Secret Sharing

Daniel Kales

Graz University of Technology

1 Introduction

Trusting a single entity with an important piece of information is something that carries a large amount of potential risks and often requires a large amount of trust placed in this entity. Historically, one can think of examples in the context of military operations, where a relatively extreme example comes in the form of nuclear launch codes. Other examples can be found in companies where, for example, the president has the sole power to authorize big business decisions such as spending large amounts of money. Due to the critical nature of some of these examples, one could look for a different solution and arrive at the following question.

> *Could we share a piece of information, such as authorization codes, between several parties, so that the individual parties learn nothing about it, but can work together to recover this information?*

In the following, we will present an approach known as *secret sharing*, and different instantiations of it. Using this approach, we can not only answer the aforementioned question in the positive, but can even implement complex access structures where only a subset of the individual people is required to recover the information.

2 Secret Sharing Schemes

We can think of a secret sharing scheme as a tuple of the following algorithms: $(\mathsf{Gen}, \mathsf{Share}, \mathsf{Reconstruct})$. The general definition of these algorithms is given in Scheme 1.

We can see one important feature of a secret sharing scheme in this definition, the concept of an access threshold t. In general, we split a secret into n shares, but for some use-cases we might want to be able to recover the secret x by combining only t (with $t \leq n$) shares. Such schemes are also called t-out-of-n secret sharing schemes.

In the following, we will look at two different, popular schemes for the specific cases of $t = n$ and $t \leq n$.

$\underline{\mathsf{Gen}()}$: Set up and return public parameters pp.

$\underline{\mathsf{Share}(\mathsf{pp}, x, n, t)}$: Share the secret x between n players, returning a set of n shares $\{x_1, \ldots, x_n\}$. Later, t of these shares can be used to reconstruct the secret x by calling $\mathsf{Reconstruct}$.

$\underline{\mathsf{Reconstruct}(\mathsf{pp}, \{x_1, \ldots, x_t\})}$: Recover the secret x from the t shares $x_1, \ldots, x_t$ and return x.

Scheme 1: Generalized Interface of a Secret Sharing Scheme.

The trivial case of $t = 1$. One can also think of a secret sharing scheme where $t = 1$, i.e., only one share suffices to recover the secret information x. However, for this case it simply suffices to give the original secret to each of the n parties.

2.1 Notation

In the following, we denote by $[n]$ the set $\{1, \ldots, n\}$ and by $x \overset{\$}{\leftarrow} S$ the sampling of an element x from the set S uniformly at random. Furthermore, for a given variable x, we denote by $\langle x \rangle$ the shared form of x, which refers to the set of the shares of x. For operations involving shared values, this means applying the operation to each share of x individually (e.g., $\langle z \rangle = \langle x \rangle + \langle y \rangle$ means adding the share x_i to the share y_i and calling the result z_i for all $i \in [n]$.). The same holds for multiplications with public values (e.g., $\langle z \rangle = 3 \cdot \langle x \rangle$ means multiplying the share x_i by 3 and calling the result z_i for all $i \in [n]$.), whereas additions with public values are only added to the first share and not to all shares (e.g., $\langle z \rangle = \langle x \rangle + 3$ means adding 3 to the share x_1 only).

2.2 Additive Secret Sharing

For the access threshold of $t = n$, we require every single share to recover the original secret. Although this is less general than allowing an arbitrary threshold $t \leq n$, it is still a popular use case. Furthermore, due to the restrictions placed on t, we can come up with very efficient schemes.

The idea behind additive secret sharing is contained in the name: We want to split a secret into n shares, so that the sum of all shares is equal to the original secret x.

Addition of Secret Shared Values Using additive secret sharing immediately results in a nice property. For any additively secret shared values $\langle x \rangle$ and $\langle y \rangle$, if we want to calculate $z = x + y$, we can perform this operation using only the secret shares.

$\underline{\mathsf{Gen}()}$: Agree on a group $\mathbb{G}$ with group operation $+$ and its inverse operation $-$ and set $\mathsf{pp} \leftarrow \mathbb{G}$. Return pp.

$\underline{\mathsf{Share}(\mathsf{pp}, x, n, t)}$: Parse pp as $\mathbb{G}$. If $n \neq t$ or $x \notin \mathbb{G}$, return $\perp$. Otherwise, for $i \in \{1, \ldots, n-1\}$ set $x_i \xleftarrow{\$} \mathbb{G}$. Set $x_n = x - \sum_{i=1}^{n-1} x_i$. Return $\{x_1, \ldots, x_n\}$.

