14. Secret Sharing

Sometimes it is necessary to store a secret in places that are not fully secure, e.g., we think it might be possible that a single place might be revealed, but it is very unlikely that all of them are revealed at the same time. This case can be effectively treated by using a secret sharing scheme to distribute the secret to several places. Secret sharing schemes prevent an attacker from learning any information if he revealed a single location, but still enables us to reconstruct the secret from a sufficiently large number of shares.

Note that this cannot be achieved in a trivial way by storing individual letters of the password in different places: On the one hand, the strong secrecy requirement would be violated even if an adversary found only one share. On the other hand, even the requirement that any sufficiently large number of shares enables us to reconstruct the secret would not be easy to fulfill because some shares would need to overlap in some letters.

14.1 Definition Sketch of Basic Secret Sharing

The following is called a sketch because we leave out the precise formalization of notions like protocol and participant. Nevertheless many cryptographers would not hesitate to call it a definition.

As in all following definitions, we will define parameters of a protocol (also called scheme), roles, i.e., different types of participants, subprotocols (also called phases or transactions or protocols again), and requirements with corresponding trust models.

The parameters are the total number of desired shares n, the minimum number k of shares from which reconstruction is possible, and a secret space S, i.e., a set of possible secrets. An instantiation with specific n and k is often called a k-of-n secret sharing scheme.

There are three roles: a so-called *dealer*, i.e., the initial owner of the secret, *shareholders*, i.e., participants who get shares, and a *reconstructor*, i.e., a party that reconstructs the secret. In a particular execution of secret sharing, one dealer, n shareholders, and one reconstructor will take part.

By the initial example, one may not see a need for shareholders and reconstructors — the shares could be in "places" instead of held by "participants", and the reconstructor is typically the dealer. However, to include more examples and some extensions, it is better to make this slightly more general definition at once.

There are two subprotocols:

- *Sharing.* This is a protocol between the dealer and all shareholders. Typically it is a very simple type of protocol with almost no interaction: The dealer has an input $s \in S$, i.e., the secret to be shared. The dealer makes some local computation given by the algorithm share (typically probabilistic) resulting in so-called *shares* $s_1, \ldots, s_n$. One share s_i is sent to each shareholder, who simply stores s_i.

- *Reconstruction.* This is a protocol between the reconstructor and k shareholders. Typically it is also a very simple type of protocol with almost no interaction: Each participating shareholder sends its share to the reconstructor. The reconstructor makes some local computation given by the algorithm $\mathsf{reconstruct}$ resulting in a value s^*. (This is supposed to be the secret s again, but it is better to use different notation and formally require equality in our availability requirement.) We will call the set of indices of shares used for reconstruction *Avail*. For instance, if s_2, s_3, and s_5 are used, we have $Avail = \{2, 3, 5\}$. More generally, this is the set of indices (or names) of the shareholders participating in reconstruction.

There are two requirements.

- *Availability*: Reconstruction works correctly, i.e., $s^* = s$, if the dealer, the reconstructor, and the k shareholders participating in this particular reconstruction are honest, and the adversary cannot modify any message.
- *Confidentiality*: Any $k-1$ shareholders gain no information about the secret s if the dealer and the reconstructor are honest, and the lines between honest participants are not tapped.

If several reconstructions of the same secret may be needed (e.g., the password is needed every three months), this requirement includes that shareholders who already participated in some reconstructions still do not learn the secret. In the non-interactive case as above it is clear that shareholders do not gain any information in reconstruction, but in some extensions this needs more care.

We can already write the two requirements more formally for the non-interactive case. For this, let share and $\mathsf{reconstruct}$ be the algorithms that the dealer and the reconstructor use. The former must be probabilistic, the latter is deterministic. We can assume that $\mathsf{reconstruct}$ does not only get the shares themselves as inputs, but also the indices of the participants who held them.

