Distributed Protocols: Notes and Links

Data Protection
Master in Cyber Security (UPC) 2025
Jorge L. Villar

Last updated: Sep 1 13:22:10 2025

Contents

1 Distributing a Cryptographic Task

In some practical settings, the typical cryptographic abilities, like decrypting a ciphertext of generating a digital signature, cannot be performed by a single entity for security reasons. This situations typically occur when there is no entity that can be totally trusted.

The usual way to overcome this problem is sharing the secret key (associated with these cryptographic abilities) among several entities. The cryptographic task is then performed without reconstructing the entire secret key. Thus, the shared key can be used many times without any security degradation. For instance, in a distributed public-key encryption scheme the decryption key is shared among several users. Each user can only compute the partial decryption of a given ciphertext, but a minimum number of partial decryptions would be required to reconstruct the plaintext.

A similar construction can be used to jointly generate a digital signature: Every shareholder can generate a partial signature, but the valid digital signature can only be generated from several partial signatures. In addition, some degree of anonymity can be achieved in some constructions. For instance, from the generated signature, there is no way to know who in particular generated the necessary partial signatures.

The most important tool in the design of distributed cryptosystems is the so-called **secret sharing**.

2 Perfect Secret Sharing

In a **secret sharing scheme**, a secret s is shared among a set of participants $\{P_1, \ldots, P_n\}$, in a way that only certain subsets of participants can recover s. These subsets are known as **authorized subsets**, and the family of all of them is called the **access structure** of the secret sharing scheme.

An example of access structure is the (t, n)-threshold access structure, consisting of all subsets with al least t participants. This was precisely the first proposed access structure in the seminal independent works by Blackley and Shamir (1979).

All possible access structures fulfil the monotonicity property: Any subset of participants containing an authorized subset is also authorized. This property is trivially fulfilled by the (t, n)-threshold access structure.

The specification of a particular secret sharing scheme includes at least a **distribution** protocol and a **reconstruction** protocol. The distribution protocol is typically run by a trusted party, called **the dealer**, who computes the shares $s_1, \ldots, s_n$ corresponding to participants $P_1, \ldots, P_n$ from a given secret s and some public information, like the identity of the participants and the access structure of the scheme. Each share is given to the corresponding participant, who keeps it secretly.

The reconstruction protocol is run by a subset of participants. If the subset is authorized, the reconstruction protocol recovers the secret s from the participants' shares.

A secret sharing scheme is called **perfect** if

1. The shares of any authorized subset uniquely determine the value of the secret,
2. The shares of a non-authorized subset give no information about the secret.

The last property in particular means that if the secret was generated with the uniform probability distribution on a set S, then it remains with the same probability distribution when the shares of any non-authorized subset of participants are revealed. In the (t, n)-threshold access structure, perfectness implies that any subset of $t-1$ shares gives no information about the secret, but t shares are enough to determine it.

The security of a perfect secret sharing scheme is defined with respect to an adversary that can only corrupt the participants in a non-authorized subset (otherwise, the adversary will trivially recover the secret).

2.1 Shamir Secret Sharing Scheme

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of participants, and let t be an integer, called the threshold, such that $1 \leq t \leq n$. Shamir's secret sharing scheme is a (t, n)-threshold scheme, and then a subset A of participants is authorized to recover the secret if and only if A has at least t participants.

The set of possible values of the secret is a finite field. For simplicity, we will assume that the finite field is a prime field (i.e., it has prime order q). In any case, it is required that $q > n$. Initially, each participant P_i is assigned a nonzero and different field element x_i. The public description of the scheme includes the parameters n, t, q and the field elements $x_1, \ldots, x_n$.

In the sharing phase of the secret sharing scheme the dealer chooses (or he is given) a secret $s \in Z_q$. Then, he chooses a random polynomial r of degree at most $t-1$, with coefficients in Z_q, such that $r(0) = s$. Finally, the dealer securely sends the share $s_i = r(x_i)$ to P_i, for $i = 1, \ldots n$.

Since the sharing phase is critical for the security and correctness of the secret sharing scheme, the dealer is assumed to be a trusted (external) party.

The secret reconstruction phase requires the collaboration of t honest participants. The t participants use their shares to compute the unique polynomial r of degree at most $t-1$ that fulfils the equations $s_i = r(x_i)$, and then they recover the secret $s = r(0)$.

