Efficient Dynamic-Resharing "Verifiable Secret Sharing" Against Mobile Adversary

Noga Alon[*] Zvi Galil[†] Moti Yung[‡]

March 25, 1995

Abstract

We present a novel efficient variant of Verifiable Secret Sharing (VSS) where the dealing of shares is dynamically refreshed (without changing or corrupting the secret) against the threat of the recently considered *mobile adversary* that may control all the trustees, but only a bounded number thereof at any time period.

VSS enables a dealer to distribute its secret to a set of trustees, so that they are assured that the sharing is valid and that they can open it later, and further no small group of trustees can open it prematurely. Recently, such sharing of cryptographic tools gained much attention, e.g., in the context of "escrowed cryptography" where a user enables a group of trustees to potentially open its information (when authorized by the court). Our dynamic-sharing VSS allows for mobile adversary attacking different sets of trustees at different time periods (modeling, e.g., network viruses that get spread as well as get killed).

Technically, we concentrate on simple direct methods that are combinatorial and number-theoretic in nature, and employ only simple public-key functions (and no other cryptographic tools as did previous methods). We also present constant round protocols (e.g., a single round VSS) and do not use general (polynomial time, but inefficient) tools. We essentially reduce n out of $t < n/2$ VSS to n out-of n one (assuming ex-or homomorphic encryption), then we reduce dynamic resharing VSS to static VSS, finally we reduce proactive VSS (dynamic VSS with no dealer presence after sharing) to our dynamic resharing VSS.

1 introduction

Verifiable secret sharing (VSS), introduced by Chor et al. [7], is an important primitive for sharing cryptographic data and control as well as a tool for other applications in distributed computing protocols (a recent application is key escrow systems such as fair cryptosystems [17]). VSS enables a dealer to transmit a secret to a set of trustees (share-holders), so that up to a threshold of trustees cannot get the secret and later a threshold of trustees (or more) can reconstruct the secret. These are the properties of secret sharing [21, 4]. Further, the sharing is verified: after the secret is shared the trustees are assured that reconstruction is possible, otherwise they reject the sharing.

In applications like key escrow (e.g., the Clipper encryption chip, but with voluntary key choice as in [17]), the dealer keeps on storing the secret but the trustees may open it upon request (this is exactly the escrow-agent scenario where a court asks the agnets to open the secret). The verifiability

[*] Tel-Aviv University

[†] Columbia University and Tel-Aviv University

[‡] IBM Research, T.J. Watson

of the sharing is crucial in such applications since the trustees when required (by court order) to open the shared secret, must be able to do so. Thus, we have to force the dealer to perform a valid sharing, and otherwise reject its actions.

Two basic methods for realizing VSS are known to withstand a *linear number of faults*. The first is based on general zero-knowledge techniques or simulation of private channel and the embedding the secret in a polynomial function; these methods are general, tolerate up to $n/2$ faults, but also generally inefficient (e.g., [13, 3, 22]. The second method is a direct (and efficient) but requires, in addition to public key system, homomorphic functions (Exponentiation over a finite field) that encrypt a polynomial function nicely and make VSS easy [10, 18] (polynomial interpolation is a traditional secret sharing method [21]). In contrast, it is interesting to note that our method will not rely on such polynomial-embedding in an encryption method (we use different encryption assumption not known before to be sufficient for efficient highly fault-tolerance VSS). We are, in fact, able to tolerate linear faults say $n/3$ faults (as many as needed for say broadcast or agreement protocols). We essentially reduce such VSS to the simpler n-out-of-n VSS.

VSS, as defined, assumes that the secret is long-lived (like a cryptographic key that is used for a long period of time). Yet the models assume that the adversary is restricted to a sub-set of trustees for the entire life of the secret. Given the fact that networks are dynamically attacked by hackers and viruses, a new model was suggested recently by Ostrovsky and Yung [19]. It assumes a bound on corrupted trustees within a time period, but lets the adversary be mobile and be able to change the set of corrupted trustees from period to period. This models the fact that viruses are detected and cleaned up, while the attacker keeps its efforts to break into various new machines. We consider here a dynamic-resharing VSS, where every time period the dealer is able to verifiably refresh the shares of the trustees, so that they are uncorrelated with previous shares. This forces the adversary to corrupt the required threshold of trustees within a time-period bound which is a harder task than the one without the time period limitation. This enables us to consider mobility of the adversary (other recent works in this model are in [19, 6, 16].)

