUBSHARES-VALID hold. We also show that Equations (5) and (6) verify that SUBSHARES-VALID and SHARES-VALID hold.

Lemma 1 SUBSHARES-VALID *holds if Equation (5) holds.*

PROOF: Proved by Feldman [14]. □

Lemma 2 SHARES-VALID *holds if Equation (6) holds.*

PROOF: Assume that Equation (6) holds. It then follows that SHARES-VALID holds from Equation (3) and the homomorphic properties of exponentiation. □

Theorem 1 (VSR correctness) *For the verifiable redistribution of shares of a secret from a $\Gamma_{\mathcal{P}}^{(m,n)}$ to a $\Gamma_{\mathcal{P}'}^{(m',n')}$ access structure for Shamir's threshold sharing scheme [39], for all secrets $k \in \mathbb{Z}_p$, and for all authorized subsets $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(m,n)}$, $\mathcal{B}' \in \Gamma_{\mathcal{P}'}^{(m',n')}$,* NEW-SHARES-VALID *holds after redistribution of k with the VSR protocol if* SHARES-VALID *and* SUBSHARES-VALID *hold.*

PROOF: Assume that SHARES-VALID and SUBSHARES-VALID hold. Then:

$$\begin{aligned}
k &= \sum_{i\in\mathcal{B}} b_i s_i && \text{(SHARES-VALID)}\\
&= \sum_{i\in\mathcal{B}}\left(b_i \sum_{j\in\mathcal{B}'} b'_j \hat{s}_{ij}\right) && \text{(SUBSHARES-VALID)}\\
&= \sum_{i\in\mathcal{B}}\sum_{j\in\mathcal{B}'} b_i b'_j \hat{s}_{ij} && (x(y+z)=xy+xz)\\
&= \sum_{i\in\mathcal{B}}\sum_{j\in\mathcal{B}'} b'_j b_i \hat{s}_{ij} && (xy=yx)\\
&= \sum_{j\in\mathcal{B}'}\sum_{i\in\mathcal{B}} b'_j b_i \hat{s}_{ij} && (x+y=y+x)\\
&= \sum_{j\in\mathcal{B}'}\left(b'_j \sum_{i\in\mathcal{B}} b_i \hat{s}_{ij}\right) && (xy+xz=x(y+z))\\
&= \sum_{j\in\mathcal{B}'} b'_j s'_j && \text{(Equation (4))}
\end{aligned}$$

□

Our correctness proof mirrors that for Desmedt and Jajodia's secret redistribution protocol [13].

4.4 Proof of security

We prove that an adversary cannot reconstruct a secret from a combination of shares distributed with Shamir's threshold sharing scheme [39] to a $\Gamma_{\mathcal{P}}^{(m,n)}$ access structure and shares distributed to a $\Gamma_{\mathcal{P}'}^{(m',n')}$ access structure. In particular, we show that an adversary that has obtained $m-1$ old shares of a secret k and $m'-1$ new shares of the same k cannot reconstruct k. It is then trivial to show that an adversary that has less than $m-1$ old shares and $m'-1$ new shares of the same k cannot reconstruct k.

To complete our security proof, we require some lemmas (presented by Beaumont [2] and Kostrikin [29]) for systems of u linear equations in v unknowns of the form

$$\begin{array}{c}
m_{11}x_1 + m_{12}x_2 + \cdots + m_{1v}x_v = b_1\\
m_{21}x_1 + m_{22}x_2 + \cdots + m_{2v}x_v = b_2\\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
m_{u1}x_1 + m_{u2}x_2 + \cdots + m_{uv}x_v = b_2
\end{array} \tag{8}$$

Let $\mathbf{M}$, $\mathbf{x}$, and $\mathbf{b}$ denote

$$\mathbf{M} = \begin{bmatrix} m_{11} & \cdots & m_{1v}\\ \vdots & \ddots & \vdots\\ m_{u1} & \cdots & m_{uv}\end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1\\ \vdots\\ x_v\end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1\\ \vdots\\ b_u\end{bmatrix}$$

let $[\mathbf{M}|\mathbf{b}]$ denote the *augmented matrix*

$$[\mathbf{M}|\mathbf{b}] = \begin{bmatrix} m_{11} & \cdots & m_{1v} & b_1\\ \vdots & \ddots & \vdots & \vdots\\ m_{u1} & \cdots & m_{uv} & b_u\end{bmatrix}$$

let $\text{rank}(\mathbf{M})$ denote the rank of $\mathbf{M}$ (number of linearly independent columns in $\mathbf{M}$), and let $\det(\mathbf{M})$ denote the determinant of $\mathbf{M}$.

