CS556 Cryptography
Final Project: Secret Sharing

Spring 2022

Kyle Monette

ABSTRACT

Secret sharing is a general procedure for distributing information about a secret in shares, in such a way that a minimum number of shares will reveal the secret but any fewer will not. We call this a (n, k) threshold scheme, and the most famous one is Shamir's Secret Sharing procedure [1] based on polynomial interpolation. Here, we discuss Shamir's procedure, provide some properties, and describe some modifications that can be done. A widely known extension is verifiable secret sharing and publicly verifiable secret sharing, which is explained later. Finally, we conclude with Asmuth-Bloom's scheme that uses the Chinese Reminder Theorem as an example of secret sharing without Shamir's approach (i.e., without interpolation).

CONTENTS

1 Introduction

Suppose the CEO of a large company has a safe in their office that contains important information, such as passwords to accounts, money, billing information, etc. If they are the only ones that can access the safe, then how would their predecessor or direct reports unlock the safe if something were to happen to the CEO? One solution is to give trusted individuals the combination to the safe to use in an emergency or unfortunate situations. But what if the password is leaked out? Obviously this is the least secure option, but it is the easiest way to access the contents of the safe if the CEO cannot. Ideally, we want something in the middle: the most security possible with the greatest redundancy.

Suppose the CEO could give out pieces of the combination, say one character or number, to each of the board members of the corporation. Then, if an emergency arose, the board members could come together and assemble the combination. This is certainly better than giving each member the combination in its entirety. However, even with all the correct characters, it would take substantial time to determine the correct ordering. This could be compounded with the possibility of members loosing their character, making the combination extremely difficult to crack.

Another approach is to use *secret* sharing. As the name implies, parts of the secret are given out to shareholders, but in such a way that a certain number of them (and no less) have to come together and share their pieces in order to acquire the secret. Any fewer shareholders will have no information about the secret.

2 Brief History

Secret sharing was proposed in 1979 by Adi Shamir [1] and George Blakley working independently. Shamir created a polynomial interpolation method by exploiting the fact that n coordinate points uniquely define a degree $n-1$ polynomial, and Blakley created a geometric procedure by exploiting the fact that n nonparallel $n-1$ dimensional planes intersect at only one point. Six years later in 1985, Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch proposed verifiable secret sharing (explained later) as a way to ensure that the secret can be uniquely determined from the distributed shares.

There are now many secret sharing procedures, including schemes based on the Chinese Remainder Theorem (explained later) and *proactive secret sharing*, which have legitimate applications. For instance, secret sharing is used in secure multiparty computation, where parties compute a function given their inputs while keeping their inputs private, and also appears in user authentication in various systems.

3 Primitive Secret Sharing

It is worth noting, before we introduce the mathematics behind Shamir's scheme, that secret sharing can be done in a quite primitive way in that there is *no* math involved at all.

Suppose your friend has a phone with a six digit password (numbers 0 to 9). If they trust you and two others, say, they could give each of you a string:

3_4___ _1__5_ ___1_9

On their own nor with only two strings together could you recover the secret password. However, once all three friends are present, the secret can be revealed easily. We can imagine that if letters (and perhaps numbers together) are used, it would be unreasonable to guess each character in a brute-force attack. This makes this primitive version unsecure but also not trivial.

4 Shamir's Secret Sharing

Shamir's secret sharing (SSS) procedure [1] is based on the concept of polynomial interpolation. Specifically, we consider these polynomials to be over a finite field to make the secret impossible to determine from brute-force attacks. In SSS, we link the idea of secret sharing to polynomial interpolation via the fact that a real valued polynomial $p(x)$ of degree n requires $n+1$ distinct points to uniquely define it (e.g., a line requires two distinct points, a parabola requires three, etc). Because polynomials are continuous functions, it is relatively easy to brute-force through possible combination of coefficients and therefore a finite field is necessary.

4.1 The Procedure

Suppose the "dealer" has a secret s which can be written as $a_0 \in \mathbb{F}$, where $a_0 \neq 0$ and $\mathbb{F}$ is a field. Ideally, we want $\mathbb{F} = \mathbb{Z}_p$ for p prime, but for the sake of explanation, we can also have $\mathbb{F} = \mathbb{R}$.

