G$^2_{its}$VSR: An Information Theoretic Secure Verifiable Secret Redistribution Protocol for Long-term Archival Storage

V. H. Gupta[*] and K. Gopinath[†]
Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore 560 012, INDIA

Abstract

Protocols for secure archival storage are becoming increasingly important as the use of digital storage for sensitive documents is gaining wider practice. Wong et al.[8] combined verifiable secret sharing with proactive secret sharing without reconstruction and proposed a verifiable secret redistribution protocol for long term storage. However, their protocol requires that each of the receivers is honest during redistribution. We proposed[3] an extension to their protocol wherein we relaxed the requirement that all the recipients should be honest to the condition that only a simple majority amongst the recipients need to be honest during the re(distribution) processes. Further, both of these protocols make use of Feldman's approach for achieving integrity during the (re)distribution processes. In this paper, we present a revised version of our earlier protocol, and its adaptation to incorporate Pedersen's approach instead of Feldman's thereby achieving information theoretic secrecy while retaining integrity guarantees.

Keywords: Archival Storage, Secret Sharing, Verifiable Secret Sharing, Secret Redistribution.

1. Introduction

Secure archival storage is needed to provide long term confidential storage. High availability is often an additional requirement. Servers can fail or could be compromised by an adversary, and therefore it should be possible to add and/or remove some of them as well. Data-integrity is another essential requirement. However, high availability combined with low latency is not a critical requirement. The primary goal, therefore, is to preserve secrecy over large time periods with data integrity and no loss of data.

In an earlier paper[3] (denoted by xVSR), we proposed a an extended version of the verifiable secret sharing protocol of Wong et al.[8]; we refer to their protocol as W^{3}VSR. Their protocol uses a combination of Shamir's secret sharing[6], Desmedt and Jajodia's[4] secret redistribution schemes incorporating Feldman's share verification scheme[1]. However, their paper requires that each of the receivers is honest during redistribution. In this paper, we relax the requirements so that only a majority among the receivers need be trustworthy during the (re)distribution process. For this, we have used a complaint mechanism similar to the one described by Gennaro et al.[2] in a different context (distributed key generation, DKG). We then present a stronger version (G$^2_{its}$VSR) that provides information theoretic security as contrasted with the protocol (G$^2_{cs}$VSR) that provides only computational security.

2. Verifiable Secret Redistribution

The emerging demand for long term confidentiality for digitally stored data has led to many design proposals for secure archival storage. High availability is often achieved by the use of redundancy which in turn is often achieved using replication. However confidentiality requirements complicate matters as high availability might imply availability for the malicious

[*] vhgupta(at)csa(dot)iisc(dot)ernet(dot)in
[†] gopi(at)csa(dot)iisc(dot)ernet(dot)in

snooper as well.

Moreover, in a typical long term storage system, there are two types of adversaries to be considered, namely static adversaries and mobile adversaries. In a distributed scenario with many servers, a static adversary means that only a few servers can be compromised and then their number remains strictly bounded even over arbitrarily long period. Therefore, once we know the upper-bound on their number, we can appropriately choose the values for the parameters of the secret sharing scheme. On the other hand, a mobile adversary means that the number of compromised servers keeps increasing and, given a sufficiently large time interval, there is no "upper bound" on the number of compromised servers. Therefore some "proactive" steps are taken, for example, say by rebooting/reinstalling a compromised server to restore trust, etc.

Similarly, secrecy comes in two flavors. Computational security means that an adversary with bounded computational ability cannot decrypt the secret even if it gets hold of the encrypted data. On the other hand, information theoretic security means that even if the adversary has unbounded computational abilities, it still will not be able to decrypt the encrypted data.

The simplest way to achieve the above goals of secrecy and longevity is to store the secret in a distributed way on many servers and put a constraint that a minimum number of shares be required to reconstruct the secret. Let n and m denote the "number of shares" stored and "the minimum number of shares required for reconstructing the secret" respectively. Shamir's secret sharing scheme provides a mechanism to do the same [6]. There are two basic weaknesses in this scheme. First, there is no way to guarantee that the shares are good enough to reconstruct the secret, and secondly since it implicitly assumes a long term upper-bound on the number of compromised nodes, m, the scheme remains vulnerable to mobile adversaries. These weaknesses are overcome using two key ideas, namely verification of shares, and proactive secret sharing. In verification of shares, the main trick is to introduce certain integrity checks on the shares so that one is sure that the shares are valid; these can be computational or information-theoretic secure witness functions. In the proactive secret sharing scheme, one redistributes the shares in such a manner that the secret remains the same while the shares change repeatedly. More over, the new shares are obtained in such a manner that even if an insufficient number of previously obtained shares from compromised nodes are used in conjunction with an insufficient number of new shares obtained from compromised nodes in the current situation, the adversary does not have sufficient information to be able to reconstruct the secret. Note that redistribution with reconstruction makes the reconstruction process a singular vulnerability during the redistribution process and thus undermines security. Hence, redistribution without reconstruction is a preferred choice.