$\underline{\mathsf{Reconstruct}(\mathsf{pp}, \{x_1, \ldots, x_t\})}$: For all $i \in [t]$, if there exists an x_i such that $x_i \notin \mathbb{G}$, return $\perp$. Otherwise, return $x = \sum_{i=1}^{t} x_i$.

Scheme 2: Additive Secret Sharing.

Theorem 1. *Let $\mathbb{G}$ be a group and $x, y \in \mathbb{G}, n \in \mathbb{N}$. Let $\langle x \rangle = \mathsf{Share}(\mathbb{G}, x, n, n)$ and $\langle y \rangle = \mathsf{Share}(\mathbb{G}, y, n, n)$ be the additive secret sharing of x and y respectively. Set $\langle z \rangle = \langle x \rangle + \langle y \rangle$. Then, $z = \mathsf{Reconstruct}(\mathbb{G}, \langle z \rangle) = x + y$.*

Proof. From the definition of additive secret sharing in Scheme 2, we get $x = \sum_{i=1}^{n} x_i$, $y = \sum_{i=1}^{n} y_i$ and $z = \sum_{i=1}^{n} z_i$. If we set $z_i = x_i + y_i$ according to the definition above, we get

$$z = \sum_{i=1}^{n} z_i = \sum_{i=1}^{n} (x_i + y_i) = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i = x + y \,.$$

$\square$

Multiplication of Secret Shared Values Instead of using a group $\mathbb{G}$, we could also consider instantiating the additive secret sharing scheme over a field $\mathbb{F}$, where, in addition to the additive operation $+$, we also have a multiplicative operation $\cdot$. Sadly, even though the addition of shared values still works as described above, this does not apply to the multiplication of shares. If one simply tries to multiply the individual shares with each other, we arrive at a wrong result:

$$z = \sum_{i=1}^{n} z_i = \sum_{i=1}^{n} (x_i \cdot y_i) \neq \sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} y_i = x \cdot y \,.$$

One approach to solve this issue was proposed by Donald Beaver [1], and is employing so called multiplication triples (also often called Beaver triples).

Definition 1. *Let $\mathbb{F}$ be a field and $a, b, c \in F$. We call a tuple of secret shared values $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ a multiplication triple if $c = a \cdot b$.*

Theorem 2. *Let $\mathbb{F}$ be a field and $x, y \in \mathbb{F}, n \in \mathbb{N}$. Let $\langle x \rangle = \mathsf{Share}(\mathbb{F}, x, n, n)$ and $\langle y \rangle = \mathsf{Share}(\mathbb{F}, y, n, n)$ be the additive secret sharing of x and y respectively. Let $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$ be a multiplication triple. Set $\langle d \rangle = \langle x \rangle -$*

$\langle a \rangle$ and $\langle e \rangle = \langle y \rangle - \langle b \rangle$. Broadcast $d = \mathsf{Reconstruct}(\mathbb{F}, \langle d \rangle)$ and $e = \mathsf{Reconstruct}(\mathbb{F}, \langle e \rangle)$. Set $\langle z \rangle = d \cdot e + d \cdot \langle b \rangle + e \cdot \langle a \rangle + \langle c \rangle$. Then, $z = \mathsf{Reconstruct}(\mathbb{F}, \langle z \rangle) = x \cdot y$.

Proof. *From the definition of additive secret sharing in Scheme 2, we get $x = \sum_{i=1}^n x_i$, $y = \sum_{i=1}^n y_i$ and $z = \sum_{i=1}^n z_i$. If we set $z_i = db_i + ea_i + c_i$ according to the definition above (while adding the public value de only the first share z_1), we get*

$$\begin{aligned} z &= \sum_{i=1}^n z_i = de + \sum_{i=1}^n (db_i + ea_i + c_i) = de + d\sum_{i=1}^n b_i + e\sum_{i=1}^n a_i + \sum_{i=1}^n c_i \\ &= de + db + ea + c = (x-a)(y-b) + (x-a)b + (y-b)a + c \\ &= xy - xb - ya + ab + xb - ab + ya - ba + ab = xy\,. \end{aligned}$$

□

This method is one of the standard methods for computing multiplications of secret shared values. However, it leaves the secure generation of multiplication triples to a preprocessing step. For security of this multiplication protocol it is important that the plain values of a, b and c are not known to any party, otherwise they could recover the values of x and y. Protocols for secure generation of these triples in environments with malicious participants are non-trivial and often involve advanced cryptographic primitives such as oblivious transfers (e.g., MASCOT [4]) or homomorphic encryption (e.g., SPDZ [3]).