In more detail, the dynamic-resharing VSS, assume that we have VSS and there are time-periods in which the sharing is refreshed. As long as at each time period the adversary is limited to up to t trustees, it cannot learn the secret prematurely, and it cannot prevent the trustees from opening the secret whenever required. Further, the dealer cannot corrupt the secret, the protocol either updates the shares or detects the cheating attempt of the dealer in which case the shares are not getting updated to prevent their potential corruption. (This last property extends the verifiability of the initial VSS to the update stages).

We remark that dynamic-resharing VSS is a different protocol from mobile-adversary proactive Secret Sharing as presented in [19, 16] (which use polynomial-based sharing methods). In the former, the dealer has a special role where it keeps on having the secret (as in escrow systems applications [17]). The goal is to keep the refreshing of secret verifiable. In the later, the secret was given to the community and forgotten by the dealer and the refresh is done by the community itself (so the set of applications of such a system is different, e.g. maintaining a secure data-base by the community). We show connections between the two and implement the second based on the first.

Our Results: The results achieved are four-fold:

1. We present a dynamic-sharing VSS.
2. It is constructed from a VSS protocol which uses simple probabilistic encryption tools yet tolerates a linear number of faults.

3. Further, the VSS protocol is constant round (and non-interactive in case of honest parties).

4. Finally we show how proactive VSS is reduced to dynamic resharing VSS.

2 Preliminaries

2.1 Crytptographic Background

We employ public keys [9, 20] and in particular Goldwasser and Micali's probabilistic encryption method [14]. It is based on composite numbers $N = PQ$ where N is a Blum integer, that is $P = Q = 3(\mod 4)$. This assures that -1 does not have a square root while its Jacobi symbol is $+1$. That is, take any element x from the multiplicative group mod N, $x \in Z_N^*$, square it get the set $QR = \{x^2(\mod N)\}$ this gives the set of quadratic-residues (q.r.'s), take this number and multiply by -1 to get the set $QNR = \{-x^2(\mod N)\}$ this is the set of quadratic non-residue (q.n.r.'s).

It is assumed that it is hard to distinguish a random element from QR from such an element from QNR, in the sense that any circuit that is able to distinguish between the two sets with a non-negligible (say inverse polynomial) probability, must be super-polynomial in size. (In general inverse polynomial probability is considered substantial while smaller than any super polynomial is negligible). On the other hand, this task becomes easy when one knows the factorization of N (P, Q). In [14] the two sets QR and QNR are used to encrypt a bit: squares (residues) to encrypt 0, and non-squares (non-residues) as above to encrypt 1. For a bit b, we denote $enc(b)$ an element from QR in case $b = 0$ and an element from QNR otherwise. This is the original probabilistic encryption method.

In our protocols, the dealer uses a Blum integer N as its key, the same key throughout the execution of the protocol.

We will employ the following simple properties of the probabilistic encryption of [14]. Note that when one multiply a q.r. with a q.r. the result is a q.r. while multiplying a q.n.r. with a q.n.r. results in a q.n.r. On the other hand, multiplication of a q.r. amd a q.n.r. (or q.n.r. with a q.r.) gives a q.n.r. as a result. Note further, that this multiplication rules of the ciphertexts of probabilistic encryptions corresponds to ex-or-ing (sum modulo 2) of the cleartext.

2.2 Model and Problem Definition

The parties in our protocols are communicating polynomial-time Turing machines [15], that flip *on-line* coins, read communication tapes and internal (private) tapes, and write messages and internal tapes. The on-line nature of coin-flipping is important to coping with mobile adversaries.

We assume a system of n trustees $\mathcal{A} = \{P_1, P_2, \ldots, P_n\}$ and a dealer D that will share secret bits, (we will describe one bit b) among the trustees. The goal of the scheme is to prevent the adversary from learning x. At the same time, the adversary cannot prevent the trustees $\mathcal{A}$ from reconstructing x themselves when they need to, after a turning point in the protocol.

2.2.1 Communication Model

the dealer D and each trustee in $\mathcal{A}$ is connected to a common broadcast medium. We assume for simplicity (to avoid dealing with authenticity of users) that each message origin is known (that is, each user has a Bulletin Board [8]). The property is that when a message sent from any party, it instantly reaches all the others and the identity of its sender gets globally known. We assume the system is synchronized and rounds are well defined, on each round all parties act based on information available from previous rounds and local coin-flips. We assume that each server in $\mathcal{A}$ has a source of true on-line randomness – a coin that is flipped on demand and is not known in advance.