Clearly, Shamir's secret sharing scheme is correct (every authorized subset of honest participants recover the secret shared by an honest dealer), and it is also perfect (any non-authorized set of participants can get no information about the secret from the public information and their shares). Indeed, given any set of less

than t shares, there exist the same number of polynomials of degree at most $t-1$ fulfilling the equations $s_i = r(x_i)$, $i = 1, \ldots, n$, for every possible value of the secret $s \in Z_q$. Therefore, even knowing $t-1$ shares, the secret still is uniformly distributed in Z_q.

Example 1 *In Z_{17} we consider four participants P_1, P_2, P_3, P_4, and threshold $t = 3$. Let's assign the public values $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, $x_4 = 6$ to the participants.*

The secret $s = 2$ can be shared with the randomly generated polynomial $r(x) = 2x^2 - 4x + 2$. Then, the generated shares are $s_1 = 0$, $s_2 = 2$, $s_3 = 8$, $s_4 = 1$.

Participants P_1, P_2 and P_3 can recover the secret, since there exists a unique polynomial of degree at most 2 passing through the points $(1, 0)$, $(2, 2)$ and $(3, 8)$. However, P_1 and P_2 alone cannot do the same, since for every possible value of the secret s there exists exactly one polynomial of degree at most 2 passing through the points $(1, 0)$, $(2, 2)$ and $(0, s)$.

2.2 Access Structures

Threshold access structures are useful in many distributed cryptographic protocols, where there is no distinction among the different participants. The threshold value is related to the security of the scheme. Indeed, the higher the threshold is, the harder is for the adversary to learn enough shares from the attacked participants to be able to reconstruct the secret.

On the other side, if the threshold is too close to the number of participants (say $t = n-1$), then the adversary only needs to corrupt $n - t + 1$ participants to prevent any set of honest participants from recovering the secret. Therefore, to guarantee that the honest participants can recover the secret while the adversary cannot (even if it corrupts $t - 1$ users), it is required that $n - t + 1 > t$. This implies that there must be an honest majority of participants.

There are some applications in which not all participants can be considered as equivalent. In consequence, several families of non-threshold access structures have been proposed in the literature. For instance, in a **weighted threshold** access structure, every participant has an associated positive weight w_i, and a subset of participants is authorized if the sum of their weights is greater than or equal to a given threshold.

In other useful access structures, the set of participants is divided into disjoint classes. Participants in the same class are completely equivalent, and the access structure is defined in terms of the required number of participants in every specific class. For instance, a subset of participants is authorized if and only if it contains at least three participants in the first class and at least two of the second class.

Not all access structures can be implemented with the same efficiency. The efficiency of a secret sharing scheme is typically measured in terms of the size of the shares, compared to the size of the secret. The efficiency of an access structure is the efficiency of the best secret sharing scheme implementing it.

It can be shown that in a perfect secret sharing scheme, the size of the shares cannot be smaller than the size of the secret. Shamir's secret sharing scheme achieves this bound. Thus, it can be considered optimal. These optimal schemes are also known as **ideal** secret sharing schemes. An access structure is ideal if it admits an ideal secret sharing scheme implementing it. Therefore, theshold access structures are ideal.

2.3 Linear Secret Sharing Schemes

Linear secret sharing schemes are a generalization of Shamir's secret sharing scheme, that keeps most of its good properties. In a linear secret sharing scheme, every participant P_i has an associated vector $\mathbf{v}_i \in V$, where V is a finite dimensional vector space over a finite field (say Z_q, for a prime number q).

A subset of participants A is authorized if and only if the vector associated to the secret (say $\mathbf{v}_0 = (1, 0, \ldots, 0)$) is in the linear span of the vectors of the participants in the subset. Therefore,

$\mathbf{v}_0 = \sum_{i \in A} \lambda_i \mathbf{v}_i$, for suitable scalars $\lambda_i \in Z_q$.

The dealer chooses a random vector $\mathbf{r} \in V$ such that the dot product $\mathbf{r} \cdot \mathbf{v}_0$ equals the secret s. Then, it

computes the shares also as dot products: $s_i = \mathbf{r} \cdot \mathbf{v}_i$.

By linearity, we know that the same linear combination of the vectors also applies to the dot products, and then, for an authorized subset of participants,

$s = \sum_{i \in A} \lambda_i s_i$.