- *Availability*: For all $s \in S$, for all n-tuples $(s_1, \ldots, s_n) \in [\mathsf{share}(s)]$ (i.e., all possible outputs of $\mathsf{share}(s)$), and all k-tuples $(i_1, \ldots, i_k)$ where all i_j are distinct elements of $\{1, \ldots, n\}$, we have $\mathsf{reconstruct}(s_{i_1}, \ldots, s_{i_k}) = s$.
- *Confidentiality*: This is defined as follows: For all probability distributions D on the secret space, all specific secrets s, and all observations c of the attacker, the a-posteriori probability of s, given the observation c, equals the a-priori probability of s. The probability space in which we can speak of all these probabilities is given by the initial distribution D on the secret space and (the random bits used by) the probabilistic algorithm share: All shares are random variables in this space, and the observation c consists of the shares an attacker has. Thus, if Pr_{D^*} is the probability in this space, we can simply write for any $i_1, \ldots, i_{k-1} \in \{1, \ldots, n\}$ (defining the attacker)
$$\mathsf{Pr}_{D^*}(s \mid (s_{i_1}, \ldots, s_{i_{k-1}})) = \mathsf{Pr}_{D^*}.$$

14.2 Remarks on the Trust

First note that we have different trust models for availability and confidentiality. Recall that one generally speaks of a k-of-n trust model if one trusts that at most $k-1$ out of n given participants are dishonest.

- For confidentiality, we have exactly such a k-of-n model for the shareholders: If at most $k-1$ of them are dishonest, the dishonest ones have no information.

- For availability, however, the trust is different: We assume that at least k shareholders are honest. This means that at most $n-k$ are dishonest, which corresponds to an $(n-k+1)$-of-n model.

Note that there is *no integrity* at all. In particular, if one of the shareholders participating in a reconstruction contributes a wrong share, a wrong secret will be reconstructed.

Furthermore, to see that considering the definition and in particular the underlying trust model can be helpful in practice, consider an example that is a rather usual introduction to secret sharing: Consider a bank with a highly secret vault that no single employee is supposed to open, but only, say, 3 of 5 directors together. The vault is opened with a secret number, and this number is shared among the directors. This sounds ok at first, but if you look closely at the trust, there are problems: Who should be the trusted dealer and reconstructor? In particular, reconstruction is needed quite often. If the vault has a mechanical combination lock, i.e., the directors themselves must reconstruct the secret first and then input it, at least one of them will know it. On the other hand, if the vault is computerized and can serve as the reconstructor itself, there is no need for secret sharing: It would be much simpler if each of the directors got an independent password and the vault counted how many correct passwords had been entered. Thus basic secret sharing is in fact much better suited to the sharing of a personal password or a personal secret key, where the owner is both reconstructor and dealer, and where the secret is given a priori from another system.

14.3 Shamir's Construction

Secret sharing was invented independently by Shamir and Blakley in 1979, but Shamir's construction is simpler and better known nowadays. We first describe conditions on the parameters and then the two protocols, sharing and reconstruction. Both are of the simple, almost non-interactive structure described above.

14.3.1 Parameter Conditions

The secret space S must be a subset of a finite field $\mathbb{F}$, e.g., a prime field $\mathbb{Z}_p$. Moreover, $|\mathbb{F}| > n$ is needed.

14.3.2 Sharing

1. First, the dealer chooses a random polynomial pol of degree $k-1$ over $\mathbb{F}$ with constant term s. In other words, he chooses coefficients $a_1, \ldots, a_{k-1} \leftarrow_{\mathcal{R}} \mathbb{F}$ and sets

   $$pol(x) = a_{k-1}x^{k-1} + \ldots + a_1 x_1 + s,$$

   where s is the secret.

2. Second, the dealer fixes a different value $x_i \in \mathbb{F}^*$ (i.e., non-zero field elements) for each shareholder and gives the pair

   $$s_i := (x_i, pol(x_i))$$

   to this shareholder as her share. Thus each pair is a point on the polynomial.

   If the shareholders are a priori numbered, we can also assume a fixed mapping to values x_i. Then only the value $pol(x_i)$ is the actual share. For example, if $\mathbb{F} = \mathbb{Z}_p$, shareholder i can

simply have $x_i = i$. The precondition $|\mathbb{F}| > n$ is needed so that each shareholder can have a different value x_i, and $x_i \neq 0$ is needed because $pol(0) = s$, i.e., a share at the point 0 would be the secret itself.

14.3.3 Reconstruction

Before presenting the actual reconstruction algorithm, we can see that the secret can in principle be reconstructed uniquely from any k shares: It is a well-known theorem in algebra that over any field, there is at most one polynomial of degree $k-1$ through any k given points. (One can easily derive this as a corollary from an even better-known theorem that any non-zero polynomial of degree $k-1$ has at most $k-1$ zeros.) E.g., there is at most one straight line through any two points, and at most one parabola through any three points. Now the k shareholders have k points, thus their points determine a polynomial uniquely, and thus also the secret as the constant coefficient of this polynomial. An efficient algorithm for actually constructing the polynomial from the given points is Lagrange interpolation.