Lemma 3 $\text{rank}(\mathbf{M}) = \text{rank}(\mathbf{M}^T)$.

Lemma 4 (Kronecker-Capelli theorem) *If (and only if)* $\text{rank}(\mathbf{M}) = \text{rank}([\mathbf{M}|\mathbf{b}])$*, then Equation (8) has a solution for* $\mathbf{x}$*. Furthermore, if* $\text{rank}(\mathbf{M}) < v$*, then Equation (8) has infinitely many solutions for* $\mathbf{x}$.

Lemma 5 (Cramer's rule) *If* $u = v$ *and* $\det(\mathbf{M}) \neq 0$*, then Equation (8) has a unique solution for* $\mathbf{x}$.

Lemma 6 *For* $u \times u$ *matrix* $\mathbf{A}$*,* $v \times v$ *matrix* $\mathbf{B}$*, and* $u \times v$ *matrix* $\mathbf{C}$*,*

$$\det\left(\begin{bmatrix}\mathbf{A} & \mathbf{C}\\ \mathbf{0} & \mathbf{B}\end{bmatrix}\right) = \det(\mathbf{A})\det(\mathbf{B})$$

PROOF: Presented by Kostrikin [29]. □

Theorem 2 (VSR security) *For the verifiable redistribution of shares of a secret from a* $\Gamma_{\mathcal{P}}^{(m,n)}$ *to a* $\Gamma_{\mathcal{P}'}^{(m',n')}$ *access structure for Shamir's threshold sharing scheme [39], and for all secrets* $k \in \mathbb{Z}_p$*, the shares* s_i *of shareholders* i *in any non-authorized subset* $\overline{\mathcal{B}} \notin \Gamma_{\mathcal{P}}^{(m,n)}$ *cannot be used with the shares* s'_j *of shareholders* j *in any non-authorized subset* $\overline{\mathcal{B}}' \notin \Gamma_{\mathcal{P}'}^{(m',n')}$ *to uniquely determine* k.

PROOF: Assume there is a unique solution for k from the shares of shareholders in $\overline{\mathcal{B}}$ and $\overline{\mathcal{B}}'$. We show that this assumption leads to a contradiction.

Consider the case where $|\overline{\mathcal{B}}| = m-1$ and $|\overline{\mathcal{B}}'| = m'-1$, and suppose that we have s_i of $i \in \overline{\mathcal{B}}$ and s'_j of $j \in \overline{\mathcal{B}}'$. We use Equation (2) to construct Equation (9) in Figure 5.

Let $\mathbf{M}$ denote the left-hand matrix in Equation (9), $\mathbf{a}$ the coefficient vector k, a_1 ... $a'_{m'-1}$, and $\mathbf{s}$ the share vector. The maximum possible value for $\text{rank}(\mathbf{M})$ is the number of rows ($m+m'-2$, by Lemma 3), which is less than the number of values in $\mathbf{a}$ ($m+m'-1$). Also, $\text{rank}(\mathbf{M}) = \text{rank}([\mathbf{M}|\mathbf{s}])$ since $\mathbf{s}$ is a linear combination of the columns of $\mathbf{M}$ (by the method of share generation). Thus, we have infinitely many solutions for $\mathbf{a}$ in Equation (9) (by Lemma 4). We arrive at the same conclusion for any $\overline{\mathcal{B}}'' \notin \Gamma_{\mathcal{P}'}^{(m',n')}$ such that $|\overline{\mathcal{B}}''| < m'-1$.

