Proactive Secret Sharing with Low Bandwidth and Fine Granularity

Qiming LI, Charng Rang GUO and Ee-Chien CHANG

November 2003
Technical Report

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

JAFFAR, Joxan
Dean of School
Abstract

We consider $(K,N)$ proactive secret sharing schemes whereby a large number of secrets are shared by $N$ servers, and the secrets are reconstructible from shares in any $K$ servers. The schemes are proactive in the sense that the shares can be refreshed periodically to withstand mobile adversaries. Motivated by the difficulties encountered in practical handling of large number of secrets, we propose to look into two requirements. In contrast to the classical schemes, which require exchanges of linear number (w.r.t the number of secrets) of random bits during refreshment, we take the communication bandwidth into account and attempt to limit the number of random bits exchanged (low bandwidth requirement). Thus, each server has to refresh all shares based on the small number of random bits it receives. For practical considerations, each individual secret should be reconstructible without revealing other secrets (fine granularity requirement). The two requirements are conflicting because low bandwidth requirement imposes dependencies among shares, which might hinder fine granularity. Current known schemes achieve one or another, but not both. We propose a scheme that considers both requirements. The scheme also maintains many features found in classical schemes, for instance, it can be verifiable with minor modifications.

1 Introduction

A \((K, N)\) secret sharing scheme splits a secret (for example, a binary file) into \(N\) shares, which are distributed to \(N\) servers. The knowledge of any \(K - 1\) shares will not reveal the secret and the secret is reconstructible by putting any \(K\) shares together. When \(K < N\), it is also known as a \(K\) out of \(N\) threshold scheme. Shamir gave such a scheme in 1979 [8].

The Shamir scheme is vulnerable to a mobile adversary, who given long enough time, is able to compromise many servers. A proactive secret sharing scheme takes into account of such mobile adversaries. Under this scheme, the servers periodically refresh their shares based on information exchanged among them. The refreshments are done in a way so that the security and proper operations are maintained if the mobile adversary does not control \(K\) servers simultaneously in a same time period. In the proactive setting, no individual holds full knowledge of the secrets. Thus it is infeasible to designate a trusted third party to compute the refreshments based on the original secrets, and redistributed them to the servers. The mobile adversaries are studied in [6] and an example of such proactive schemes can be found in [4].

In practice, proactive secret sharing schemes are very difficult to implement. We identify two fundamental issues that hinder practical implementation. Firstly, classical schemes require exchanges of linear number (w.r.t the number of shares) of random bits during refreshment. The large amount of data exchanged might overload the network and lead to problems in synchronization. Thus, the number of random bits exchanged should be limited. We call this the low bandwidth requirement.

Another practical issue is on the independency among different secrets. It is often required that each secret can be reconstructed without revealing other secrets. We call this requirement fine granularity. If fine granularity is not a concern, one could employ method proposed in [5], in which all the secrets are encrypted using a relatively shorter key, which is then shared using modified Shamir scheme, such as in [4, 1].

Intuitively, the requirements on bandwidth and fine granularity are conflicting. To achieve low bandwidth, the refreshments of different secrets should be based on small amount of information exchanged, and thus are related. However, for fine granularity, we need independencies among them. Note that the schemes in [4, 1] achieve fine granularity but not low bandwidth, while [5] is in the opposite.

Another notable requirement for secret sharing is verifiability. A scheme is verifiable if the information exchanged during refreshment can be verified to be consistent. This is important because the corrupted data (due to error or malicious activities) will lead to failure in reconstruction. Chor et al. [2] introduced verifiable secret sharing (VSS), and the schemes in [3, 7] are also verifiable.
The concerns of verifiability and our main focus (low bandwidth and fine granularity) can be separated. For clarity, we will not discuss verifiability here. Nevertheless, by some modifications, our scheme can achieve verifiability while maintaining low bandwidth and fine granularity requirement. The is done by integrating known VSS as a building blocks, just like what is done in [4].

Contributions. We identify two interesting conflicting requirements for proactive secret sharing schemes, low bandwidth and fine granularity, that are crucial in practical implementations. We propose a \((K, N)\) proactive secret sharing scheme, Chaining Scheme, that achieves both requirements. The Chaining Scheme also maintains certain desirable properties found in known schemes. In particular, it can be modified to achieve verifiability.