First one computes "basis polynomials" $bpol_{i^*}(x)$ which have value 1 at one point x_{i^*}, and value 0 at all the other points under consideration.

To do this, one starts with even simpler polynomials

$$cpol_{i^*}(x) := \prod_{i \in Avail \setminus \{i^*\}} (x - x_i).$$

Recall that *Avail* is the set of indices of shares used for reconstruction. Obviously, $cpol_{i^*}(x_i) = 0$ for all $i \in Avail \setminus \{i^*\}$ already, but the value at x_{i^*} is typically not yet 1 (but different from 0!). Thus we normalize these polynomials by dividing them through their value at the point x_{i^*} and obtain

$$bpol_{i^*}(x) := \frac{cpol_{i^*}(x)}{cpol_{i^*}(x_{i^*})} = \prod_{i \in Avail \setminus \{i^*\}} \frac{(x - x_i)}{(x_{i^*} - x_i)}$$

Now we simply use the appropriate linear combination of basis polynomials to obtain a polynomial *pol* through the points (x_i, y_i) for all $i \in Avail$:

$$pol(x) = \sum_{i \in Avail} y_i \cdot bpol_i(x).$$

14.3.4 Proof of Availability

Availability is the fact that reconstruction works correctly, and this was almost proven in the last section. Section However, we can still check it in the final formula: For all $i^* \in Avail$ we have

$$\begin{aligned} pol(x_{i^*}) &= \sum_{i \in Avail} y_i \cdot bpol_i(x_{i^*}) \\ &= 0 + \ldots + 0 + y_{i^*} \cdot bpol_{i^*}(x_{i^*}) + 0 + \ldots + 0 \\ &= y_{i^*}. \end{aligned}$$

Thus Lagrange interpolation gives in fact a polynomial through all the given points, and by uniqueness this is the polynomial the dealer had chosen.

14.3.5 Proof of Secrecy

We prove secrecy by the following lemma. Note that here we assumed that any attacker has in fact $k-1$ shares and not less; it is clear that any weaker attacker cannot have more information.

Lemma 14.1 *For any attacker indices $i_1, \ldots, i_{k-1} \in \{1, \ldots, n\}$, for all possible observations $c = (s_{i_1}, \ldots, s_{i_{k-1}})$, where each share s_{i_k} is a point (x_{i_j}, y_{i_j}), and for every secret $s \in S$, there exists exactly one polynomial pol that is consistent with both the observation and the secret.* □

Proof. The secret and the shares together fix k points on the polynomial (recall that the secret was the point at $x = 0$). By the algebraic theorem, there is at most one such polynomial, and by Lagrange interpolation, there is also at least one. ■

14.4 Efficiency Improvements and Further Properties

Now we consider some improvements and interesting facts about Shamir's scheme.

14.4.1 Simple Improvements

Not reconstructing the entire polynomial As we only need the constant coefficient of the polynomial at the end, it is not necessary to reconstruct all the others. Thus we actually only need to compute

$$s = pol(0) = \sum_{i \in Avail} y_i \cdot bpol_i(0)$$

where

$$bpol_{i^*}(0) = \prod i \in Avail \setminus \{i^*\} \frac{-x_i}{x_{i^*} - x_i}$$

Splitting long secrets For sharing large secrets exactly as above, one would have to compute in very large fields, and the complexity of multiplication is quadratic. However, the complexity becomes linear if we simply split long secrets into fixed-length blocks and share those independently.

Sharing several secrets For secrecy, one clearly needs a completely new polynomial for each secret to be shared. Nevertheless, there is a lot of work one only needs to do once, at least if the same subset of participants reconstructs several secrets: The basis polynomials depend only on the subset and values x_i assigned to the participants, which can be fixed. Thus this subset can compute the value $bpol_i(0)$ for each of its members once, and reconstruction is a simple linear combination with the current values y_i.