For now, we assume that the adversary can corrupt at the start of the protocol up to $t = n(1/2 - \epsilon)$ trustees, and it may or may not corrupt the dealer, (in particular 1/3 faulty processors are possible which is the bound also required to implement a broadcast channel via Byzantine Agreement Protocols). Corrupting a trustee means both learning its memory and making it behave arbitrary.

2.3 The problem

A Verifiable Secret Sharing (VSS) is a protocol between a dealer and a set of trustees, all are polynomial-time probabilistic communicating machines. The VSS is a two part protocol.

The first part: sharing, by which a dealer with input a secret (bit) at the end of this stage each of the n trustees has a private value and decides to accept or reject the sharing.

The second part: reconstruction, where the trustees pull their values to gether and reconstruct an output value.

We would like the following properties to hold:

1. Secrecy: till reconstruction, no set of up to $t, t < n/2$ trustees can get any information about the secret. (Our notion is computational: no such set will be able to get an advantage in computing the secret).

2. Reconstructability: After the first part, if validity of sharing is assumed, at any moment any group of at least $n - t$ trustees can produce the output via the second part protocol and with overwhelming probability the result is the secret.

3. Verifiability: After the first part, the trustees can decide whether the part is successful or not. If successful, the trustees will decide that the sharing is valid and accept, otherwise they will reject the sharing. A honest dealer D can always cause the trustees to decide "valid" (in the presence of up to t faults), while a dishonest dealer that executes a sharing protocol part that reconstructability cannot be successful, will be rejected by the majority.

The dynamic resharing has additional properties which are in the spirit of the above problems. We require that a misbehaving dealer will not be able to corrupt its shares throughout. More details about this variant are in section 6.

3 The Construction Ideas

We describe below how the dealer sends its bit b to the set of trustees. (For many bits the protocol is run concurrently).

3.1 The Assignment Idea:

The first idea is random *family - of- committees assignment* (assignment for short) which generalizes the idea of partition the participants. The network will be organized by an assignment, which consists of a number of *families*, each family consists of a number of *committees*. The latter are subsets of processors (committees may overlap).

We require some properties from an assignment. It is good (called "ϵ-terrific") if (1) in each family there is a committee of only good guys and (2) in most (say 90 percent) of the families no committee consists of bad guys only. (A good random assignment which we will employ is analogous to but more involved than "Bracha assignment" [5].) For number of bad guys $t = (1/2 - \epsilon)n$, a good assignment exists, furthermore a random assignment is good with overwhelming probability $1 - 2^{(-K)}$ where K is the security parameter.

Remark: We can make the random assignment good with probability as high as the probability that the underlying public keys are secure. We first assume an assignment is given, then we will show how to generate one by the processors themeselves.

3.2 The Encoding

Given an assignment, the encoding of a bit will be as follows: Let m be the number of families and r the number of committees per family. The bit b will be encrypted m times, we call these repetitions $b_1, ..., b_m$.

Then, each time we will use partitioned encryption: The bit b will be distributed to r bits ($r-1$ of which are random) $b_{i,1}, b_{i,2}, ...b_{i,r}$, where

$$\sum_{j=1}^{r} b_{i,j} = b(\mod 2)$$

(the sum modulo 2 of these bits is b). The bit $b_{i,j}$ will be "sent encrypted" to the j-th committee of the i-th family.

Let $enc(b_{i,j})$ = a random q.r. if $b_{i,j} = 0$, and random q.n.r. otherwise. Recall that when $\sum_{j=1}^{r} b_{i.j} = b(\mod 2)$ then

$$\prod_{j=1}^{r} enc(b_{i,j}) = enc(b)$$

and let for all i, $ENC(b) = ENC(b_i)$ and (abusing notation) we represent it as a vector:

$$ENC(b_i) =< enc(b_{i,1}), ...enc(b_{i,r}) >$$

For each vector $ENC(b_i)$ (n-out-of-n sharing) the bad guys will not get b since one committee is good in each family and therefore one bi,j is not known ahead of time (since the XOR of the other perhaps opened bits of $b_{i,1}, ..b_{i,r}$ is just a random bit independent from the partial opened bits of other families).