2.3 Shamir Secret Sharing

A different approach for a secret sharing scheme was proposed by Adi Shamir in 1979 [5]. The scheme is based on polynomial interpolation in the plane of dimension 2. Given t points $(x_1, y_1), \ldots, (x_t, y_t)$ in this 2-dimensional plane, there exists a unique polynomial p of degree $t-1$ so that $p(x_i) = y_i \forall i \in [t]$. To create a threshold secret sharing scheme with threshold t, Shamir proposed to embed the secret x as the coefficient a_0 in the polynomial $p(X) = a_0 + a_1 X + \cdots + a_{t-1} X^{t-1}$ of degree $t-1$, where all other coefficients a_i are chosen at random. To generate n shares, the polynomial is evaluated over the domain $[n]$, i.e., the shares are $(1, p(1)), (2, p(2)), \ldots, (n, p(n))$. In the following, we assume the x-coordinate is equal to the party's index and will therefore always omit it. Given a subset of k of these points, we can find the original polynomial

p by means of polynomial interpolation (a method to perform this interpolation would be Lagrange interpolation). The exact procedure is given in Scheme 3.

```
Gen(): Agree on a finite field $\mathbb{F}$ and set $\mathsf{pp} \leftarrow \mathbb{F}$. Return $\mathsf{pp}$.
Share($\mathsf{pp}, x, n, t$): Parse $\mathsf{pp}$ as $\mathbb{F}$. If $n < t$ or $x \notin \mathbb{F}$ or $|\mathbb{F}| \leq n$, return $\perp$. Otherwise, for
    $i \in \{1, \ldots, t-1\}$ set $a_i \overset{\$}{\leftarrow} \mathbb{F}$ and $a_0 \leftarrow x$. Define the polynomial $p(X) = \sum_{i=0}^{t-1} a_i X^i$.
    For $i \in \{1, \ldots, n\}$, set $x_i \leftarrow p(i)$. Return $\{x_1, \ldots, x_n\}$.
Reconstruct($\mathsf{pp}, \{x_1, \ldots, x_t\}$): For all $i \in [t]$, if there exists an $x_i$ such that $x_i \notin \mathbb{F}$,
    return $\perp$. Otherwise, find the unique polynomial $p(X)$ interpolating the points
    $x_1, \ldots, x_t$. Return $x = p(0)$.
```

Scheme 3: Shamir Secret Sharing [5].

Addition of Secret Shared Values Similar to the case of additive secret sharing, Shamir secret sharing also allows us to perform additions of two secret shared values locally by adding the y-coordinates of the individual shares (we can only add shares with the same x-coordinate, so only shares for the same party). We show this in the following for the case of $t = n$, but this can easily be generalized to the case $t \leq n$.

Theorem 3. *Let $\mathbb{F}$ be a finite field and $x, y \in \mathbb{F}, n \in \mathbb{N}$. Let $\langle x \rangle = \mathsf{Share}(\mathbb{F}, x, n, n)$ and $\langle y \rangle = \mathsf{Share}(\mathbb{F}, y, n, n)$ be the Shamir secret sharing of x and y respectively. Set $\langle z \rangle = \langle x \rangle + \langle y \rangle$ by adding only the y-coordinate of the points. Then, $z = \mathsf{Reconstruct}(\mathbb{G}, \langle z \rangle) = x + y$.*

Proof. From the definition of Shamir secret sharing in Scheme 3, we get $x = p_x(0)$, $y = p_y(0)$ and $z = p_z(0)$, where p_x, p_y and p_z are the polynomials interpolating the shares $\langle x \rangle, \langle y \rangle$ and $\langle z \rangle$, respectively. Due to basic properties of polynomial additions ($\forall x \in \mathbb{F} : f(x) + g(x) = (f + g)(x)$), adding the y-coordinate of all points with the same x-coordinate results in a polynomial p_z that is the addition of the two polynomials p_x and p_y as shown below:

$$
\begin{aligned}
p_x(X) &= x + a_1 X + \cdots + a_{n-1} X^{n-1}\,, \\
p_y(X) &= y + b_1 X + \cdots + b_{n-1} X^{n-1}\,, \\
p_z(X) &= (x + y) + (a_1 + b_1) X + \cdots + (a_{n-1} + b_{n-1}) X^{n-1}\,.
\end{aligned}
$$