Note: If $\mathbb{F} = \mathbb{Z}_p$, all operations below are carried out modulo p.

1. The dealer chooses a prime p and the number of shares n such that $2 \leq n < p$ and the threshold k such that $2 \leq k \leq n$.

2. The dealer chooses nonzero constants $a_1, \ldots, a_{k-1} \in \mathbb{F}$ randomly, and then constructs the polynomial

$$f(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}.$$

3. The dealer creates a set of n shares[1]

$$\{(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n))\} \qquad (x_i \in \mathbb{F},\ x_i \neq x_j\ \forall\, i \neq j).$$

4. The dealer assigns to each "shareholder" one and only one share from this set, and tells each shareholder the value of k (it is not necessary, and can be dangerous, to share n).

5. Given any subset of k shares, shareholders can obtain $a_0 = f(0)$ using polynomial interpolation by computing

$$s = a_0 = \sum_{j=0}^{k-1} f(x_j) \prod_{\substack{m=0 \\ m \neq j}}^{k-1} \frac{x_m}{x_m - x_j}. \tag{1}$$

It may be possible to not share k with the shareholders, however this is not a convenient situation. If the dealer shares k, then the shareholders can communicate and ensure that enough ($= k$) of them are present.

4.2 Mathematical Background

Equation (1) is derived using *Lagrange interpolating polynomials.*

Definition 4.1.
Given a set of $k+1$ data points $(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$, where the x_j are distinct, we define the Lagrange interpolating polynomial as

$$L(x) = \sum_{j=0}^{k} f(x_j) \prod_{\substack{m=0 \\ m \neq j}}^{k} \frac{x - x_m}{x_j - x_m}.$$

For SSS, it is not necessary for us to calculate the entire polynomial as we only care about the root (a_0), hence the modified result in equation (1). Furthermore, if the field is $\mathbb{Z}_p$ then we compute the polynomial modulo p.

It is necessary that this interpolant is unique for each set of data points—else, we would not necessarily be able to recover our secret. Fortunately, this is guaranteed with the following theorem [2].

Theorem 4.2.
Let $(x_1, y_1), \ldots, (x_n, y_n)$ be n points with distinct x_i. Then there exists only one polynomial P of degree $n-1$ or less that satisfies $P(x_i) = y_i$ for $i = 1, \ldots, n$.

[1] In the original paper [1] and many online resources, it is common to choose the sequence to be $x = 1, 2, \ldots, n$. It is unclear at this time why this is the standard as it is significantly easier to crack if $\mathbb{F} = \mathbb{R}$. If $\mathbb{F} = \mathbb{Z}_p$, we must choose x_i from $0, 1, \ldots, p-1$ obviously, but it seems that the x_i need not be consecutive in the sequence.

Proof.
The existence is given by the Lagrange interpolating polynomial. We show here that the interpolant is unique.

Suppose there are two polynomials P and Q of degree $n-1$ or less that interpolate the points. That is, $P(x_1) = y_1, Q(x_1) = y_1$, etc. Define $H(x) = P(x) - Q(x)$. Then $\deg H \leq n-1$ as $\deg P \leq n-1$ and $\deg Q \leq n-1$. Also we can see that

$$H(x_1) = P(x_1) - Q(x_1) = y_1 - y_1 = 0, \quad H(x_2) = P(x_2) - Q(x_2) = y_2 - y_2 = 0, \ldots.$$

Therefore $H(x)$ has n distinct zeros since we assumed x_i are distinct. By the fundamental theorem of algebra, a nonzero degree n polynomial has at most n zeros. However, H has degree $n-1$ or less and has n zeros. Therefore, it must be that $H \equiv 0$ (the zero polynomial). Therefore $P \equiv Q$. ■

Remark 4.3.
This theorem holds for polynomials of degree $n-1$ or less where we are given n points. In fact, there are infinitely many degree n polynomials that interpolate n points.