2.1. A High Level Description of VSR

Let us say that there is a client C who has a secret k which is to be stored over a long time. The client stores it on n servers such that at least m shares are required to reconstruct the secret k. We call it a $[n, m]$ access structure. As time passes, the servers redistribute the shares to, say, n' servers such that at least m' are required to reconstruct the same secret k. Let the new structure be called the $[n', m']$ access structure. After passage of further time, another redistribution takes place, and so on. Finally the client may reconstruct the share at some stage.

Mathematically, the central operations are evaluations of a polynomial on a field (with coefficients chosen suitably; in this protocol, they are chosen randomly) and Lagrangian interpolations with one variable being the id of the server. Further, proactive secret sharing, as proposed by Desmedt and Jajodia [4], uses homomorphic properties of exponentiation. This facilitates computation of shares from sub-shares without reconstructing the secret.

For a uniform description of the various protocols we consider in this paper, we define a common set of assumptions and terminology.

2.2 Assumptions and Notation

Let the client be denoted as C while the underlying group communication system (GCS) is denoted as $\mathcal{G}$. The latter is needed to provide the following services to all the servers that are considered part of the group:

1. The communication channels used guarantee reliable delivery of messages.

2. There is a point-to-point private channel between any two nodes.

3. There is a broadcast channel such that all participating nodes receive any message sent on this channel.

4. An identity service is available so that any server can declare its identity in an authenticated manner. This also means that a node cannot spoof its identity.

Let p and r be two large primes such that $r = pq+1$. Let Z_p, and Z_r be the finite fields corresponding to modulo p, and modulo r arithmetic, respectively. Let the secret be k (where $k \in Z_p$).

For generating witnesses, a generator g is chosen from Z_r such that $g^p \equiv 1 \bmod r$.

Access structures are denoted by $[n, m]$, $[n', m']$,... where m, m',... shares from n, n',... respectively are needed for reconstructing the secret. Also, $n = 2m - 1$, $n' = 2m' - 1$,... so that the secret is reconstructible if and only if there is an honest majority. In an $[n, m]$ access structure the secret is distributed to n servers such that a necessary and sufficient condition to be able to reconstruct the secret is to know any m shares. Similarly an admissible set of servers is denoted by $\mathcal{B}$ with or without suitable subscripts.

2.3. W^{3}VSR

We reproduce the protocol from Wong et.al.[8]. It has two main parts, viz., initial distribution (denoted **INITIAL:**) and redistribution (denoted **REDIST:**), as described below:

INITIAL: To distribute a secret k to the access structure $[n, m]$:

1. Use the polynomial $a(i) = k + a_1 i + \cdots + a_{m-1} i^{m-1}$ to compute the shares $s_i = a(i)$ of k and send s_i to each of the n members over private channels.

2. Use generator g to compute $g^k, g^{a_1}, \ldots, g^{a_{m-1}}$, and send them to all n members over the broadcast channel.

3. Each party i verifies that:

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

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

4. If all i agree to commit, each i stores s_i and g^k. Otherwise they abort the protocol.

REDIST: Similarly, to redistribute k from access structure $[n, m]$ to $[n', m']$ using an appropriate authorized subset $\mathcal{B}$ (the set of recipients is denoted $\mathcal{B}'$):

1. For each sender $i \in \mathcal{B}$, use the polynomial $a'(i) = s_i + a'_{i1} j + \cdots + a'_{i(m'-1)} j^{m'-1}$ to compute the sub-shares $\hat{s}_{ij}$ of s_i and send $\hat{s}_{ij}$ to each of the n' members $j \in \mathcal{B}'$ over private channels.

2. For each $i \in \mathcal{B}$ use generator g to compute $g^{s_i}, g^{a'_{i1}}, \ldots, g^{a'_{i(m'-1)}}$, and send them to all n' members over the broadcast channel.

3. For each recipient $j \in \mathcal{B}'$, verify that:

$$\forall i, g^{\hat{s}_{ij}} \equiv g^{s_i} \prod_{l=1}^{m'-1} (g^{a'_{il}})^{j^l} \qquad (2)$$

   and:

$$g^k \equiv \prod_i (g^{s_i})^{b_i} \text{where} \quad b_i = \prod_{l \in \mathcal{B}, l \neq i} \frac{l}{l-i}. \quad (3)$$

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

4. If all j agree to commit, each j generates new shares s'_j: $s'_j = \sum_{i \in \mathcal{B}} b_i \hat{s}_{ij}$ where $b_i = \prod_{l \in \mathcal{B}, l \neq i} \frac{l}{l-i}$ and stores s'_{ij} and g^k. Otherwise, they abort the redistribution, and restart with a different set of senders $\mathcal{B}$.

The reconstruction simply consists in redistributing the secret with a single party as recipient (i.e. $\mathcal{B}' = \{\text{client}\}$).