Since we have assumed that there is a unique solution for k, we re-write Equation (9) as Equation (10) in Figure 5. Let $\mathbf{M_k}$ denote the left-hand matrix in Equation (10), and let $\mathbf{a_k}$ denote the coefficient vector a_1 ... $a'_{m'-1}$. Let $\mathbf{M_k^{UL}}$

$$\begin{bmatrix} 1 & 1 & \cdots & 1^{m-1} & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & & \vdots \\ 1 & i & \cdots & i^{m-1} & \vdots & \ddots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & & \vdots \\ 1 & (m-1) & \cdots & (m-1)^{m-1} & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 & 1 & \cdots & 1^{m'-1} \\ 1 & \vdots & & \vdots & \vdots & \cdots & \vdots \\ 1 & \vdots & \ddots & \vdots & j & \cdots & j^{m'-1} \\ 1 & \vdots & & \vdots & \vdots & \cdots & \vdots \\ 1 & 0 & \cdots & 0 & (m'-1) & \cdots & (m'-1)^{m'-1} \end{bmatrix} \begin{bmatrix} k \\ a_1 \\ \vdots \\ a_{m-1} \\ a'_1 \\ \vdots \\ a'_{m'-1} \end{bmatrix} = \begin{bmatrix} s_1 \\ \vdots \\ s_i \\ \vdots \\ s_{m-1} \\ s'_1 \\ \vdots \\ s'_j \\ \vdots \\ s'_{m'-1} \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} 1 & \cdots & 1^{m-1} & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \vdots & & \vdots \\ i & \cdots & i^{m-1} & \vdots & \ddots & \vdots \\ \vdots & \cdots & \vdots & \vdots & & \vdots \\ (m-1) & \cdots & (m-1)^{m-1} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1^{m'-1} \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ \vdots & \ddots & \vdots & j & \cdots & j^{m'-1} \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ 0 & \cdots & 0 & (m'-1) & \cdots & (m'-1)^{m'-1} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{m-1} \\ a'_1 \\ \vdots \\ a'_{m'-1} \end{bmatrix} = \begin{bmatrix} s_1 - k \\ \vdots \\ s_i - k \\ \vdots \\ s_{m-1} - k \\ s'_1 - k \\ \vdots \\ s'_j - k \\ \vdots \\ s'_{m'-1} - k \end{bmatrix} \tag{10}$$

Figure 5. Equations for the proof of Theorem 2.

and $\mathbf{M_k^{LR}}$ denote the upper-left and lower-right square submatrices of $\mathbf{M_k}$

$$\mathbf{M_k^{UL}} = \begin{bmatrix} 1 & \cdots & 1^{m-1} \\ \vdots & \ddots & \vdots \\ (m-1) & \cdots & (m-1)^{m-1} \end{bmatrix}$$

and

$$\mathbf{M_k^{LR}} = \begin{bmatrix} 1 & \cdots & 1^{m'-1} \\ \vdots & \ddots & \vdots \\ (m'-1) & \cdots & (m'-1)^{m'-1} \end{bmatrix}$$

We express $\det(\mathbf{M_k^{UL}})$ as

$$\begin{aligned} \det(\mathbf{M_k^{UL}}) &= 1 \cdots (m-1) \begin{vmatrix} 1 & \cdots & 1^{m-2} \\ \vdots & \ddots & \vdots \\ 1 & \cdots & (m-1)^{m-2} \end{vmatrix} \\ &= 1 \cdots (m-1) \prod_{1 \le i,j \le m-1; i>j} (i-j) \end{aligned}$$

and observe immediately that $\det(\mathbf{M_k^{UL}})$ is non-zero; similarly, $\det(\mathbf{M_k^{LR}})$ is non-zero. Thus, $\det(\mathbf{M_k})$ is non-zero since $\det(\mathbf{M_k}) = \det(\mathbf{M_k^{UL}})\det(\mathbf{M_k^{LR}})$ (by Lemma 6).