4.3 Interpolation Methods

While using Lagrange's method for constructing the polynomial is sufficient, it is not computationally friendly. One disadvantage of this approach is that we have to recalculate the polynomial if another point is added to the interpolation set. An alternative is to use *Newton's divided difference* (NDD) method. This will not create a different polynomial—we showed previously that the interpolating polynomial is unique. To use NDD, we first need to define the divided difference [2].

Definition 4.4.
The *divided difference* is denoted by $f[x_1, \ldots, x_n]$ and is the coefficient of the x^{n-1} term in the polynomial that interpolates $(x_1, f(x_1)), \ldots, (x_n, f(x_n))$.

The interpolation polynomial is given by

$$P(x) = f[x_1, \ldots, x_n](x - x_1)\ldots(x - x_{n-1}).$$

For example, the points $(0,1), (2,2), (3,4)$ yield the interpolating polynomial $f(x) = \frac{1}{2}x^2 - \frac{1}{2}x + 1$. Therefore $f[0,2,3] = \frac{1}{2}$, or any permutation of $0, 2, 3$. That is, $f[0,3,2] = f[3,0,2] = f[0,2,3]$.

Divided differences can be calculated recursively:

$$\begin{aligned} f[x_k] &= f(x_k) \\ f[x_k, x_{k+1}] &= \frac{f[x_{k+1}] - f[x_k]}{x_{k+1} - x_k} \\ f[x_k, x_{k+1}, x_{k+2}] &= \frac{f[x_{k+1}, x_{k+2}] - f[x_k, x_{k+1}]}{x_{k+2} - x_k} \\ &\vdots \end{aligned}$$

Example 4.5.
Say we are given $(0,1), (2,2), (3,4)$. Then

$$\begin{aligned} f[x_1] &= f(0) = 1 \\ f[x_1, x_2] &= f[0,2] = \frac{f[2] - f[0]}{2 - 0} = \frac{1}{2} \\ f[x_1, x_2, x_3] &= f[0,2,3] = \frac{f[2,3] - f[0,2]}{3 - 0} = \frac{1}{2} \end{aligned}$$

Therefore

$$P(x) = 1 + \frac{1}{2}(x-0) + \frac{1}{2}(x-0)(x-2) = \frac{1}{2}x^2 - \frac{1}{2}x + 1.$$

Now suppose we need to add another points $(1, 0)$. Then we only have to calculate

$$f[x_1, x_2, x_3, x_4] = f[0, 2, 3, 1] = -\frac{1}{2}.$$

Then the new polynomial $\widetilde{P}(x)$ is

$$\widetilde{P}(x) = P(x) - \frac{1}{2}(x-0)(x-2)(x-3) = -\frac{x^3}{2} + 3x^2 - \frac{7}{2}x + 1.$$

△

Note that in this example we could have added *any* point to the interpolating set, and would have received a new polynomial each time. This justifies the claim made in Remark 4.3 that there are infinitely many degree n polynomials that interpolate n points.

4.4 Digression: Chebyshev Nodes

In the Lagrange and Newton's divided difference methods, we use use equally spaced nodes to generate the interpolating polynomial. For a higher degree interpolant, this can lead to *Runge's phenomenon*, where the interpolant oscillates at a high frequency at the end of the interpolation domain. In the case of SSS, if we use polynomials defined over $\mathbb{R}$, we might encounter this phenomenon if high degree polynomials are used (k is taken to be large). In theory, this should not cause any problems as the polynomial is still interpolating correctly. Rather the danger comes in floating point representations and calculation error as the coefficients and share coordinates could become too large for the computer to accurately compute.

A way to combat Runge's phenomenon is to use *Chebyshev nodes*. Then, we use these values to generate the interpolant in the same way that we did in the classical approach.

Chebyshev nodes [2] are defined on the interval $[-1, 1] \subset \mathbb{R}$ by

$$x_i = \cos\left(\frac{\pi(2i-1)}{n}\right) \qquad i = 1, 2, \ldots, n.$$

We can extend this to any interval $[a, b]$ by

$$x_i = \frac{b+a}{2} + \frac{b-a}{2}\cos\left(\frac{\pi(2i-1)}{n}\right).$$

Note that we cannot use Chebyshev nodes if the field is $\mathbb{Z}_p$. Therefore, if the dealer insists on using $\mathbb{R}$ as the field, then the use of Chebyshev nodes (with NDD for computational advantages) is perhaps the next best option.