14.4.2 Optimality

As you may have noticed, each share is as long as the original secret. This may not correspond to an intuitive idea of "sharing". However, one can quite easily see that one cannot do better. The proof sketch goes as follows: With any $k-1$ shares, one should still have no information at all, in an information-theoretic sense, about the secret. However, with k shares, one suddenly knows the secret exactly, i.e., has the full information about it. Thus as much information as in the secret must

have been in the k-th share. If one knows elementary information theory, one can formalize this in a natural way. Here we prove a slightly weaker version of the optimality theorem in an elementary way: We show that each share must have at least as many possible values as the secret. Thus we show that

$$|S_i| \geq |S|,$$

where S_i is the set from which the i-th share is chosen, and S was the secret space.

Proof. (Sketch) First consider an attacker who has $k-1$ shares, not including share i, i.e., the attacker indices are $i_1, \ldots, i_{k-1} \in \{1, \ldots, n\} \setminus \{i\}$. From the point of view of this attacker, all secrets $s \in S$ must still be possible by the information-theoretic confidentiality. Thus for each s, the probabilistic algorithm $\mathsf{share}(s)$ must have an output in which all the attacker's shares occur. Let $s_i^{(s)}$ be share i in this same output. (I.e., it is a value for the i-th share that is consistent with both the secret s and the observation c.) Now we consider reconstruction by the k participants i and $i_1, \ldots, i_{k-1}$. By the availability requirement, $\mathsf{reconstruct}((i, s_i^{(s)}), (i_1, s_{i_1}), \ldots, (i_{k-1}, s_{i_{k-1}}))$ must be s. This implies that all the values $s_i^{(s)}$ for different s must be different: If $s_i^{(s)}$ were equal to $s_i^{(s*)}$ for $s \neq s^*$, then $\mathsf{reconstruct}$ would have two different values s, s^* as output on the same input tuple. But reconstruct is a deterministic function. Hence there are at least as many possible values in S_i as in S. ∎

14.4.3 Computing with Secrets

Sometimes we will want to compute with secrets that have been shared, and it is dangerous to put the secret together. In particular, this holds in group cryptography. For the moment, we will just consider the simplest case, which is to compute linear combinations of shared secrets. This is equivalent to being able to compute additions and scalar multiplications, i.e., multiplications with a value that is not hidden as a shared secret, but known to all participants. Assume that two secrets s and s' are shared using polynomials pol and pol'. Thus the secrets are

$$s = pol(0) \text{ and } s' = pol'(0),$$

and the shares of a participant with fixed value x_i are

$$y_i = pol(x_i) \text{ and } y_i' = pol'(x_i).$$

Now consider the sum pol'' of the polynomials, which is also a polynomial of degree at most $k-1$: We have

$$pol''(0) = pol(0) + pol'(0) = s + s'.$$

Hence this polynomial pol'' can be used to share the sum of the secrets. Moreover,

$$pol''(x_i) = pol(x_i) + pol'(x_i) = y_i + y_i'.$$

Hence the shares of this polynomial pol'' are the sums of the shares of the original polynomials. This last operation can be carried out by each shareholder locally: He just adds his two shares together. Thus they now hold shares to the sum $s + s'$ of the original two secrets. Similarly, one obtains the shares of a value ks, where $k \in \mathbb{F}$ is a known value and s is a shared secret, if each shareholder locally multiplies her share by k.

14.5 Sketch of Verifiable Secret Sharing

As the name says, verifiable secret sharing (abbreviated VSS) is an extension of secret sharing. The point is to add more robustness (availability) to the original secret sharing. This has two aspects:

- *Malicious shareholders*: So far we assumed that shares are either correct or get lost completely. In other words, each share that is used in a reconstruction is correct. However, a malicious shareholder might input a wrong share, and then a wrong secret would be reconstructed. (One can easily see that Shamir's scheme has no redundancy at all if exactly k shares are used for reconstruction, i.e., whatever one inputs, there is always some result.) Depending on the situation, one might notice that the reconstructed secret is incorrect (e.g., it is not a secret key that corresponds to a certain public key), and try other subsets of shareholders. But this procedure is cumbersome. Hence it would be nicer if one could verify each share individually.

- *Untrusted dealer*: The more complicated problem is if the person who originally holds the secret is not trusted by the others, and the others need a guarantee that they can put a consistent secret together. An example is that a person with an important job in a company has to share her private key with some others in case she dies or leaves the company in a quarrel. Now the others will want a guarantee that they hold shares of a private key that correctly fits the known public key. In other examples it is enough that any secret is consistently fixed in the shares. (For instance, in a multi-party coin flipping protocol the individual random values r_i might not be fixed by commitments, but by sharing them with a verifiable secret sharing system to avoid the disruption problem at the cost of a different trust model.)