It can be shown that the scheme is perfect (mainly because the vector $\mathbf{v}_0$ is not in the span of the participants' vectors, for any non-authorized subset), and it is also ideal (since both the shares and the secret are field elements, and then they have the same size).

It is easy to see that Shamir's secret sharing scheme is a particular case of a linear secret sharing scheme. Indeed, it sufices to take the vectors $\mathbf{v}_i = (1, x_i, x_i^2, \ldots, x_i^{t-1})$. Computing the dot product $\mathbf{r} \cdot \mathbf{v}_i$ is actually the same as evaluating a polynomial, which coefficients are the coordinates of $\mathbf{r}$, at the point x_i.

In a (non-ideal) extension of the above scheme (still called linear) a participant can be associated to more than one vector, and it receives several dot products as its share.

2.4 Homomorphic Properties

Linear secret sharing schemes have some homomorphic properties (in the same way as happens with some public-key encryption schemes). Namely, if two secrets s, s' are shared with the same linear secret sharing scheme (i.e., with the same vectors associated to the participants), then the shares of the secret sum $s'' = s + s'$ are just the sum of the shares corresponding to the two secrets, $s_i'' = s_i + s_i'$. This means that the participants can compute the new shares locally (witout any interaction) without any extra action performed by the dealer.

There is actually a more general homomorphic property: the participants can locally compute the shares of any linear combination of already shared secrets (not only the sum), provided that the coefficients of the linear combination are publicly known (i.e., known by all participants). In particular, the shares corresponding to the secret $s'' = \lambda s + \mu s'$ are locally computed as $s_i'' = \lambda s_i + \mu s_i'$.

3 Computational Secret Sharing

Perfect secret sharing can be efficiently implemented for some specific access structure families (like the threshold one). In order to remove the efficiency constraints the notion of perfectness is relaxed to a computational setting. In **computational secret sharing**, a non-authorized subset cannot recover any additional information about the secret from their shares unless some computational problem (like factoring or computing discrete logarithms) is easy.

Relaxing the perfectness notion also allows to build efficient secret sharing schemes with additional features, like the verifiability of the dealer in the sharing phase, or of the participants' shares during the reconstruction phase.

3.1 Verifiable Secret Sharing

A **verifiable secret sharing** scheme is a secret sharing scheme with an additional dealer verification subprotocol. This new subprotocol is run by the participants, in such a way that, assuming that there are enough honest participants, if a (possibly dishonest) dealer will pass the verification, then all authorized subsets of honest participants will recover the same secret.

A verifiable secret sharing scheme can also provide some mechanism to detect the validity of individual shares during the reconstruction phase. In this case, they are the participants who are verified. Any dishonest participant trying to use a corrupted share will be detected by the other honest participants taking part in the reconstruction protocol, if there are enough honest participants among them.

3.2 Feldman Secret Sharing Scheme

Feldman's verifiable secret sharing scheme is a variant of Shamir's one that adds verifiability of the dealer, at the cost of relaxing the perfectness of the scheme to the computational setting. In addition, each individual share can also be verified during the reconstruction phase.

In addition to the normal setup of Shamir's secret sharing scheme, the usual description of a prime order group (G, q, g) is generated. Then, the shares s_i are computed as in Shamir's scheme with the random polynomial $r(x) = a_0 + a_1 x + \ldots + a_{t-1}x^{t-1}$, that is $s_i = r(x_i)$, and they are securely sent to the participants, while the group elements $A_j = g^{a_j}$ for $j = 0, \ldots, t-1$ are published.

Every participant P_i can verify the received share s_i by checking the equation

$g^{s_i} = A_0 A_1^{x_i} \cdots A_{t-1}^{x_i^{t-1}}$.

If the share does not fulfil the equation, the participant raises a complaint against the dealer. If t or more complaints are raised, then all (honest) participants abort the protocol and reject any further request to reconstruct the secret.

During the reconstruction phase, every participant can verify the shares received from other participants exactly in the same way. Thus, a dishonest participant will immediately be detected by his honest partners.

Honest participants will try to recover the secret only from the verified shares. Therefore, assuming that the adversary can corrupt up to $t-1$ participants, in any set of at least $2t-1$ participants there are at least t honest participants, and all of them will reconstruct the right secret.

Observe that the security of the scheme is only computational, since an adversary capable to solve the discrete logarithm problem can find the secret without the knowledge of any share, using only the public information. Indeed, the secret is just the discrete logarithm of A_0 with respect to g.