4 The Protocol

Assume we have an assignment of processors, and each committee has a committee public key with corresponding private key known to its members. We next show that:

Theorem 1 *(QR is hard) There is a single-round protocol for VSS in a broadcast network with up to any subset of $O((1/2-\epsilon)n)$ being faulty processors (given assignment into committees as preprocessing).*

THE VSS PROTOCOL

Part 1: Distribution of the secret

- 1a. The dealer encodes its bit b (as described above) as $ENC(b)$ (a value it is committed to). It broadcast ENC(b) and the individual vectors For each family it independently encode this value as a vector encoding: $ENC(b_i), i = 1, .., m$ for each family F_i. It broadcasts the vector encodings.
- 1b.The dealer also sends its private random bits used in the encryption of $enc(b_{i,j})$ To the j-th committee of the i-th family using the committee's public key.
- 2a. Each trustee verifies that $ENC(b_i)$ is the same value for all i, by verifying that for all i:

  $$\prod_{j=1}^{r} enc(bi, j) = ENC(b)$$

  and all elements has Jacobi symbol +1; this shows that the "vector encodings" represent the same bit b that the dealer committed itself to.
- 2b. If in one of the rounds a trustee catches the dealer cheating , it announces this (in fact all honest trustees will complain). The dealing is cancelled if there is a protest by a majority.
- 2c. Also, each party that gets random bits from the dealer that are different from the bits used in the encryption protests. The parties can verify the protest and reject the dealing if protest justified, or ignore the protest. This is one round of protest.

  When this single round is over this is the **TURNING POINT** of the protocol, from which on the dealer is committed to a specific value.

Part 2. Reconstruction of the secret

- 1. Each member (in each committee for all families) opens the bits of the encodings. The parties calculate the resulting bits

  Note that bad committees can simply disappear or all present a false random bits (assuming they collaborated with the bad dealer, etc.) or behave in any devious way.

  As long as there is a family where each committee bit of which is revealed the bit is reconstructed.

Let us sketch the proof of the theorem above and check the properties of VSS. We have to argue that this is secure, as long as the dealer is honest. Note that in each family there is a bit which is in the hand of a good committee that has the information about the bit encrypted randomly using [14] semantically secure system. The random bits used in the encryption were encrypted by the committee's public key, which is a good key (distributed, in turn, encrypted under good keys of honest participants)– so the dealer hides these bits semantically.

The formal proof will show that an advantage in guessing the bit b can be translated to the fact that the dealer's or the committee's good members' keys or the committee keys are not semantically secure (a contradiction).

Note that a honest dealer will be able to get into the turning point successfully as only up to t (dishonest) trustees may unjustifyingly complain at any time.

On the other hand note that 0.9 of the families have a good participant in any committee and these bits will be opened, so a successful cheating have to give these honest participants bits not corresponding to the committee encrypted bit, but then this will be publicly protested.

In reconstruction we are assured that a family exists which all its committees' bits are opened, this will generate the witness for the encryption of b.

5 Preprocessing

Next we describe the assignment, the preprocessing protocol and its properties. We show that:

Theorem 2 *There exists a constant-round preprocessing protocol that in a network with up to any subset of $O((1/2 - \epsilon)n)$ being faulty processors, assign the processor to an "ϵ-terrific" assignment, each committee with a public key, where good committees have secure keys.*

5.1 The Assignment

Let $N = \{1, 2, ..., n\}$ be a set of n elements.

An (n, m, r, s)-assignment A is a collection $(F_1, F_2, ..., F_m)$,where each F_i is family of r subsets $\{S_{i,j}\}, j = 1, ..., r$ of N that satisfy $|S_{i,j}| \leq s$ for $i = 1, ..., m$, $j = 1, ..., r$.

For a subset T of N, and a subset S of N, S is T-bad if S is a subset of T,

S is T-good if S is a subset of $N - T$.

A family $F = (F_1, F_2, ..., F_r)$ of subsets of N is T-reasonable if at least one of the Fj's is T-good.

F is T-wonderful if no F_j is T-bad.

Finally, an (n, m, r, s)-assignment $A = (F_1, F_2, ..., F_m)$ is T-terrific if each Fi is T-reasonable and at least $0.9m$ of the F_i's are T-wonderful.