We now present Pedersen's verifiable secret sharing scheme.We omit a few details about parameters, the choice of the group etc.

1. Basically the scheme uses Shamir's secret sharing to share the secret s. Thus the dealer chooses a polynomial
$$pol(x) = a_{k-1}x^{k-1} + \ldots + a_1 x^1 + s,$$
where the constant coefficient is the secret and the other coefficients are random. We assume that the x-values x_i of the shares for each participant P_i are fixed in advance, e.g., $x_i = i$.

2. Now, to fix everything, the dealer makes commitments to the whole polynomial, i.e., to all its coefficients. The discrete-logarithm commitments from Section 13.4.1 are used. Let us denote the commitments as follows:
$$\begin{aligned} com &= (com_{k-1}, \ldots, com_0) \\ &= (g^{a_{k-1}}h^{b_{k-1}}, \ldots, g^{a_0}h^{b_0}). \end{aligned}$$
Here, for simplicity, we also used the notation $a_0 := s$. The b_i's are all random. The dealer sends these commitments to everyone, i.e., the shareholders and the reconstructor. This should be done by broadcast, so that everybody agrees on these commitments.

   Note that so far, we have already made sure that there is really a polynomial of degree k, and that the secret is the content of com_0. Hence we can use the scheme for situations where an arbitrary but fixed secret is needed, and for situations where an external secret is given by such a commitment $g^s h^{b_0}$.

3. The dealer gives a normal Shamir share to each shareholder P_i, i.e.,

$$y_i = pol(x_i).$$

In addition (to help the verification in the next step), the dealer gives P_i a corresponding "helpshare" on the polynomial pol' defined by the values b_i. Thus, let

$$pol'(x) = b_{k-1}x^{k-1} + \ldots + b_1x^1 + b_0$$

and

$$z_i = pol'(x_i).$$

4. Now each shareholder verifies that his share s_i does in fact lie on the polynomial that is hidden in the published commitments. For this, we use the homomorphic property of the commitment scheme, i.e., the fact that we can perform additive operations on hidden values by performing the corresponding multiplicative operation on the commitments. If y_i and z_i are correct, we have

$$\begin{aligned} g^{y_i}h^{z_i} &= g^{pol(x_i)}h^{pol'(x_i)} \\ &= g^{a_{k-1}x_i^{k-1}+\ldots+a_1x_i^1+a_0} \cdot h^{b_{k-1}x_i^{k-1}+\ldots+b_1x_i^1+b_0} \\ &= (g^{a_{k-1}}h^{b_{k-1}})^{x_i^{k-1}} \cdot \ldots \cdot (g^{a_1}h^{b_1})^{x_i^1} \cdot (g^{a_0}h^{b_0}) \\ &= com_{k-1}^{x_i^{k-1}} \cdot \ldots \cdot com_1^{x_i^1} \cdot com_0 \\ &=: comshare_i. \end{aligned}$$

The last formula only contains publicly known values. Hence everybody can compute $comshare_i$. Thus P_i can simply verify whether the values y_i and z_i that the dealer gave him fulfil the equation $g^{y_i}h^{z_i} = comshare_i$.

5. During reconstruction, the reconstructor collects both the shares y_i and the help-shares z_i from k shareholders. He can also verify whether $g^{y_i}h^{z_i} = comshare_i$. If this is true, he knows that these shareholders gave him their correct shares, and he uses normal Shamir reconstruction on the values y_i to obtain the secret s.

In the full protocol, one must also specify what happens if some shareholders complain in Step 4 that their shares don't lie on the polynomial. (Either these shareholders or the dealer might be incorrect.) It should be clear from the explanations within the protocol, in particular in Step 4, that the scheme works in the correct case. Secrecy and availability should be plausible: All the commitments hide their contents information-theoretically, so they should not hurt secrecy. And the computational binding property of the commitments should guarantee that all the commitments have, for practical purposes, unique contents, so that in particular no shareholder can later open a commitment in different ways and thus define different polynomials, even if he colludes with a dishonest dealer. But if we had more time, a real proof should be given in this place.