4.5 Example: Integer Arithmetic

Here we consider SSS with $\mathbb{F} = \mathbb{R}$, both as an example of the calculations involved and to illustrate the lack of security that comes from integer arithmetic instead of modulo arithmetic.

Suppose our secret is given by $a_0 = 123$ with $n = 3$ and $k = 3$. We then choose $k - 1 = 2$ random natural numbers $a_1 = 94, a_2 = 166$. Therefore our polynomial is

$$f(x) = a_0 + a_1 x + a_2 x^2 = 123 + 94x + 166x^2.$$

We then make $n = 3$ shares of

$$(1, 383), \quad (2, 643), \quad (3, 903).$$

These points then go to the shareholders. To recover a_0, we need all $3 = k$ shareholders to come together and share their pieces. Then, we recover a_0 from

$$a_0 = \sum_{j=0}^{k-1} f(x_j) \prod_{\substack{m=0 \\ m \neq j}}^{k-1} \frac{x_m}{x_m - x_j}$$

$$
\begin{aligned}
&= \sum_{\substack{j=0 \\ j\neq m}}^{2} f(x_j)\left[\frac{x_0}{x_0-x_j}\frac{x_1}{x_1-x_j}\frac{x_2}{x_2-x_j}\right] \\
&= \frac{f(x_0)x_1x_2}{(x_1-x_0)(x_2-x_0)} + \frac{f(x_1)x_0x_2}{(x_0-x_1)(x_2-x_1)} + \frac{f(x_2)x_0x_1}{(x_0-x_2)(x_1-x_2)} \\
&= 1149 - 1929 + 903 \\
&= 123
\end{aligned}
$$

Note that we used a consecutive sequence for the shares by $\{x_n\} = 1, 2, 3$. Suppose a third party (or a malicious shareholder) Eve found two points $(1, 383)$ and $(2, 643)$. In theory, Eve should not be able to find a_0, but because we are using integer arithmetic and we have a consecutive sequence of $\{x_n\}$, Eve can narrow down their search for the missing coefficient quite dramatically.

Eve can construct the polynomial $f(x) = a_0 + a_1x + a_2x^2$. The shares she know tell her that

$$
\begin{aligned}
383 &= a_0 + a_1 + a_2 \\
643 &= a_0 + 2a_1 + 4a_2
\end{aligned}
$$

Combining these together,

$$643 - 383 = a_1 + 3a_2 \quad \Rightarrow \quad a_1 = 260 - 3a_2.$$

As she knows that $a_2 \in \mathbb{N}$, she can start guessing:

$$
\begin{aligned}
a_2 = 1 &\Rightarrow a_1 = 257 \\
a_2 = 2 &\Rightarrow a_1 = 254 \\
a_2 = 3 &\Rightarrow a_1 = 251 \\
&\vdots \\
a_2 = 86 &\Rightarrow a_1 = 2
\end{aligned}
$$

After this she can stop, as it was assumed that $a_2 \in \mathbb{N}$, and going further would make $a_1 < 0$. Therefore she has narrowed down the problem of determining the last coefficient of $f(x)$, where there are infinitely many choices, to only 86.

This example tells us some very important choices about SSS:

1. We should use finite fields with modular arithmetic to avoid brute-force attacks to narrow down the choices for the coefficients.
2. A higher degree polynomial with many coefficients is more secure, both in cases where $\mathbb{F} = \mathbb{R}$ and $\mathbb{F} = \mathbb{Z}_p$.
3. While literate suggests that the dealer should use a consecutive integer sequence for x values in the shares, it is unclear why this is necessary. Choosing random values (within a reasonable interval), especially in the case where finite fields are *not* used, results in a large increase in security.