The assignment A is ϵ-terrific if it is T-terrific for every subset T of N that satisfies

$$|N - T|/|T| >= 1 + \epsilon$$

The next proposition shows that for every n and every $\epsilon > 0$, there exists an ϵ-terrific (n,m,r,s)-assignment A, where m, r, s are polynomials (of degree $O(1/\epsilon)$ in n. Given n and $\epsilon > 0$, let k satisfy

1. $k \geq \max\{2n + 2, 100\}$, and

2. define s, r, and m by: $s = ceiling\{2 \log k / \log(1+\epsilon)\}$, $r = ((2+\epsilon)^s)/k$, $m = 10k$.

Notice that $s = O(\log k/\epsilon)$ and $r = k^{O(1/\epsilon)}$.

Proposition 1 *For every integer n and $\epsilon > 0$,there exists an ϵ- terrific-assignment for every m, r, s that satisfy (1) and (2). Moreover,if k, m, r, s satisfy (1) and (2) and we construct randomly an (n, m, r, s)-assignment $A = (F_1, ..., F_m)$ where $F_i = (S_{i,j}), j = 1, ..., r$ and each $S_{i,j}$ is chosen by picking s random elements (with repetitions) from N, then the probability that A is ϵ-terrific is at least $1 - 2^{(-k/2)}$.*

The proof uses some standard probabilistic arguments including Chernoff's Inequality [1].

5.2 Preprocessing Protocol

We assume each trustee has a public key published and the dealer has one based on a Blum integer. (This fact can be certified earlier, a zero-knowledge certification was shown in [11]).

THE PROTOCOL

- Choosing a random assignment which is represented by :
    1. Each trustee commits to $mrs \log n$ bits which represent an assignment.
    2. Each trustee opens it commitment. Then the bits of trustees which fully open their commitments are bitwise ex-ored together to result in an assignment.
- Let the trustee with a smallest ID in a committee be the leader of the committee. Each leader chooses a public key for the committee, publishes it (with the committee tag $\{i, j\}$) and then transfers its corresponding secret key to the committee members, using probabilistic encryption of each committee member.

The following propositions imply Theorem 2.

Proposition 2 *The assignment is ϵ-terrific with probability $1 - 2^{-k}$ for any subset of misbehaving parties of size up to $(1/2 - \epsilon)n$.*

Assume the dishonest trustees are committed and they have a hidden random string that when opened, they may decide how to open their strings and ex-or their strings with this random string. The idea of the proof is that the set of faulty trustees has an exponential number of ways to behave and determine outputs (after commitment), but without knowing a random starting point determined by the commitment of the good trustees: the bit-wise sum modulo 2 of the strings of commitments of the good trustees. However the density of bad assignments is exponentially smaller than the number of strings the dishonest trustees may have. This means that a set of faulty guys will not be able to hit an assignment which does not qualify – still with an exponentially vanishing probability.

This is true when they start from a random string. They actually start from a string that is the ex-or of the honest guys' commitments. If they can bias it now much better than when against a random unknown string, then the encryptions of the commitment by honest trustees can be broken (a contradiction).

Proposition 3 *The committee key of any good committee is secure with respect to the adversary.*

This is implied by the fact that a good trustee chooses with very high probability a good key and probabilistically encrypts it with good keys and the key is stored at secure memory (good trustees' memory). An encryption by the committee's key which is semantically secure remain such given the probabilistic encryption of it with honest trustees' keys.

Note that the above protocol is a constant round preprocessing.

6 Mobile Adversary and Dynamic-Resharing

Now we have the computation being partitioned into time periods at the end of which a resharing protocol is executed by the dealer and the trustees. We will show that:

Theorem 3 *(QR is hard) There is a constant-round dynamic-sharing VSS protocol in a broadcast network with up to any subset of $O((1/2-\epsilon)n)$ being faulty processors at each time period.*

6.1 The Mobile Adversary Model

We now change our underlying paradigm so that an adversary does not control a server forever. Instead, an adversary can succeed in breaking-in for limited periods of time and move on to attack other servers. In other words, the adversary is mobile as was initially suggested in [19]. This assumption is motivated by two facts; first, intrusions in networks (such as software modification, viruses, etc.) are eventually exposed and cleared, and second the fact that machines are easily accessible through the network for an adversary to attack. The main goal behind dynamic protocols solution is to make the information gained by the adversary during a break-in useless during future break-ins (to the same or other servers), thus neutralizing the power given by the mobility property.