As pointed out in our earlier work[3], one assumption of Wong et al.[8] is that during the distribution or all of the redistribution, all the receivers (the servers who will be receiving the shares or the sub-shares, as the case may be) are honest. Although it is not a completely unnatural assumption, it can be relaxed to facilitate a more realistic scenario in the long-term archival context. In a realistic scenario, for every (re)distribution, we may not start with "fresh" servers (added or rebooted just the instant before induction in to the group). Secondly, some of the servers could possibly be compromised during the (re)distribution phase. Finally, after the (re)distribution phase, compromises may take place depending upon adversarial activity. In light of this, it is desirable if we could treat server compromise as an independent event. Moreover, since we are making the assumption that no more than $m - 1$ servers ($m' - 1$ and so on, i.e. less than the threshold number required to reconstruct the secret) are compromised at a given instant, it would be better if we were able to work without making any assumption regarding the timing of their compromise.

In the presence of dishonest servers on the receiver group, step 4 in INITIAL and REDIST subsections can be exploited to repeatedly abort the protocol with resulting data-loss. Similar vulnerability exists during redistribution as well. These limitations can readily be circumvented if we incorporate a complaint mechanism that we discuss next. We had earlier described such a mechanism in [3] but we provide additional details.

2.4. A Possible Complaint Mechanism

The crux of the complaint mechanism is as follows: In case a receiver considers the share (sub-share) it received as "suspicious", it cannot "abort" the protocol. Rather, it must lodge (broadcast) a complaint against the sender, upon which the sender reveals (broadcasts) the share (sub-share) that it sent to the receiver. Now the majority can examine the revealed information, and if the share (sub-share) fails the verification test (we note that the witnesses necessary for verification are already known since they have been broadcast as part of the protocol), the receiving servers classify the sender as "VC" (indicating that a valid complaint exists against the sender). The original complaint mechanism was described in Ref[2], which uses Pedersen's VSS [5], and it was adapted for Feldman's VSS [1] for use in xVSR[3]. We will argue later (section 5) why there is no leakage of information inspite of revealing a subshare in the complaint mechanism.

Note that in our protocols xVSR (presented earlier in [3]), G$^2_{cs}$VSR and G$^2_{its}$VSR (both presented in this paper), no servers are disqualified (unlike in DKG[2]). The complaint mechanism is useful to the honest servers only inasmuch as to decide which share (or sub-share) is to be trusted. Another protocol is needed on how to always ensure majority of honest parties by penalizing VC-tagged ones but we will not discuss that here.

Now we present two protocols which combine verifiable secret redistribution with a complaint mechanism. The first one (section 3) is a revised version of xVSR[3] and provides computational security of the secret. The revision is necessitated since xVSR did not specify the combinatorial reasoning needed to identify the correct subset of replies for progress in the protocol. We will discuss this next below. The second (section 4) is a further refinement of the revised version and provides information theoretic security.

During the execution of all the VSR family of protocols, all the servers send their sub-shares during redistribution. A crucial question is which sub-shares are to be used for computing the new shares. We note that this confusion does not arise in W^{3}VSR, since all the receiver nodes are assumed to be honest. The receivers re-initiate (demand) redistribution from different admissible (a set containing more than threshold number of servers) sender sets till the receivers obtain consistent and valid (ascertained using VSS) new sub-shares. Since there is an honest majority in the senders, there certainly exists at least one admissible set which sends valid sub-shares. This can be shown using a combinatorial reasoning, the details of which are given in Wong et. al.[8] section 4.1.

Fortunately, a similar combinatorial reasoning can be extended for our design. An outline is as follows: An honest sender will not receive complaints from a majority of receivers (since there is a majority of honest servers in the receiver side as well) and hence it will not "lose" its credibility and will not be marked "VC" (akin to disqualification in DKG). Therefore, after each of the senders has communicated its sub-shares, there will be more than threshold number of senders who retain credibility (i.e. who do not have a majority of receivers lodging complaint against them). Since the identity of servers is known (a service provided by the communication infrastructure), the set of senders who have retained their credibility can be ordered (sequenced) by their id's. Now the honest receiver group uses a subset sequence generating algorithm $\mathcal{A}$ specified beforehand (see section 3 for details) and uses the first admissible set satisfying the constraints on the subshares (as given in the protocols). In our earlier paper[3], we had proposed to use the first of the many shares as needed. This strategy has a security weakness. If a dishonest sender sends "correct" sub-shares (meaning those satisfying sub-share verification), which however, correspond to an "incorrect" share (meaning that the share will not satisfy the share verification), then such a sender *may not* get classified in the "VC" category. However if honest receivers use the sub-shares sent by it to construct their shares, then these constructed shares cannot be used to reconstruct (or redistribute) the secret, for they may not satisfy the share verification. In the two protocols proposed here, we have solved the problem by generating a subset sequence using the algorithm $\mathcal{A}$ and using the first one in the sequence. This ensures that at every redistribution step, the secret does get redistributed in a valid manner, and at the most fewer than threshold shares can be obtained by the adversary.

3. G$^2_{cs}$VSR Protocol: Revised Version of xVSR

3.1. Preliminaries:

In addition to the preliminaries in W^{3}VSR, we make the following additional assumption. This is needed to make progress in the algorithm so that all the honest participants can decide on *one* admissible set of respondents: the first one in the precomputed sequence that satisfies a validity condition. An algorithm $\mathcal{A}(x, I, \mathcal{Z})$ is therefore agreed on as a part of initialization. The algorithm has two inputs, viz. x (an integer) and I (a set of server identities, with y elements, and $x < y$). The output $\mathcal{Z}$ is a sequence of subsets of I each containing exactly x members. It is clear that there are "y choose x" members in the sequence each containing a combination of x servers from the set I.