2 Preliminaries & Notations

For the \((K, N)\) scheme, there are \(N\) servers, \(P_1, P_2, \ldots, P_N\). The secrets to be shared are \(Z_1, Z_2, \ldots, Z_M \in \mathbb{Z}_n^*\) where \(n\) is a large composite that satisfies certain properties as stated in detail in Section 3.2. The number \(M\) is large and the actual value is of no particular interest here\(^*\).

Under the proactive setting, the shares are refreshed periodically. Similar to many previous works (such as [4]), we assume that the servers are synchronized. We also assume that there exists a private channel between each pair of servers so that shares can be distributed securely.

We assume that time is divided into periods. At the end of each time period, there is an update phase, during which the shares are refreshed. Let \(s_{m,j}^{(t)}\) be the share of the \(m\)-th secret, held by the server \(P_j\) in period \(t\), where \(t = 0, 1, 2, \ldots\). When \(t = 0\), it is the share initially held by the server. The initial shares \(s_{m,j}^{(0)}\) are computed and distributed by a trusted dealer. It is assumed that the shares can be distributed securely using methods like those in [3, 7]. Once the initial shares are distributed, all information, in particular the secrets, held by the dealer is purged and lost permanently. Thereafter, no individual knows any secret unless it is reconstructed by a group of \(K\) servers.

For simplicity, when focusing on one refreshment, we drop the notation \(t\) and refer to the shares before the refreshment as \(s_{m,j}\) and the shares after that as \(s'_{m,j}\). After a share is refreshed, we expect the servers (who are not compromised by the adversary), to purge the outdated shares.

\(^*\)\(M\) should be bounded by a polynomial of \((\log n)\). This is because we assume that the input size is \((\log n)\), and it is computationally infeasible to process, say, \(n\) number of secrets.
We consider threshold scheme, thus, any $K$ servers can reconstruct the secrets by putting their shares together. Specifically, in any time period $t$, the secret $Z_m$ can be reconstructed from any $K$ shares of the secret $s^{(t)}_{m,j_1}, s^{(t)}_{m,j_2}, \ldots, s^{(t)}_{m,j_K}$, where $j_1, j_2, \ldots, j_K$ are $K$ distinct integers.

2.1 Adversary/Security Model

We consider mobile (or adaptive) adversary who can choose some servers to compromise in any time period during the execution of the protocol. We assume that the adversary is passive (or “honest but curious”). When a server is compromised, the adversary knows all information (the shares) held by it, but the server still follows the protocol. Since outdated shares are purged, if an adversary compromise $P_j$ in time period $t$, it would obtain only the shares $s^{(t)}_{m,j}$ for all $m$.

There is another type of adversary, who is active (or malicious) in the sense that they can make the compromised servers behave arbitrarily. However, as mentioned in Section 1, concerns about active adversaries can be separated from our main focus, and they can be dealt with by integrating VSS into our scheme, like what is done in [4]. Note that special care is required in the integration, so that low bandwidth is maintained.

By introducing dependencies among the shares we achieve low bandwidth. However, as a trade-off, the security of our scheme is computational, while the original Shamir’s scheme is unconditionally secure. We seek a proactive scheme such that, it is computationally infeasible for an adversary to reconstruct any secret, if it does not know the shares in at least $K$ servers within one time period.

3 Proposed Scheme

Similar to Shamir’s scheme, our scheme is also based on polynomial interpolation. In fact, it can be considered as a special case of Shamir’s scheme.

We will describe our main ideas in Section 3.1, and introduce our settings in Section 3.2. The details of the scheme will be given in Section 3.3.

3.1 Main Ideas

To achieve low bandwidth, instead of refreshing the shares of each secret independently, in the proposed scheme, each server receives the refreshment of only one share. Other refreshments are successively derived from the received value. That is why we name our scheme Chaining Scheme.

Before describing the scheme, let us investigate a few “patches” to the Shamir scheme that do not meet the requirements.