We assume the computation is divided into periods of time, at the end of a period of time there is an update phase. If a server is corrupted during an update phase, we consider the server as corrupted during both periods adjacent to that update phase. We assume that the adversary corrupts *no more* then $t \leq n(1/2-\epsilon)$ out of n parties. (Note that there are enough "good parties" at any time period). If at the time period prior to the update there are t_1 corruptions and t_2 corruptions at the time-period after the update and t_3 corruption during the update then $t_1+t_3 \leq t$ and $t_2+t_3 \leq t$.

Corrupting a trustee means both learning its memory and making it behave arbitrarily. When a trustee is cleaned, it waits till the next resharing step and starts afresh (announcing its . We require that as long as the dealer is not corrupted throughout, its secret remains secure, and that the adversary is unable to prevent reconstruction of the message at any point. Let us define the protocol in more details.

6.2 The problem

A Dynamic-Sharing Verifiable Secret Sharing (VSS) is a protocol between a dealer and a set of trustees. Its first part is sharing as in VSS, its last part is reconstruction as in VSS. Between the two stages it has time periods, in each time period at most t trustees are controlled by the adversaries (as explained above). At the end of a period a resharing protocol is performed, with the following properties. At the end of the resharing the trustees either reject and hold the old shares, or accept and update the shares they hold. Reconstruction is maintained throughout. We would like the following security properties to hold:

1. Secrecy: As long as at each period the adversary has up to $t, t < n/2$ trustees, it cannot get any information about the secret.
2. Reconstructability: If the original sharing was valid, no further resharing can reduce the probability of reconstruction by more than a negligible probability.
3. Verifiability: The trustees can decide whether the resharing is successful or not. If successful, the trustees will decide to update, otherwise they will reject the sharing. A honest dealer D can always cause the trustees to decide "valid" (in the presence of up to t faults), while a dishonest dealer that executes a resharing protocol part that reconstructability cannot be successful, will be rejected by at least $n - t$ trustees.

Note that for secrecy, the dealer has to kept honest (otherwise the secret is learned). For reconstructability, any behavior of the dealer cannot hurt this property.

6.3 Basic ideas

In the protocol to follow we have to refresh the public-keys at each update (resharing) so that previous keys learned are not valid anymore. This limits the power of historic record.

We then let the dealer refresh the shares. To do this we use the fact that an encryption when multiplied with a q.r. does not change its cleartext value. Let us review how it looks in the cleartext domain. So a family will get a random vector of cleartext bits with sum modulo 2 equals 0, each committee will ex-or its new bit with its old bit. As a result each committee now has a newly random bit independent of any previous partial bit pattern. At the same time the value of the family bit (the "family value") remains the same. At any period the adversary is only able to get partial bit encryptions earlier and partial updates or partial update and new shares which are less than the required threshold. Recall that the assignment is such that any subset of size t cannot "cover" any family in full.

Note that this is a reduction where the ex-or properties of the encryption, and the resharing of zero are employed.

6.4 The Dynamic-Sharing Protocol

THE DYNAMIC RESHARING PROTOCOL (STAGE l)

- 0. Each trustee i sends a new public key $E_{i,l}$.

  0a. Each head of a committee sends a new committee key.

- 1. The dealer encodes the bit 0 (as described above) as $ENC^l(0), i = 1, .., m$ for each family F_i.

- 2. The dealer proves that $ENC^l(0)$ indeed encodes the same bit 0.

  It computes, $ENC^l(0) = e^l = e^l i = enc^l(bi, 1) * enc^l(bi, 2) * ... * enc^l(bi, r)$, each family's vector gives a random $ENC(0)$ It shows that $e^l 1$ is a quadratic residue as well. To this end the protocol of [2] is used (the proofs to all trustees can be done in parallel and they take constant round– just think about the trustees as a special verifier). Each trustee can observe the transcript of messages exchanged by the other trustees and the dealer. (This shows that the encryptions are all of the value 0, namely are quadratic residues).

- 2.a If in one of the rounds a trustee catches the dealer cheating , it announces this fact by revealing its secret bits used in the interaction and the trustees get convinced that the dealer has cheated.