3.2. Protocol Execution

INITIAL: To distribute a secret k to the access structure (n, m):

1. The client C chooses a polynomial $a(i) = k + a_1 i + \cdots + a_{m-1} i^{m-1}$ to compute the shares $s_i = a(i)$ of k and sends the shares s_i to each of the n members over private channels.

2. The client uses generator g to compute $g^k, g^{a_1}, \ldots, g^{a_{m-1}}$, and sends them to all n members over the broadcast channel.

3. Each party i verifies that:

$$g^{s_i} \equiv g^k \prod_{l=1}^{m-1} (g^{a_l})^{i^l} \tag{4}$$

   - if the condition does not fail, node i keeps the share s_i as its share.
   - If the condition fails, i lodges a "complaint" against C by broadcasting a "complaint message". The "complaint message" is just a simple text "node i did not receive valid share". We note that the complaint is to be authenticated; this the underlying GCS (group communications system) facilitates.

     On lodging of the complaint by the node i, the client reveals, in the broadcast channel, the share $\hat{s}_i$ that it putatively sent to the node i. All the nodes now verify using equation 4 by using $\hat{s}_i$ in the place of s_i.

     - If the verification succeeds, node i takes the revealed share $\hat{s}_i$ as the share for itself.
     - Otherwise, when the verification fails, the nodes make a note that there is a valid complaint against the client. And the client is marked "VC"(Valid Complaint). Since the majority is honest, with a majority vote (using GCS facilities), they abort the protocol.

   Remark: In the absence of a signature scheme, a receiver has no way of asserting that it got a particular value from the sender. For example, the client may send an erroneous share, a correct witness (the witness corresponding to what should have been the value). However, when the complaint is lodged, the client reveals the correct share along with the correct witness. This is the reason why we have no way of identifying the dishonest party.

4. It is clear that if the protocol is not aborted, then the honest majority of the nodes have "correct" share.

REDIST: Similarly once can redistribute k from access structure (n, m) to (n', m'). Members of the first access structure are called sender nodes, and those of the second one called receiver nodes.

1. Each sender node i, uses the polynomial $a'_i(j) = s_i + a'_{i1}j + \cdots + a'_{i(m'-1)}j^{m'-1}$ to compute the sub-shares $\hat{s}_{ij} = a'_i(j)$ of s_i and sends $\hat{s}_{ij}$ to each of the n' members j over private channels.

2. Each sender node i uses generator g to compute $g^{s_i}, g^{a'_{i1}}, \ldots, g^{a'_{i(m'-1)}}$, and sends them to all receiver nodes (i.e. all the n' members of the receiver set) over the broadcast channel.

3. Each receiver node j, verifies that:

$$\forall i, g^{\hat{s}_{ij}} \equiv g^{s_i} \prod_{l=1}^{m'-1} (g^{a'_{il}})^{j^l} \qquad (5)$$

   - if the condition does not fail, receiver node j keeps the sub-share $\hat{s}_{ij}$ as its sub-share from sender node i.
   - If the condition fails, receiver node j lodges a "complaint" against sender node i by broadcasting a "complaint message". The "complaint message" is just a simple text "receiver j did not receive valid sub share from sender i". We again note that the complaint is to be authenticated, that is the underlying GCS (group communications system) facilitates.

     On lodging of the complaint by the receiver node j, the sender node i reveals, in the broadcast channel, the share $\hat{\hat{s}}_{ij}$ that it putatively sent to the node j. All the nodes now verify the correctness of the sub-share using equation 5 by using $\hat{\hat{s}}_{ij}$ in the place of $\hat{s}_{ij}$.

     - If the verification succeeds, receiver node j takes the revealed share $\hat{\hat{s}}_{ij}$ as the sub-share for itself.
     - Otherwise, when the verification fails, the receiver nodes make a note that there is a valid complaint against the sender node i. And the node i is marked "VC" by the receiver nodes.

       Remark: Since the complaint related transactions are performed over the broadcast channel, all parties view the *same* messages. It is clear that the honest nodes, who are following the protocol, will perform the same computation, and therefore will obtain the same results. Hence, all honest nodes have a consistent marking of the sender nodes. That is, if a sender node has been marked "VC" by an honest node, the same sender has been marked "VC" by *all* honest nodes. Similarly if a sender has not been marked "VC" by an honest node, *no other* honest node has marked this sender "VC".

     Remark: Here too like earlier we have no way of identifying the dishonest party.