**Pseudo Random Number Generator (PRNG).** During refreshment, one of the servers broadcasts a seed. Based on this seed, and a commonly agreed PRNG, each server \( P_j \) randomly chooses a polynomial \( f_i \) for the \( i \)-th share. The refreshed share is computed by \( s'_{i,j} = s_{i,j} + f_i(j) \). Although achieving low bandwidth, this “patch” is not secure in many ways. For instance, if an long-lived adversary controls a server all the time, it will know how all other servers refresh their shares. By comprising \((K - 1)\) more servers (each in different time period), this adversary can reconstruct all the secrets.

**Chain Squaring.** During refreshment, each server \( P_i \) picks a polynomial \( f_i \) and send \( f_i(j) \) to \( P_j \). Let \( f = f_1 + f_2 + \ldots + f_N \). After receiving all the data, each server \( P_j \) know \( f(j) \). The \( P_j \) uses \( f(j), f^2(j), f^4(j), \ldots \) for refreshement. Specifically, the first share is refreshed as \( s'_{1,j} = s_{1,j} + f(j) \), and the \( m \)-th share is refreshed as \( s'_{m,j} = s_{m,j} + f^{2(m-1)}(j) \), for \( m > 1 \). In this case, even though there are some dependencies between shares for different secrets, it is difficult for the adversary to make any use of the dependencies. However, since the degree of the polynomial increase quickly, it is not clear how to reconstruct the secret. In other words, it seems that this “patch” is so secure that legitimate reconstruction is also computationally infeasible.

The Chaining Scheme is based on chain squaring described above. The main idea is to limit the degree of the polynomials. This is done by designate a special identity \( v_j \) to each server \( P_j \). The server \( P_j \), instead of using \( f(j) \) as the refreshment, uses \( f(v_j) \) as the refreshment. The \( v_j \) satisfies this property: its order in the multiplicative group \( \mathbb{Z}_n^* \) is \( K - 1 \) (i.e. \( v_j^{K-1} = 1 \mod n \)). Thus, we can treat \( f^k \) a polynomial of degree at most \( K - 2 \) for any positive integer \( k \). To illustrate, when \( n = 341 \), we can see that \( 4^5 = 47^5 = \ldots = 1 \mod n \). Therefore, if we use these numbers as the identities, the degrees of the polynomials will be limited to at most 4.

However, using the above construct, the polynomial \( f^k \) might contain non-zero coefficient for the constant term. This will hinder reconstruction. To overcome, we introduce one more variable to the polynomial. Instead of \( f^k(v_j) \), the refreshment for \( s_{i,j} \) is \( x_j f^k(v_j) \). By choosing the right \( n \), \( x_j \)’s and \( v_j \)’s, efficient reconstruction is possible.

### 3.2 Parameters

We now describe the parameters required by the proposed \((K, N)\) scheme. First, we need to pick a composite \( n \). Let \( V_n \) denote the set of integers in \( \mathbb{Z}_n^* \) that are of order \((K - 1)\). The integer \( n \) should be chosen such that \( |V_n| > N \). Then we choose
$N$ distinct $v_i \in V_n$ to be the identity of server $P_i$. The identities of the servers are known by all (including the adversary), and fixed during the life time of the system.

In addition, we choose another $N$ values $x_i \in \mathbb{Z}_n^*$ for each host $P_i$, which are also fixed and known by all parties. In order for the construction of secrets to be always successful, we require that those values are chosen such that the following matrix is $K$-wise linearly independent.

$$
\begin{bmatrix}
1 & x_1 & x_1 v_1^1 & \cdots & x_1 v_1^{K-2} \\
1 & x_2 & x_2 v_2^1 & \cdots & x_2 v_2^{K-2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_N & x_N v_N^1 & \cdots & x_N v_N^{K-2}
\end{bmatrix}
$$

(1)

For large enough $n$, we can easily find the $x_i$’s that satisfy the above properties. In fact, a randomly chosen sequence satisfies the requirement with high probability. Furthermore, we need to choose a large integer $W$, whose size is comparable to that of $n$. For example, we could choose $W = \sqrt{n}$.

In sum, given $N$ and $K$, the parameters required are $n$, $W$ and the sequences $v_i$’s and $x_i$’s.