The result of $p_z(0)$ is therefore equal to $x + y$. □

Multiplication of Secret Shared Values For the case of additions we used the fact that for any two polynomials over a finite field $\mathbb{F}$ it holds that $\forall x \in \mathbb{F} : f(x) + g(x) = (f + g)(x)$. However, a careful reader might remark that this is also true for multiplications. Indeed, $(\forall x \in \mathbb{F} : f(x) \cdot g(x) = (f * g)(x))$, where $*$ denotes polynomial multiplication. It seems that we can now also perform multiplication of shares locally by multiplying the y-coordinates.

Theorem 4. *Let $\mathbb{F}$ be a finite field and $x, y \in \mathbb{F}, n \in \mathbb{N}$. Let $\langle x \rangle = \mathsf{Share}(\mathbb{F}, x, n, n)$ and $\langle y \rangle = \mathsf{Share}(\mathbb{F}, y, n, n)$ be the Shamir secret sharing of x and y respectively. Set $\langle z \rangle = \langle x \rangle \cdot \langle y \rangle$ by multiplying only the y-coordinate of the points. Then the polynomial p_z, interpolating the points $\langle z \rangle$, evaluated at 0 is equal to xy.*

Proof. From the definition of Shamir secret sharing in Scheme 3, we get $x = p_x(0)$, $y = p_y(0)$ and $z = p_z(0)$, where p_x, p_y and p_z are the polynomials interpolating the shares $\langle x \rangle, \langle y \rangle$ and $\langle z \rangle$, respectively. Due to basic properties of polynomial additions $(\forall x \in \mathbb{F} : f(x) \cdot g(x) = (f * g)(x)$, where $*$ denotes polynomial multiplication), multiplying the y-coordinate of all points with the same x-coordinate results in a polynomial p_z that is the multiplication of the two polynomials p_x and p_y as shown below:

$$\begin{aligned} p_x(X) &= x + a_1 X + \cdots + a_{n-1} X^{n-1} \,, \\ p_y(X) &= y + b_1 X + \cdots + b_{n-1} X^{n-1} \,, \\ p_z(X) &= (xy) + (a_1 y + b_1 x) X + \cdots + (a_{n-1} b_{n-1}) X^{2n-2} \,. \end{aligned}$$

The result of $p_z(0)$ is therefore equal to xy. □

However, the method detailed in Theorem 4 has one practical problem: Although the evaluation of p_z at 0 correctly results in xy, the degree of that polynomial has grown. A call to $\mathsf{Reconstruct}$ would now need $2t - 1$ unique shares to be able to correctly reconstruct the result. In our case of $t = n$, we see that we therefore can never have enough shares to correctly recover the value of xy.

2.4 Polynomial Degree Reduction

To combat this problem when using Shamir secret sharing in practice, parties holding the shares need to engage in a polynomial degree reduction protocol. The goal of this protocol is to reduce the degree of the polynomial p_z, which is the result after the local multiplication back down to a polynomial of degree $t-1$, so t points are enough to uniquely determine this polynomial.

Such degree reduction protocols require interaction between the parties holding the shares, again leaving us in a similar situation as we had in the case of additive secret sharing: additions can be performed completely locally, however for multiplications, the shareholders need to engage in some form of communication.

The oldest protocol for this is by Ben-Or, Goldwasser and Widgerson [2] and involves a re-randomization of the coefficients of the truncation of p_z. For a detailed description, we refer the reader to [2, The degree reduction step & The randomization step].

References

1. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: CRYPTO. Lecture Notes in Computer Science, vol. 576, pp. 420–432. Springer (1991)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC. pp. 1–10. ACM (1988)
3. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: CRYPTO. Lecture Notes in Computer Science, vol. 7417, pp. 643–662. Springer (2012)
4. Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In: ACM Conference on Computer and Communications Security. pp. 830–842. ACM (2016)
5. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)