4. It is clear that all honest receivers have marked the same sender nodes as "VC". Now the receiver nodes remove those sender nodes which are marked "VC". Since there is an honest majority amongst sender nodes, there are at least m sender nodes who do not have "VC" mark against them (even if a dishonest receiver complains against an honest sender, on sub-share revelation it becomes clear that the sender sent "correct" sub-share, and since there is an honest majority amongst receivers as well, at least m' receivers who do not mark the senders who sends/reveals "correct" sub-shares, as "VC"). Hence it is clear that the reduced set of senders (senders who are not marked "VC" by the node) contains at least m members, and each honest member has the same reduced set of senders. Let the number of such senders be denoted by U (for "untagged"). Now the receiver node sequences the sender nodes by their ids and makes a sequence of m-subsets us-

ing the sequencing algorithm $\mathcal{A}$. Let this sequence contain M (say) elements. It is clear that $1 \leq M \leq N$ where N is "U choose m". Let us denote the members of this sequence by $\mathcal{B}_u$ where $u = 1, 2, \cdots, M$.

5. Each sender node also broadcasts g^k. It is clear that all honest nodes broadcast the same (so the correct witness is broadcast by a majority). Each receiver node, takes this g^k as the witness for performing the verification performed in the following item.

6. Now the receiver nodes take these m-subsets one by one and check for equation for each u,

$$g^k \equiv \prod_i (g^{s_i})^{b_i} \text{where} \quad b_i = \prod_{l \in \mathcal{B}_u, l \neq i} \frac{l}{l-i}. \quad (6)$$

It is clear that there is at least one such $\mathcal{B}_u$ such that equation 6 is satisfied. The receiver nodes take the first such $\mathcal{B}_u$. The members of $\mathcal{B}_u$ satisfy the following properties. Firstly, that they are not marked "VC", and secondly their original "shares", from which they computed the "subshares" satisfy equation 6.

7. Each receiver node j computes its new share s'_j using sub-shares from sender set $\mathcal{B}_u$ by means of the formula: $s'_j = \sum_{i \in \mathcal{B}} b_i \hat{s}_{ij}$ where $b_i = \prod_{l \in \mathcal{B}, l \neq i} \frac{l}{l-i}$
and stores s'_j and g^k.

RECONSTRUCT: To reconstruct a secret k from the access structure (n, m):

1. The client C seeks shares from each node i. Each node i sends its share s_i to the client.

2. The client forms, using the algorithm $\mathcal{A}$, a sequence of m-subsets of the n server nodes. Let us denote this sequence $\mathcal{B}_u$ where $u = 1, 2, \ldots, M$, where $M \leq N$, N being "n choose m"

3. The client verifies in sequence successive $\mathcal{B}_u$, the test for the validity of shares given by the following equation (same as equation 6)

$$g^k \equiv \prod_i (g^{s_i})^{b_i} \text{where} \quad b_i = \prod_{l \in \mathcal{B}_u, l \neq i} \frac{l}{l-i}. \quad (7)$$

We note that the client has the witness g^k with it. However, we can easily relax this assumption. since each party has the witness g^k and if they send it to the client, the client can readily take the one sent by the majority.

4. The client reconstructs k using the first $\mathcal{B}_u$ for which the test succeeds. Since there are at least m honest nodes it is clear that the test succeeds for at least one $\mathcal{B}_u$. The reconstruction is done using the following equation:

$$k = \sum_i s_i b_i \text{where} \quad b_i = \prod_{l \in \mathcal{B}_u, l \neq i} \frac{l}{l-i}. \quad (8)$$

Now we present the second of our protocols.

4. $\mathbf{G}^2_{its}$VSR Protocol

In addition to the assumptions of the two earlier protocols, we also need to choose an arbitrary random number t (where $t \in Z_p$) that serves as the one-time pad for the secret. We also choose a generator h (like the already chosen generator g) for the exponential witness scheme. The generator h too is chosen from Z_r such that $h^p \equiv 1$ mod r and that $log_g h$ is not known.

4.1. Protocol Execution

The protocol structure is similar to the previous protocol except that in place of each share (sub-share), a share-pair (sub-share-pair) is sent. It is as if two "secrets" are being (re)distributed simultaneously. However one of the "secrets" is only a "random" number and is being used as a one time pad for blinding. Again, instead of one polynomial, two polynomials are chosen, one for each value of the pair.

INITIAL: To distribute a secret k in an information theoretic secure way to the access structure $[n, m]$:

1. The client C chooses a random number t from the field Z_p. Then the client chooses two polynomials $a(i) = k + a_1 i + \cdots + a_{m-1} i^{m-1}$ and $b(i) = t + b_1 i + b_2 i^2 + \cdots + b_{m-1} i^{m-1}$ to compute the shares $s_i = a(i)$ and $t_i = b(i)$ of k and t respectively and sends the pair of shares (s_i, t_i) to each of the n members over private channels.

2. The client uses generator g to compute $g^k, g^{a_1}, \ldots, g^{a_{m-1}}$, and the generator h to compute $h^t, h^{b_1}, \ldots, h^{b_{m-1}}$ and then computes the witnesses $g^k h^t, , g^{a_1} h^{b_1}, \ldots, , g^{a_{m-1}}, h^{b_{m-1}}$ and sends them to all n members over the broadcast channel.

3. Each party i verifies that:

   $$g^{s_i} h^{t_i} \equiv g^k h^t \prod_{l=1}^{m-1} (g^{a_l} h^{b_l})^{i^l} \qquad (9)$$

   - if the condition does not fail, node i keeps the share-pair (s_i, t_i) as its share-pair.
   - If the condition fails, i lodges a "complaint" against C by broadcasting a "complaint message" similar to that described in G$^2_{cs}$VSR. The "complaint message" is just a simple text "node i did not receive valid share". We note that the complaint is to be authenticated, this the underlying GCS (group communications system) facilitates.