4.6 Example: Finite Fields

Here we consider SSS with $\mathbb{F} = \mathbb{Z}_p$, with $p = 1613$. As before, we'll take $k = n = 3$. Suppose our secret is given by $a_0 = 1234$. We then construct the polynomial

$$f(x) = a_0 + a_1x + a_2x^2 = 1234 + 166x + 94x^2 \pmod{p}.$$

The dealer distributes the three shares of

$$(1, 1494), \quad (2, 329), \quad (3, 965).$$

For the sake of explanation, we'll use NDD to recover the secret. As defined, the shareholders would calculate the polynomial

$$f(x) = f[x_1] + f[x_1, x_2](x - x_1) + f[x_1, x_2, x_3](x - x_1)(x - x_2).$$

The divided differences are

$$\begin{aligned} f[x_1] &= f[1] = 1494 \\ f[x_1, x_2] &= f[1, 2] = -1165 = 448 \\ f[x_1, x_2, x_3] &= f[1, 2, 3] = \frac{f[2, 3] - f[1, 2]}{3 - 1} = 94 \end{aligned}$$

Recall that these operations are being done modulo $p = 1613$. The interpolant is

$$f(x) = 1494 + 448(x - 1) + 94(x - 1)(x - 2) = 1234 + 166x + 94x^2.$$

Note that $a_0 \neq f[x_1]$! That is, it is not enough to stop the procedure at this one step.

Now imagine that Eve was sneaking around again and got two shares: $(1, 1494)$ and $(2, 329)$. Then she knows (by knowing that $k = 3$) that

$$\begin{aligned} 1494 &= a_0 + a_1 + a_2 \\ 329 &= a_0 + 2a_1 + 4a_2 \end{aligned}$$

This implies that

$$\begin{aligned} 1494 &= a_0 + a_1 + a_2 + k_1 p \\ 329 &= a_0 + 2a_1 + 4a_2 + k_2 p \end{aligned}$$

for $k_1, k_2 \in \mathbb{N}$ from modular arithmetic. Subtracting these,

$$329 - 1494 = a_1 + 3a_2 + p(k_2 - k_1) \quad \Rightarrow \quad a_1 = -1165 - p(k_2 - k_1) - 3a_2.$$

Eve can then begin to guess a_2 by:

$$\begin{aligned} a_2 = 0 &\Rightarrow a_1 = -1165 - p(k_2 - k_1) \\ a_2 = 1 &\Rightarrow a_1 = -1168 - p(k_2 - k_1) \\ &\vdots \end{aligned}$$

Unlike before, this sequence can continue on and thus Eve does not know anything about the missing coefficient because she does not know the constants k_1 and k_2.

4.7 Properties

In the original framework [1], some properties of SSS are stated:

1. Secure: SSS with a finite field is a system that has *information-theoretic security* in that it is impossible to break even with infinite computational power (having fewer than k shares provides no information about the secret).
2. Minimal: The size of each share does not exceed the size of the original data.
3. Extensible: More shares can be generated without affecting other shares (which is easier to do with NDD, as we saw).
4. Dynamic: Security can be improved without changing the secret by increasing k. Granted the shareholders would have to know about this, but it can be done.
5. Flexible: People of higher importance can be given a smaller "local" threshold number. For example, the president only needs three shares but a secretary needs five.

5 Verifiable Secret Sharing

There are two main issues in SSS:

1. The shareholders could contribute false shares, perhaps to get information about other shares in an effort to reconstruct the secret without them.

2. The dealer could distribute false shares so that multiple secrets are generated, perhaps in the scenario of a very malicious dealer.

These issues can be solved using *verifiable secret sharing* (VSS). Here, we allow participants to ensure that they can indeed recover a unique secret without (in theory) the dealer revealing any information about it.

How do we know that other shareholders received valid shares, without revealing their shares to the audience? This question can be solved using *publicly verifable secret sharing* (PVSS). Here, not only can shareholders verify their shares individually but outsiders (shareholders or not) can determine if all shares are valid.

That is, we seek *verification of correctness* so that the shareholders are required to submit accurate and valid shares, and so that the shareholders are sure they recovered the actual secret disclosed by the dealer.

5.1 VSS: Discrete Logarithms

As described in [3], we now explain a method that can verifiably share discrete logarithms. This method can be extended to PVSS by means of an encryption scheme that allows us to verify that cipher text contains the discrete logarithm of a given value.