### 3.3 Chaining Scheme

This scheme can be described by the two stages, Initial Dealing and Refreshing stage, and the reconstruction operation.

**Initial Dealing.** Similar to Shamir’s scheme, the dealer does the following steps.

§1. For each secret $Z_m$, the dealer randomly choose $K - 1$ integers, $r_{m,1}, r_{m,2}, \ldots, r_{m,K-1} \in \mathbb{Z}_n^*$ to form a polynomial.

$$f_m(x) = r_{m,1}x + r_{m,2}x^2 + \ldots + r_{m,K-2}x^{K-2} + r_{m,K-1} \pmod{n}$$

§2. For each server $P_i$ and secret $Z_m$, the dealer computes the shares

$$s_{m,i}^{(0)} = Z_m + x_i f_m(v_i) \pmod{n}$$

and sends $s_{m,i}^{(0)}$ to $P_i$ securely.

**Refreshing.**

§1. Each server $P_i$ randomly chooses $K - 1$ integers $r_1, r_2, \ldots, r_{K-1} \in \mathbb{Z}_n^*$ to form a polynomial

$$g(x) = r_1 x + r_2 x^2 + \ldots + r_{K-2} x^{K-2} + r_{K-1} \pmod{n}.$$
§2. For each server $P_j$, $P_i$ computes

$$\alpha_{i,j} = g(v_j) \pmod{n}$$

and sends $\alpha_{i,j}$ to $P_j$ via the private channel between them.

§3. After receiving all $\alpha_{i,j}$’s, $P_j$ now can compute the new share $s^{(t+1)}_{m,j}$ for secret $Z_m$.

$$s^{(t+1)}_{m,j} = s^{(t)}_{m,j} + x_j \sum_{i=1}^{N} \alpha_{i,j} W^{m-1} \pmod{n}. \quad (2)$$

Reconstruction. Reconstruction of a secret $Z_m$ requires shares from $K$ honest servers. The share from honest server $P_i$ is of the form

$$s_{m,i} = Z_m + x_i \sum_{k=0}^{K-2} v_i^k c_k \pmod{n} \quad (3)$$

where $c_k$’s are some coefficients. Since (1) is $K$-wise linearly independent, from the $K$ shares we can solve the system of $K$ equations where $Z_m$ and $c_k$’s are the unknowns.

Remark on Low Bandwidth. During each refreshing stage, the number of integers communicated among the servers is $O(N^2)$, which is independent of $M$, the number of secrets. Thus, this scheme uses low bandwidth.

Remark on the Security. Since $W$ is of a size comparable to $n$, it is computationally infeasible to expand the polynomials to exploit the relationships among different shares. Therefore, we believe that the scheme is computationally secure.

4 Remarks and Future Work

Proof of Security Although we believe that the scheme is computationally secure, it remains unproven. It would be interesting to investigate the security of the scheme further.

Variations There are a number of variations of Chaining Scheme. Note that, to achieve feasible reconstruction, the current chain exponentiation can be replaced by many variants. For example, we can raise it to a lower power, which makes it easier for the servers to compute the refresh, or higher power, for better security.
Note that the sequence at which the new shares are computed is fixed for every refresh. Another variation uses a random sequence each time. This random sequence is then purged after refreshing. Thus no honest individual knows how the shares depend on each other. However, theoretically, the security of this variation does not improve because such random sequence is made known to every server.

5 Conclusions

In practice, proactive secret sharing schemes are difficult to implement. We identify two interesting conflicting requirements, low bandwidth and fine granularity, that are crucial in practical implementation.

We give a \((K, N)\) threshold proactive secret sharing scheme that achieves both requirements. Our main idea is to designate each server an identity, which is a number of low order in the multiplicative group \(\mathbb{Z}_n^*\). This choice ensures that polynomials remain low degree even after chain exponentiation. The use of identity with group structure is in contrast to Shamir’s scheme where each server is simply designated any number as its identity.

Since we can view the proposed scheme as an instant of Shamir’s scheme, many features, for e.g. verifiability, of known schemes based on Shamir’s scheme can be inherited.

References