     On lodging of the complaint by the node i, the client reveals, in the broadcast channel, the share-pair $(\hat{s}_i, \hat{t}_i)$ that it putatively sent to the node i. All the nodes now verify using equation 9 by using $\hat{s}_i$ and $\hat{t}_i$ in the place of s_i and t_i respectively. We note that here too like in G$^2_{cs}$VSR we have denoted the revealed pair as different from the originally sent pair. This is because in principle the revealed pair could be "different" from the one sent originally, for one or both of sender and receiver could be dishonest.

     - If the verification succeeds, node i takes the revealed share $\hat{s}_i$ as the share for itself.
     - Otherwise, when the verification fails, the nodes make a note that there is a valid complaint against the client. And the client is marked "VC" (Valid Complaint). Since the majority is honest, with a majority vote, they abort the protocol.

     Remark: We mention again that in the absence of a signature scheme, a receiver has no way of asserting that it got a particular value from the sender. For example, the client may send an erroneous share, a correct witness (the witness corresponding to what should have been the value). However, when the complaint is lodged, the client reveals the correct share along with the correct witness. This is the reason why we have no way of identifying the dishonest party in the sense that we cannot discern if the client sent an incorrect share in the first place or if the complainant had lodged a unwarranted complaint. This is exactly similar to the corresponding situation described in the previous protocol.

4. It is clear that if the protocol is not aborted, then the honest majority of the nodes have "correct" shares.

REDIST: Similarly, to redistribute the pair (k, t) from access structure (n, m) to (n', m'). Members of the first access structure are called sender nodes, and those of the second one called receiver nodes.

1. Each sender node i, uses the two polynomial $a'_i(j) = s_i + a'_{i1} j + \cdots + a'_{i(m'-1)} j^{m'-1}$ and $b'_i(j) = t_i + b'_{i1} j + \cdots + b'_{i(m'-1)} j^{m'-1}$ to compute the pair of sub-shares $\hat{s}_{ij} = a'_i(j)$ and $\hat{t}_{ij} = b'_i(j)$ of s_i and t_i respectively and sends $\hat{s}_{ij}$ to each of the n' members j over private channels.

   Remark: We note that, unlike in INITIAL, the senders do not choose new random one-time pads. Rather they take the "share" corresponding to t as the "random" pad. However, this does not weaken the protocol, since the original one time pad t can only be constructed by more than threshold number of nodes, and that requires an honest majority. And so long as t remains elusive, the witness $g^k h^t$ remains an information theoretic secure secret.

2. Each sender node i uses generator g to compute $g^{s_i}, g^{a'_{i1}}, \ldots, g^{a'_{i(m'-1)}}$, and also generator h to compute $h^{t_i}, h^{b'_{i1}}, \ldots, h^{b'_{i(m'-1)}}$ and sends their respective products, viz, $g^{s_i} h^{t_i}, g^{a'_{i1}} h^{b'_{i1}}, \ldots, g^{a'_{i(m'-1)}} h^{b'_{i(m'-1)}}$ to all receiver nodes (i.e. all the n' members of the receiver set) over the broadcast channel.

3. Each receiver node j, verifies that:

$$\forall i, g^{\hat{s}_{ij}} h^{\hat{t}_{ij}} \equiv g^{s_i} h^{t_i} \prod_{l=1}^{m'-1} (g^{a'_{il}} h^{b'_{il}})^{j^l} \tag{10}$$

   - if the condition does not fail, receiver node j keeps the sub-share-pair $(\hat{s}_{ij}, \hat{t}_{ij})$ as its sub-share from sender node i.
   - If the condition fails, receiver node j lodges a "complaint" against sender node i by broadcasting a "complaint message". The "complaint message" is just a simple text "receiver j did not receive valid sub share from sender i". We again note that the complaint is to be authenticated; this the underlying GCS (group communications system) facilitates.

     On lodging of the complaint by the receiver node j, the sender node i reveals, in the broadcast channel, the sub-share pair $(\hat{\hat{s}}_{ij}, \hat{\hat{t}}_{ij})$ that it putatively sent to the node j. All the nodes now verify the correctness of the sub-share using equation 10 by using $\hat{\hat{s}}_{ij}$ and $\hat{\hat{t}}_{ij}$ in the place of $\hat{s}_{ij}$ and $\hat{t}_{ij}$ respectively.

     - If the verification succeeds, receiver node j takes the revealed share $\hat{\hat{s}}_{ij}$ as the sub-share for itself.
     - Otherwise, when the verification fails, the receiver nodes make a note that there is a valid complaint against the sender node i. And the node i is marked "VC" by the receiver nodes.

       Remark: Since the complaint related transactions are performed over the broadcast channel, all parties view the *same* messages. It is clear that the honest nodes, who are following the protocol, will perform the same computation, and therefore will obtain the same results. Hence, all honest nodes have a consistent marking of the sender nodes. That is, if a sender node has been marked "VC" by an honest node, the same sender has been marked "VC" by *all* honest nodes. Similarly if a sender has not been marked "VC" by an honest node, *no other* honest node has marked this sender "VC".