Since $\det(\mathbf{M_k})$ is non-zero, then Equation (10) has a unique solution for $\mathbf{a_k}$ (by Lemma 5). If Equation (10) has a unique solution for $\mathbf{a_k}$, then Equation (9) has a unique solution for $\mathbf{a}$ (since we know k). But we have already established that we have infinitely many solutions for $\mathbf{a}$, and our assumption that we have a unique solution for k has led to a contradiction. Thus, we cannot uniquely determine k with the shares of shareholders in $\overline{\mathcal{B}}$ and $\overline{\mathcal{B}}'$. $\square$

5 Related work

New storage systems have emerged that use encryption or threshold secret sharing to preserve the long-term availability and confidentiality of data. In such systems, the storage nodes run code that implements the system, but are not trusted with plaintext data. Farsite [1, 8] and OceanStore [30] all encrypt replicas of the original data prior to storage. Publius [41] encrypts replicas, and in addition uses threshold sharing to creates shares of the encryption key; it then stores a share with each replica, so that one may reconstruct the key and decrypt a replica provided a threshold number of replicas are available. Farsite and OceanStore rely on replication to tolerate server failures; Publius simply assumes that a sufficient number of servers will remain available to reconstruct the encryption key. In our prototype system, we use threshold sharing to hide data from the servers (and avoid the key management problems associated with encryption), and use our VSR protocol to redistribute shares in response to the addition or removal of servers.

Other storage systems make stronger assumptions about server security to obviate the need for encryption or to use faster data dispersal algorithms. Pangaea [38] distributes plaintext replicas to servers, and uses a similar failure detector to Ensemble [24, 40]. Intermemory [10, 22] uses error-correcting encoding algorithms to disperse shares of data (also referred to as "slices" or "fragments") for servers; in order to recover from the loss of shares, the system reconstructs the data and redisperse new shares. e-Vault [19, 27] and OceanStore (for its deep archive storage mode) use encoding algorithms similar to those in Intermemory, but rely on having enough servers remain non-faulty to allow reconstruction of the original data. Since we assume that the servers are untrusted, we require a decentralized redistribution mechanism, i.e, VSR, to recover from server failures.

Our use of threshold sharing schemes to distribute shares of data, as opposed to keys, is a radical departure from that envisioned by Blakley and Shamir, who invented threshold schemes. In Shamir's (m,n) scheme [39], interpolation of an $m-1$ degree polynomial from m of n points yields a constant term in the polynomial that corresponds to the secret. In Blakley's scheme [6], the intersection of m of n vector spaces yields a one-dimensional vector that corresponds to the secret. Desmedt surveys other sharing schemes [12].

Our VSR protocol expands on the concept embodied in VSS schemes, that of protecting shareholders from a faulty dealer. Chor *et al* present a scheme in which the dealer and shareholders perform an interactive secure distributed computation [11]. Benaloh [3], Gennaro and Micali [20, 21], Goldreich *et al* [23], and Rabin and Ben-Or [34, 36] propose schemes in which the dealer and shareholders participate in an interactive zero-knowledge proof of validity; the schemes of Gennaro and Micali and of Rabin and Ben-Or are information-theoretically secure. Feldman [14] and Pedersen [33] present schemes in which the dealer broadcasts a non-interactive zero-knowledge proof to the shareholders. Beth *et al* [4] present a VSS scheme for monotone access structures based on finite geometries. Our VSR protocol differs from VSS schemes in that the multiple "dealers" of the new shares (the old shareholders) do not have the original secret, and must use other information to generate a proof for the new shareholders. Also, each new shareholder performs a two-part verification, first of the validity of its received subshares, and second of the validity of the shares used by the old shareholders to generate the subshares.

Other researchers present redistribution protocols that do not involve the physical redistribution of shares. Blakley *et al* consider threshold schemes that *disenroll* (remove) shareholders from the access structure with broadcast messages [5]; the new shareholders are a subset of the old ones. Cachin proposes a secret sharing scheme that *enrolls* (adds) shareholders in the access structure after the initial sharing [9]; the new shareholders are a superset of the old ones. Blundo *et al* present a scheme in which the dealer broadcasts messages to activate different, possibly disjoint, authorized subsets [7]. Blundo's scheme requires shareholders to have a share regardless of whether or not they are in the active authorized subset, in contrast to Desmedt and Jajodia's scheme. Our VSR protocol alters the access structure by physical redistribution of shares, and allows new shareholders to verify that they have valid shares.