First, we give a theoretical and general framework and then move to a (n, k) threshold scheme. As done in [3], we define and use *access structures*.

Definition 5.1 (Access structure)**.**
An access structure $\mathcal{A}$ is a subset of $\mathcal{P}(\{1, 2, \ldots, n\})$ (the power set).

In essence, the access structure is the set of qualified subsets that are allowed to use the structure. We also require that the structure is monotone; that is,

$$A \in \mathcal{A} \quad \text{and} \quad A \subseteq B \quad \Rightarrow \quad B \in \mathcal{A}.$$

In SSS, we could have defined an access structure as

$$\mathcal{A} = \{A \in \mathcal{P}(\{1, \ldots, n\}) \mid |A| \geq k\}$$

where $|A|$ is the cardinality of A.

Now we can explain the procedure for a given $\mathcal{A}$. Let p be a large prime, let $q = \frac{p-1}{2}$ be prime, and $h \in \mathbb{Z}_p^*$ such that $\operatorname{ord} h = q$. Let G be a group of order p and g a generator of G.

1. The dealer determines the secret $s \in \mathbb{Z}_p$ and defines $S = g^s$, where S is public.

2. For each $A = \{j_1, \ldots, j_k\} \in \mathcal{A}$, the dealer computes the shares

$$S_{A_i} = \begin{cases} \text{random element of } \mathbb{Z}_p & i = j_1, \ldots, j_{k-1} \\ s - \sum_{l=1}^{k-1} S_{A_{i_l}} \pmod p & i = j_k \end{cases}.$$

   Each S_{A_i} gets sent to shareholder P_i.

3. The dealer publishes $F_{A_i} \coloneqq g^{S_{A_i}}$ so that everyone can verify that

$$\forall\, A \in \mathcal{A},\ \prod_{i \in A} F_{A_i} = S = g^s.$$

4. The shareholders can verify their own shares by checking whether S_{A_i} is the discrete logarithm of F_{A_i}.

Now in a (n, k) threshold scheme, we assign to each shareholder $x_i \in \mathbb{Z}_p$, $x_i \neq 0$. The dealer chooses a random $f_j \in \mathbb{Z}_p$ for $j = 1, \ldots, k-1$ and publishes $S = g^s$ and $F_j = g^{f_j}$ for $j = 1, \ldots, k-1$. Each shareholder then secretly receives the share

$$S_i = s + \sum_{j=1}^{k-1} f_j x_i^j \pmod{p}.$$

Any subset of k shareholders can recover s from Lagrange's interpolating polynomial. To verify the share S_i that shareholder P_i received, they compute

$$S \prod_{j=1}^{k-1} F_j^{x_i^j}$$

and checks that this is g^{S_i}.

Proof.
In the verification, the shareholder i computes F_i, which is

$$F_i = S \prod_{j=1}^{k-1} F_j^{x_i^j}.$$

We show here that if the share is valid, $F_i = g^{S_i}$.

$$\begin{aligned} F_i &= g^s \prod_{j=1}^{k-1} \left(g^{f_j}\right)^{x_i^j} = g^s \left[g^{f_1 x_i} \times g^{f_2 x_i^2} \times \cdots \times g^{f_{k-1} x_i^{k-1}}\right] \\ &= g^s g^{\sum_{j=1}^{k-1} f_j x_i^j} \\ &= g^s g^{S_i - s} = g^{S_i} \end{aligned}$$

■

5.2 PVSS: Discrete Logarithms

In order to make the scheme in the last section publicly verifiable, we need a public key encryption system that allows us to verifiably encrypt the discrete logarithm of a public element. Here, we use ElGamal, but there are other alternatives. With p, q, and h from the last section, we illustrate the scenario:

1. Each shareholder chooses $z \in \mathbb{Z}_q$ secretly and publishes their $y = h^z \pmod{p}$.

2. The dealer encrypts $m \in \mathbb{Z}_p$ with public key y by randomly choosing $\alpha \in \mathbb{Z}_q$ and calculating

$$A = h^\alpha \pmod{p}, \quad B = m^{-1} y^\alpha \pmod{p}.$$

3. Shareholders can decrypt by calculating $m = A^z B^{-1} \pmod{p}$.

Further, we need a protocol for verifying that (A, B) encrypts the discrete logarithm of $V = g^m \in G$. The idea is that if $A = h^\alpha$ and $B = m^{-1} y^\alpha$, then

$$V^B = v^{m^{-1} y^\alpha} = (g^m)^{m^{-1} y^\alpha} = g^{m m^{-1} y^\alpha} = g^{y^\alpha}.$$