- 3. All trustees compute the accumulated vector encryption of every committee which is the encryption of its bit at stage l

  $$enc^l(bi, j) = enc^{l-1}(b_i, j) * enc^{l-1}(0, j)$$

  which is
  $$enc^l(b_i, j) * enc^{l-1}(0, j) * ... * enc^1(0, j) * enc(b_i, j)$$

  (This is the initial encrypted bits encryption multiplied by all the updates).

  Each trustee sends a list of all the accumulated public encryptions.

  3.a. Each trustee takes the majority of echoed encryptions and for as the right encryption.

  3.b. The dealer computes majority and decrypts the correct current bits and sends to the trustees. (It sends the random bits used in the encryption of $enc(b_{i,j})$ To the j-th committee of the i-th family using the committee's public key.)

  3.c. Committees complain about the bits they receive, complaints are verifiable, if a complaint is valid the dealer is dishonest, his resharing is ignored (each trustees restores its state prior to the protocol). Otherwise, update takes place, and old state is erased.

  When this is over this is the **START NEW PERIOD POINT**.

Note that any choice of t faulty guys, will keep the assignment such that every family has a good committee, and the majority of families will be able (if necessary) to open the right encryption.

7 Proactive VSS from Dynamic-Resharing

Next we will discuss how we can use the dynamic-resharing to have a proactive secret sharing (a secret maintained by the community). In such a protocol, after the first sharing, the dealer will forget its secret and the community will maintain the secret (secrecy and reconstructability is verifiably kept).

7.0.1 Dynamic-Resharing VSS vs. proactive VSS

To claim secrecy in dynamic-resharing VSS we assume that the dealer does not get corrupted. In [19, 16] "proactive secret sharing was defined" which deals with the case that the secret is shared initially (securely say) and then the trustees maintain it by themselves and assure secrecy and reconstructability. Our goal here is maintaining secrecy and reconstructability for the scenarios where the dealer re-deals its secret (e.g. for the case of fair cryptosystems where the trustee gives its secret to escrow agents (trustees)). The dealer is active at all times in this dynamic protocol and our goal is to "refresh dealing" rather than maintaining secrecy by the parties where the secret is fully distributed. We note that the previous proactive VSS schemes employed polynomial-based schemes.

7.1 The result

We will show that:

Theorem 4 *(QR is hard) There is a proactive VSS protocol in a broadcast network with up to any subset of $O((1/2-\epsilon)n)$ processors being faulty at each time period.*

We only sketch the ideas below.

The first idea is that the prover in the dynamic-resharing actually needs to know the square of the encryption rather than the factorization of the key N. Thus, every user can serve as a dealer of an encryption of a zero and prove the validity of its actions– being the actual dealer does not help!

The second idea is a transformation of a dynamic resharing protocol where anyone can refresh the shares, into a proactive VSS.

In fact, we will have the following initialization steps: (1) A Sharing stage protocol. (2) A sharing of the shares, using the same protocol each user will share its shares. At this point the secret can be erased.

Then we maintain the shares. At the end of each time period we have a refresh protocol. Each dynamic resharing is done by all users (each user, or at least $t+1$ of them to make sure one is honest, rerandomizes the shares, acting as a dealer). All shares-of-shares get refreshed. We have the following two refresh stage:

- (1) Before refreshing, users that have recovered in the period must have a chance to rejoin. So they should get from holders of shares of their share to send them secretly their share for reconstruction, so they can rejoin gracefully.

In fact, we can have a global "secret echoing" step, where a user sends a new public key first, and the parties send to him secretly the random secret bits of the shares of its share encrypted with this new key, and the parties also publish the current share-of-share. This gives the party a "reconstruction of its share". Now we are ready for refresh.

- (2) Each party (or at least at least $t + 1$ of them to assure honest participant) is refreshing (re-randomizing) all the shares-of-shares. This is done via the dynamic-resharing protocol above.

  Note that this can be viewed as maintaining a memory against mobile adversary. Share-of-share ideas have been used in various ways in the distributed computing literature, and were first introduced in [12]. (As in the dynamic resharing, only updates which are valid are taken into accounts and other resharings are ignored).

A few remarks: First note that maintaining only the shares directly is not enough as a share is held by a committee and the adversary can erase it– slowly erasing all committees' bits. But, shares of shares enable global maintenance via intermediate reconstruction when needed. A misbehaving processor that sends a weak key is controlled by the adversary which can anyway learn its secret to start with (and does not need the other processors to open its share– in any case at most t shares get opened).