     Remark: Needless to mention that in the absence of a signature scheme, a receiver has no way of asserting that it got a particular value from the sender. For example, the sender may send an erroneous share pair, a correct witness (the witness corresponding to what should have been the value). However, when the complaint is lodged, the sender then reveals the correct sub-share pair along with the correct witness. This is the reason why we have no way of identifying the dishonest party.

4. It is clear that all honest receivers have marked the same sender nodes as "VC". Now the receiver nodes remove those sender nodes which are marked "VC". Since there is an honest majority amongst sender nodes, there are at least m sender nodes who do not have "VC" mark against them (even if a dishonest receiver complains against an honest sender, on sub-share revelation it becomes clear that the sender sent "correct" sub-share, and since there is an honest majority amongst receivers as well, at least m' receivers who do not mark the senders who sends/reveals "correct" sub-shares, as "VC"). Hence it is clear that the reduced set of senders (senders who are not marked "VC" by the node) contains at least m members, and each honest member has the same reduced set of senders. Let the number of such senders be denoted by U (Untagged! say). Now the receiver node sequences the sender nodes by their ids and makes a sequence of m-subsets using the sequencing algorithm $\mathcal{A}$. Let this sequence contain M (say) elements. It is clear that $1 \leq M \leq N$ where N is "U choose m". Let us denote the members of this sequence by $\mathcal{B}_u$ where $u = 1, 2, \cdots, M$.

5. Each sender node also broadcasts $g^k h^t$. It is clear that all honest nodes broadcast the same (so the correct witness is broadcast by a majority). Each receiver node, takes this $g^k h^t$ as the witness for performing the verification performed in the following item.

6. Now the receiver nodes take these m-subsets one by one and check for equation for each u,

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

   It is clear that there is at least one such $\mathcal{B}_u$ such that equation 11 is satisfied. The receiver nodes take the first such $\mathcal{B}_u$. The members of $\mathcal{B}_u$ satisfy the following properties. Firstly, that they are not marked "VC", and secondly their original "pairs of shares", from which they computed the "pairs of sub-shares" satisfy equation 11.

7. Each receiver node j computes its new pair of shares (s'_j, t'_j) using sub-shares from sender set $\mathcal{B}_u$ by means of the formula: $s'_j = \sum_{i \in \mathcal{B}} b_i \hat{s}_{ij}$, and $t'_j = \sum_{i \in \mathcal{B}} b_i \hat{t}_{ij}$ where $b_i = \prod_{l \in \mathcal{B}, l \neq i} \frac{l}{l-i}$ and stores the pairs (s'_j, t'_j) and the witness $g^k h^t$.

RECONSTRUCT: To reconstruct a secret k from the access structure $[n, m]$:

1. The client C seeks shares from each node i. Each node i sends its pair of shares (s_i, t_i) to the client.

2. The client forms, using the algorithm $\mathcal{A}$, a sequence of m-subsets of the n server nodes. Let us denote this sequence $\mathcal{B}_u$ where $u = 1, 2, \ldots, M$, where $M \leq N$, N being "n choose m"

3. The client verifies in sequence successive $\mathcal{B}_u$, the test for the validity of shares given by the following equation (same as equation 11)

$$g^k h^t \equiv \prod_i (g^{s_i} h^{t_i})^{b_i} \text{where} \quad b_i = \prod_{l \in \mathcal{B}_u, l \neq i} \frac{l}{l-i}. \tag{12}$$

   We note that the client has access to the witness $g^k h^t$ with it. Since each party has the witness $g^k h^t$ and they send it to the client, the client can readily take the witness sent by the majority as the "true" witness. Since we assume that the majority is honest, it is clear that the client has the desired witness.

4. The client reconstructs k using the first $\mathcal{B}_u$ for which the test succeeds. Since there are at least m honest nodes it is clear that the test succeeds for at least one $\mathcal{B}_u$. The reconstruction is done using the following equation:

$$k = \sum_i s_i b_i \quad \text{where} \quad b_i = \prod_{l \in \mathcal{B}_u, l \neq i} \frac{l}{l-i}. \tag{13}$$

   We note that it is not necessary to reconstruct the random number t at this stage. This is because, the random number serves no utility other than a one time random padding, and thereby providing information theoretic secrecy.

5. Security Analysis of the Two Protocols

It is evident that both of these protocols have essentially similar design. The former has shares (sub-shares) while the latter has share-pairs (sub-share-pairs). Hence, in the following analysis we mention share (sub-share) to indicate both the solo and the paired case. The reasoning presented here is common to both the protocols.

We also note that the sender and the receiver group can be partitioned into two sets, viz., honest set and dishonest set respectively. For $[n, m]$ access structure, we denote the honest set by H and the dishonest set by D. Similarly for $[n', m']$ access structure, we denote the sets by H' and D' respectively, and so on. It is evident that during distribution (or redistribution) there are four different possibilities, each of which is discussed below:

1. **An honest server and an honest receiver.** In this case, assuming the reliability in message passing (the basic services assumed in the protocol which are provided by the infrastructure), there will be no complaint and therefore no disclosure of any share (sub-share) to the adversary.