We motivate the design of our archive system and our VSR protocol by the need to defend against mobile adversaries. Ostrovsky and Yung introduce the concept of a mobile adversary [32] that corrupts participants in a distributed protocol at a constant rate. Herzberg *et al* [25, 26] propose a PSS protocol in which each shareholder periodically distributes *update shares* to all other shareholders. Zhou, Schneider, and van Renesse propose a PSS protocol for asynchronous, wide-area networks, and employ it in an on-line certification authority [46]; they also independently postulated conditions similar to our SHARES-VALID and SUBSHARES-VALID conditions as sufficient for ensuring the validity of shares after protocol execution [45]. Our VSR protocol, unlike these PSS protocols, can redistribute shares to arbitrary access structures. However, we assume that there exists reliable broadcast among all participants in our protocol, which Zhou *et al* avoid in their protocol.

Frankel *et al* [16, 17, 18] and Rabin [35] propose PSS protocols in which each shareholder periodically distributes a subshare of its share to each of the other shareholders. Each shareholder combines the received subshares to generate a new share. A drawback of these protocols is that their witnesses for verification depend on the initial threshold scheme parameters m and n, and thus one cannot redistribute from an (m,n) to an (m',n') access structure.

Our VSR protocol, in contrast to the earlier PSS protocols, can guard against mobile adversaries with permanent compromise; that is, we can deal with compromise that cannot be recovered with a reboot operation. Of course, we still require that at any given point of time, the number of faulty shareholders in the current set of shareholders is less than the threshold value.

6 Summary

We have presented a verifiable secret redistribution protocol in the context of building archive systems. The archival nature of the system calls for heavyweight protection mechanisms to ensure the long-term availability and confidentiality of stored data. Additionally, we must account for the addition and removal of storage servers within the lifetime of the data. Our protocol uses threshold sharing schemes and incorporates a verification capability to support redistribution between arbitrary sets of shareholders.

We identified a vulnerability in Desmedt and Jajodia's redistribution protocol and proved that two conditions, SHARES-VALID and SUBSHARES-VALID, are sufficient to guarantee that new shareholders have valid shares after redistribution. We also proved that an adversary cannot combine old shares and new shares to reconstruct the secret, provided that the adversary has less than m old shares and m' new shares. Our redistribution protocol can tolerate up to $m-1$ faulty old shareholders (provided that there are at least m honest members). We pointed out that the identification and removal of faulty members is not immediately possible if the new members must rely on the old shareholders to distribute verification information. In the worse case, $\binom{n}{m} - \binom{n-m+1}{m}$ restarts are required to eliminate faulty shareholders and complete the protocol.

Our research follows other work that employs threshold schemes to build secure, distributed archive systems [27, 30, 41]. In contrast to earlier systems, ours can accommodate dynamic server group membership changes. We have implemented a simple prototype of the protocol itself using the Ensemble group communication toolkit [24], and are currently implementing a prototype archive system that is based upon the PASIS survivable storage system [43, 44].

Acknowledgements

This research is sponsored in part by the Army Research Office (ARO) under contract no. DAAD19-01-1-0485 and the National Science Foundation under contracts no. CCR-0121547 and no. CCR-0208853. This research is also sponsored in part by the DARPA/ITO OASIS program, under Air Force contract no. F30602-99-2-0539-AFRL.

We thank the members of the PDL Consortium (including EMC, Hewlett-Packard, Hitachi, IBM, Intel, Microsoft, Network Appliance, Panasas, Seagate, Sun, and Veritas) for their interest, insights, feedback, and support.

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ARO, NSF, DARPA, PDL Consortium members, or the U.S. Government.

References

[1] A. Adya, W. J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. R. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. P. Wattenhofer. FARSITE: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment. In *Proc. of the 5th Symp. on Operating Systems Design and Implementation*. Dec. 2002.

[2] R. A. Beaumont. *Linear algebra*. Harcourt, Brace & World, Inc., 1965.