8 Conclusions

We have presented a new method that relies only on probabilistic encryption [14] with its ex-or homomorphism and still gives a VSS protocol with linear faults. Previous methods give more faults (up to 1/2 faults, but this is not meaningful in a broadcast environment as agreement requires anyway at most 1/3 to be allowed to fail). On the other hand we can rely on quadratic residuosity being hard and have a direct and efficient method (no need to have encryption based on Discrete Logarithm, and also we achieve the quality of hiding information as defined by Goldwasser and Micali [14]). The fact that this can be done directly and efficiently (one round static VSS) but under complexity assumptions other than discrete logarithm hardness is interesting by itself and makes it incomparable with previous solutions.

The new concept presented is in the area of coping with "mobile adversary". It achieves immunity against mobile adversary in interesting applications where the dealer is present. This area is interesting given the nature of open public networks and their potential threats. The idea is to have a Dynamic-Resharing VSS which is shown and constructed based on the static VSS, and can be used to implement proactive VSS as a basic maintenance primitive in such environments.

References

[1] N. Alon, and J. Spencer, *The Probabilistic Method*, (1991), John Wiley and Sons.

[2] M. Bellare, S. Micali, and R. Ostrovsky, *Perfect Zero Knowledge in Constant Rounds,* Proceedings of the 22th Annual Symposium on the Theory of Computing, 1990, pp. 482–493.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," ACM STOC 1988, 1–9.

[4] G.R. Blakley, *Safeguarding Cryptographic Keys*, AFIPS Con. Proc (v. 48), 1979, pp 313–317.

[5] G. Bracha, *An $O(\log N)$ Expected Round Randomized Byzantine Generals Protocol*, Proc. of ACM STOC 1985.

[6] R. Canetti and A. Herzberg, Maintaining Security in the Presence of Transient Faults, Crypto 94.

[7] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, *Verifiable Secret Sharing and Achieving Simultaneous Broadcast*, Proc. of IEEE Focs 1985, pp. 335-344.

[8] J. Cohen and M. Fischer, *A robust and verifiable cryptographically secure election scheme*, Proc. 26th Annual Symposium on the Foundations of Computer Science, 1985, pp 372–382.

[9] W. Diffie and M. Hellman, *New Directions in Cryptography* , IEEE Trans. on Information Theory 22 (6), 1976, pp. 644-654.

[10] P. Feldman, *A Practical Scheme for Non-Interactive Verifiable Secret Sharing*, Proc. of the 28th IEEE Symposium on the Foundations of Computer Science, 1987, 427-437

[11] Z. Galil, S. Haber and M. Yung, *Minimum-Knowledge Interactive Proof for Decision Problems*, SIAM J. Comp., 18, 1989, pp 711–739.

[12] Z. Galil, S. Haber, and M. Yung, "Cryptographic computation: secure fault-tolerant protocols and the public-key model," Crypto 87.

[13] O. Goldreich, S. Micali, and A. Wigderson, *How to play any mental game*, Proceedings of the Nineteenth annual ACM Symp. Theory of Computing, 1987, pp 218–229.

[14] S. Goldwasser and S. Micali, *Probabilistic Encryption*, J. Com. Sys. Sci. 28 (1984), pp 270-299.

[15] S. Goldwasser, S. Micali and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, Siam J. on Computing, 18(1) (1989), pp 186-208.

[16] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, *Proactive Secret Sharing*, submitted.

[17] S. Micali, *Fair public-key cryptosystems*, Crypto '92.

[18] T. P. Pedersen, Distributed Provers with Applications to Undeniable Signature, Eurocrypt '91. 1991.

[19] R. Ostrovsky and M Yung, *How to withstand mobile virus attacks*, Proc. of the 10th ACM Symposium on the Principles in Distributed Computing, 1991, pp. 51-61.

[20] R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signature and Public Key Cryptosystems*, Comm. of ACM, 21 (1978), pp 120-126.

[21] A. Shamir. *How to share a secret*, Commun. ACM, 22 (1979), pp 612-613.

[22] T. Rabin and M. Ben-Or, *Verifiable Secret Sharing and Multiparty Protocols with Honest Majority*, STOC 1989, ACM, pp. 73-85.

[23] G. J. Simmons. *An introduction to shared secret and/or shared control schemes and their application*, In G. J. Simmons, editor, *Contemporary Cryptology*, pp. 441–497. IEEE Press, 1992.