2. **A dishonest sender and a dishonest receiver.** Since all dishonest servers can be assumed to be operating in collusion, there will be a complaint only if it can result in a denial of service type attack. The shares of all the dishonest parties can be assumed to be known to the adversary. However, as has been shown by Wong et al. [8], so long as the dishonest servers (faulty nodes) are less than the threshold, they cannot reconstruct the secret. So we must just guard against the possibility of

a denial of service attack. This is assumed to be taken care of by the underlying communication infrastructure which guarantees availability of message sending window as well as the reliability of delivery. This has already been mentioned earlier also.

3. **A dishonest sender and an honest receiver.** In the case of a complaint, the dishonest node has to reveal its message to the receiver, since the sender is dishonest, its data can be assumed to be available to the adversary and hence its revelation in reaction to the complaint yields no additional information to the adversary. Another possibility is that a dishonest server sends "wrong" shares (or sub-shares) to some honest nodes and "correct" shares (or sub-shares) to the remaining honest nodes. After the complaint is lodged, the dishonest server can "pretend" honesty by now revealing "correct" shares (or sub-shares) meant for these complainant nodes. In such a case the complainant nodes modify their shares (or sub-shares) to this new "correct" value. We note, that the "correctness" of the shares (or sub-shares) is being verified using the procedure given in VSR [8]. As mentioned earlier it is Feldman's VSS [1] adapted for VSR [8].

4. **An honest sender and a dishonest receiver.** This is a tricky case. If a dishonest receiver complains on its own behalf, the sender reveals the information meant for the dishonest node. Since this information is already available to the adversary, she gains no new information. However, if a dishonest node can complain on behalf of an honest node, then the information meant for the honest node will be broadcast (as part of the complaint redressal) and this is additional information that becomes available to the adversary. This situation has not explicitly been mentioned by Gennaro et al. [2] since it is a non-issue in the context of distributed key generation. Nevertheless, in our case, we can circumvent it by mandating that each complaint be "authenticated" by the complainant. If we assume that the complainant identity is reliably known through the communication infrastructure, a dishonest node cannot complain on behalf of an honest server and hence no additional information is revealed to the adversary. This is critical in our adaptation of the complaint mechanism.

5.1. Outline of the Proof of Secrecy

First we show that the secret continues to be available using induction.

In step 4 of the initial distribution step, since the client is honest, the secret gets distributed (with at most fewer than the threshold number of shares getting disclosed to the adversary) to the honest majority of the receivers during the initial distribution step.

In the redistribution subsection, since there is a persistent majority of honest nodes, the secret gets redistributed to the honest majority (with at most less than threshold number of shares getting disclosed to the adversary) at each of the redistribution step. This argument is applied successively till the penultimate step to reconstruction, therefore, the penultimate access structure ($[n^*, m^*]$) has an honest majority.

Next, since the penultimate access structure has the secret (as shown in the previous paragraph), the secret can be reconstructed by the client using honest majority (at least equal to the threshold number of shares in the secret sharing scheme). Hence the secret remains available to the client.

Secondly, we show that confidentiality is assured. At every step, at most fewer than threshold number of shares can be obtained by the adversary (as discussed in section 2.4). Wong et al.[8] (in Theorem 2 on VSR Security) have shown that two successive sets of less than threshold number of shares are not sufficient to reconstruct the secret. Their proof technique can readily be applied inductively (we omit the details) to show that any finite number of sets of less than threshold number of shares are insufficient for the reconstruction of the secret. Hence confidentiality is assured. □

6 Conclusions

We have presented an extension of the VSR algorithm that is resilient to mobile adversaries, a contingency that needs to be handled in long-term archival storage. In addition, we have also extended it so that the protocol is information theoretic secure. POTSHARDS[7] is another recent approach that has information theoretic secrecy but uses a RAID-based approach instead of verifiable secret redistribution.

Acknowledgments

The authors acknowledge the support from the Ministry of Information Technology, Government of India, Grant No. MITO 014. The authors also thanks their colleagues M.C. Dharmadeep for lively discussions, and Matthieu Moy for pointing out lacunae in the xVSR description.

References

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

[2] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Proceedings of EUROCRYPT'99, Springer LNCS 1592*, pages 295–310, 1999.

[3] V. H. Gupta and K. Gopinath. An extended verifiable secret redistribution protocol for archival systems. In *International Conference on Availability, Reliability and Security (ARES)*, pages 100–107, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

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

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

[6] A. Shamir. How to share a secret. *Comm. ACM*, 22(11):612–613, November 1979.

[7] M. W. Storer, K. Greenan, E. L. Miller, and K. Voruganti. Potshards: Secure long-term storage without encryption. *Proceedings of the 2007 USENIX Technical Conference*, June 2007.

[8] T. M. Wong, C. Wang, and J. M. Wing. Verifiable secret redistribution for archive systems. *Proceedings of The First IEEE Security in Storage Workshop*, December 2002.