[3] J. C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Proc. of CRYPTO 1986, the 6th Ann. Intl. Cryptology Conf.*, pp 213–222. 1987.

[4] T. Beth, H.-J. Knobloch, and M. Otten. Verifiable secret sharing for monotone access structures. In *Proc. of the 1st ACM Intl. Conf. on Computer and Communications Security*, pp 189–194. Nov. 1993.

[5] B. Blakley, G. R. Blakley, A. H. Chan, and J. L. Massey. Threshold schemes with disenrollment. In *Proc. of CRYPTO 1992, the 12th Ann. Intl. Cryptology Conf.*, pp 540–548. Aug. 1992.

[6] G. R. Blakley. Safeguarding cryptographic keys. In *Proc. of the Natl. Computer Conf.*, 1979.

[7] C. Blundo, A. Cresti, A. D. Santis, and U. Vaccaro. Fully dynamic secret sharing schemes. *Theoretical Comput. Sci.*, 165(2):407–440, Oct. 1996.

[8] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In *Proc. of SIGMETRICS 2000, the Intl. Conf. on Measurement and Modeling of Computing Systems*, pp 34–43. June 2000.

[9] C. Cachin. On-line secret sharing. In *Proc. of the 5th IMA Conf. on Cryptography and Coding*, pp 90–198. Dec. 1995.

[10] Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. Yianilos. A prototype implementation of archival intermemory. In *Proc. of the 4th ACM Intl. Conf. on Digital Libraries*, pp 28–37. Aug. 1999.

[11] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (Extended abstract). In *Proc. of the 26th IEEE Ann. Symp. on Foundations of Computer Science*, pp 383–395. Oct. 1985.

[12] Y. Desmedt. Some recent research aspects of threshold cryptography. In *Proc. of the 1st Intl. Information Security Workshop*, pp 158–173. Sept. 1997.

[13] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Technical Report ISSE TR-97-01, George Mason University, Fairfax, VA, July 1997.

[14] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. of the 28th IEEE Ann. Symp. on Foundations of Computer Science*, pp 427–437. Oct. 1987.

[15] Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal resilience proactive public-key cryptosystems. In *Proc. of the 38th IEEE Ann. Symp. on Foundations of Computer Science*, pp 384–393. Oct. 1997.

[16] Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Proactive RSA. In *Proc. of CRYPTO 1997, the 17th Ann. Intl. Cryptology Conf.*, pp 440–454. Aug. 1997.

[17] Y. Frankel, P. D. MacKenzie, and M. Yung. Adaptively-secure optimal-resilience proactive RSA. In *Proc. of ASIACRYPT1999, the 5th Intl. Conf. on the Theory and Application of Cryptology and Information Security*, pp 180–194. Nov. 1999.

[18] Y. Frankel, P. D. MacKenzie, and M. Yung. Adaptive security for the additive-sharing based proactive RSA. In *Proc. of PKC 2001, the 4th Intl. Workshop on Practice and Theory in Public Key Cryptography*, pp 240–263. Febraury 2001.

[19] J. A. Garay, R. Gennaro, C. S. Jutla, and T. Rabin. Secure distributed storage and retrieval. *Theoretical Comput. Sci.*, 243(1–2):363–389, July 2000.

[20] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Proc. of EUROCRYPT 1996, the Intl. Conf. on the Theory and Application of Cryptographic Techniques*, pp 354–371. May 1996.

[21] R. Gennaro and S. Micali. Verifiable secret sharing as secure computation. In *Proc. of EUROCRYPT 1995, the Intl. Conf. on the Theory and Application of Cryptographic Techniques*, pp 168–182. May 1995.

[22] A. V. Goldberg and P. N. Yianilos. Towards an archival Intermemory. In *Proc. of the IEEE Forum on Reasearch and Technology Advances in Digital Libraries*, pp 147–156. Apr. 1998.

[23] O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP statements in zero-knowledge and a methodology of cryptograhpic protocol design. In *Proc. of CRYPTO 1986, the 6th Ann. Intl. Cryptology Conf.*, pp 171–185. 1987.