Notice that verifiability of Feldman's scheme is quite limited, because only the participants can verify the dealer. In a **publicly verifiable** scheme, any external party can verify the dealer using only the public information.

4 Distributed Public-Key Encryption

The good algebraic properties of some secret sharing scheme, like Shamir's scheme, make it possible to apply them to share the secret key of a public-key encryption scheme, in a way that the decryption operation can be performed without reconstructing the whole secret key. This last property helps protecting the secrecy of the scheme when several decryption operations are performed with the same long-term secret key.

Achieving security against a passive adversary that corrupts up to t out of n shareholders for $n > 2t$ is quite easy for public-key encryption schemes like ElGamal. However, it is far more difficult (resulting in a noticeable efficiency lose) to resist active attacks. For simplicity, we only consider below a threshold version of ElGamal that is secure against passive attacks. Also, for simplicity, we will assume that a trusted party generates the secret key and the corresponding shares. This trusted party is only necessary during the system setup, and it takes no part in the encryption or decription procedures.

The dealer executes the sharing phase of Feldman's secret sharing scheme for a (t, n)-threshold access structure, where the participants are the shareholders that will later execute the decryption protocol. After the sharing phase, each participant P_i knows a partial secret key x_i (not to be confused with the public nonzero field element ξ_i used in the underlying Shamir secret sharing scheme). The shares implicitly define a polynomial $P(\xi) = a_0 + a_1\xi + \ldots + a_{t-1}\xi^{t-1}$ of degree at most $t-1$, such that $x_i = P(\xi_i)$. Also, the group elements $A_j = g^{a_j}$ are published. The public key of the encryption scheme is $pk = y = A_0 = g^{P(0)}$ and the corresponding secret key is $sk = x = P(0)$ (only known to the dealer).

The encryption operation is exactly the same as in plain ElGamal encryption scheme, but the decryption operation is now distributed among the participants holding the partial secret keys. Given an ElGamal encryption $(c_0, c_1) = (g^r, my^r)$ every P_i locally computes the partial decryption $z_i = c_0^{x_i} = g^{rP(\xi_i)}$, using his partial secret key, and sends it to the other participants. From t partial decryptions it is easy to recover the

plaintext. Indeed, using Shamir's reconstruction equations, one can compute coefficients λ_i such that $x = \sum_{i \in A} \lambda_i x_i$, where A is the set of participants pooling their partial decryptions.

Thus, using the same coefficients in the exponent, we have
$y^r = g^{rx} = \prod_{i \in A} g^{r\lambda_i x_i} = \prod_{i \in A} z_i^{\lambda_i}$.

Once y^r is recovered, the plaintext is computed as $m = c_1 (y^r)^{-1}$.

Most parts of the public output of Feldman's secret sharing used in the key generation procedure are not used in the decryption protocol, but they can be used to verify the correctness of the individual partial decryptions, by using a non-interactive zero-knowledge proof that the discrete logaritms of z_i with respect to c_0 and of $y_i = g^{x_i} = A_0 A_1^{\xi_i} \cdots A_{t-1}^{\xi_i^{t-1}}$ with respect to g are equal.

5 Distributed Signatures

Some digital signature schemes can also be distributed in a similar way. For instance, a (n, n)-threshold version of the RSA-FDH signature scheme is described below.

We assume that the RSA modulus N and the public exponent e are generated by a trusted party, and it splits the secret exponent d as the sum of n integers $d = d_1 + \ldots + d_n$, where each integer is a random number in the interval $[-2^k N, 2^k N]$, for a large enough k (say $k = 160$). The integer d_i is the share of participant (signer) P_i.

Given a message m, the corresponding RSA-FDH signature, using a hash function H, is
$\sigma = H(m)^d = H(m)^{d_1 + \cdots + d_n} = H(m)^{d_1} \cdots H(m)^{d_n} \bmod n$.

Thus, each participant can compute a partial signature $\sigma_i = H(m)^{d_i} \bmod n$ and then the final signature is computed as the product of all n partial signatures.

More general (for smaller thresholds) and more robust distributed signature schemes exist, but the constructions are far more technical that the above simple example, mainly because in the RSA setting the modulus used in the exponents, $\phi(n)$, is not publicly known, and then, polynomial interpolation or other related techniques cannot be directly applied.