[24] M. Hayden and R. van Renesse. Optimizing layered communications protocols. In *Proc. of the 6th IEEE Symp. on High Performance Distributed Computing*, Aug. 1997.

[25] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *Proc. of the 4th ACM Intl. Conf. on Computer and Communications Security*, pp 100–110. Apr. 1997.

[26] A. Herzberg, S. Jarekci, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Proc. of CRYPTO 1995, the 15th Ann. Intl. Cryptology Conf.*, pp 339–352. Aug. 1995.

[27] A. Iyengar, R. Cahn, J. A. Gray, and C. Jutla. Design and implementation of a secure distributed data repository. In *Proc. of IFIP/SEC 1998, the 14th Ann. Intl. Conf. on Information Security*. Sept. 1998.

[28] K. P. Kihlstrom, L. Moser, and P. Melliar-Smith. The SecureRing group communication system. *ACM Trans. on Information and System Security*, 4(4), Nov. 2001.

[29] A. I. Kostrikin. *Introduction to algebra*. Springer-Verlag, 1982.

[30] J. Kubiatowicz, D. Bindel, P. Eaton, Y. Chen, D. Geels, R. Gummadi, S. Rhea, W. Weimer, C. Wells, H. Weatherspoon, and B. Zhao. OceanStore: An architecture for global-state persistent storage. In *Proc. of ASPLOS IX, the Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, pp 190–201, Nov. 2000.

[31] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. Prog. Lang. Syst.*, 4(3):382–401, July 1982.

[32] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proc. of the 10th Ann. ACM Symp. on Principles of Distributed Computing*, pp 51–59. Aug. 1991.

[33] T. P. Pedersen. Non-iteractive and information-theoretic secure verifiable secret sharing. In *Proc. of CRYPTO 1991, the 11th Ann. Intl. Cryptology Conf.*, pp 129–140. Aug. 1991.

[34] T. Rabin. Robust sharing of secrets when the dealer is honest or cheating. *J. ACM*, 41(6):1089–1109, Nov. 1994.

[35] T. Rabin. A simplified approach to threshold and proactive RSA. In *Proc. of CRYPTO 1998, the 18th Ann. Intl. Cryptology Conf.*, pp 89–104. Aug. 1998.

[36] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. of the 21st Symp. on the Theory of Computing*, pp 73–85. May 1989.

[37] M. K. Reiter. A secure group membership protocol. *IEEE Trans. Softw. Eng.*, 22(1):31–42, Jan. 1996.

[38] Y. Saito, C. Karamonolis, M. Karlsson, and M. Mahalingam. Taming aggressive replication in the Pangaea wide-area file system. In *Proc. of the 5th Symp. on Operating Systems Design and Implementation*. Dec. 2002.

[39] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, Nov. 1979.

[40] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-based failure detection service. In *Proc. of the IFIP Intl. Conf. on Distributed Systems Platforms and Open Distributed Processing*, pp 55–70, Sept. 1998.

[41] M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. of the 9th USENIX Security Symp.*, pp 59–72. Aug. 2000.

[42] T. M. Wong, C. Wang, and J. M. Wing. Verifiable secret redistribution for threshold sharing schemes. Technical Report CMU-CS-02-114-R, Sch. of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, Sept. 2002.

[43] J. J. Wylie, M. Bakkaloglu, V. Pandurangan, M. W. Bigrigg, S. Oguz, K. Tew, C. Williams, G. R. Ganger, and P. K. Khosla. Selecting the right data distribution scheme for a survivable storage system. Tech. Rep. CMU-CS-01-120, Sch. of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, May 2001.

[44] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kiliççöte, and P. K. Khosla. Survivable information storage systems. *IEEE Computer*, pp 61–68, Aug. 2000.

[45] L. Zhou, F. B. Schneider, and R. van Renesse. APSS: Proactive secret sharing in asynchronous systems. In preparation.

[46] L. Zhou, F. B. Schneider, and R. van Renesse. COCA: A secure distributed on-line certification authority. *ACM Trans. Comput. Syst.*, 20(4):329–368, Nov. 2002.