The prover (dealer) proves that the discrete logarithm of A in base h is the double discrete logarithm of V^B in bases g and y. In other words, they prove that they have an α such that $A = h^\alpha$ and $V^B = g^{y^\alpha}$.

1. The dealer chooses $w \in \mathbb{Z}_q$ randomly, computes $t_h = h^w \pmod{p}$, and $t_g = g^{y^w}$. They send t_h and t_g to the verifier.

2. The verifier chooses $c \in \{0, 1\}$ randomly and sends c back.
3. The dealer computes $r = w - c\alpha \pmod q$ and sends r.
4. The verifier checks that
$$t_h = h^r A^c \pmod p, \qquad t_g = \begin{cases} g^{y^r} & c = 0 \\ V^{By^r} & c = 1 \end{cases}.$$

Say that $c = 0$. Then $r = w \pmod q$, so $t_h = h^w = h^r = h^r A^0$ and this checks. Further, $t_g = g^{y^w} = g^{y^r}$, and this checks.

Say that $c = 1$. Then $r = w - \alpha \pmod q$, so $t_h = h^w = h^{r+\alpha} = h^r h^\alpha = h^r A^1$ and this checks. Further, $t_g = g^{y^w} = g^{y^{r+\alpha}} = g^{y^r y^\alpha}$ and $B = m^{-1} y^\alpha$, so this becomes $g^{y^r mB}$. Since $V = g^m$, we then get the desired $V^{y^r B}$.

5.3 Feldman's Scheme

In the last section, consider letting f_j be the a_j for $j = 1, \ldots, k-1$ that we saw in SSS. That is, regard f_j as the polynomial coefficients. Then
$$S_i = s + \sum_{j=1}^{k-1} f_j x_i^j = f(x_i).$$
These are exactly the shares we saw in SSS. Converting this theoretical approach in the last two sections back to SSS is (essentially) Feldman's scheme [4].

1. The dealer chooses a cyclic group G, such as $\mathbb{Z}_n$, of prime order q together with a generator g. The group and generator are public. Typically, $G = \mathbb{Z}_p^*$ with order $q | p - 1$ for a prime q.
2. The dealer computes secretly a random polynomial
$$f(x) = a_0 + a_1 x^2 + \cdots + a_{k-1} x^{k-1}$$
of degree $k - 1$ from $\mathbb{Z}_q[x]$ such that $f(0) = a_0$, where a_0 is the secret.
3. The n shares are computed (as in SSS) by $f(1), \ldots, f(n) \pmod q$. Any k shares can recover the secret using interpolation.

So far, this is identical to SSS over a finite field $\mathbb{Z}_q$. Next, the dealer distributes auxiliary information to the coefficients of f modulo q:
$$c_0 = g^{a_0}, c_1 = g^{a_1}, \ldots, c_{k-1} = g^{a_{k-1}}.$$
With this information distributed, any shareholder can verify the consistency of their share. That is, for shareholder i where $1 \le i \le n$, they can verify that $y = f(i) \pmod q$ where their share is $f(i)$ and y is some image (ideally their share) from the polynomial:

$$\begin{aligned} g^y &= c_0 c_1^i c_2 i^2 \ldots c_{k-1} i^{k-1} \\ &= \prod_{j=0}^{k-1} c_j^{i^j} \\ &= \prod_{j=0}^{k-1} (g^{a_j})^{i^j} && \text{definition of } c_j \\ &= \prod_{j=0}^{k-1} g^{a_j i^j} \\ &= g^{\sum_{j=0}^{k-1} a_j i^j} && \text{power laws} \\ &= g^{f(i)} && \text{definition of } f \end{aligned}$$

Feldman's scheme is secure, but perhaps only for attackers that have some computing restraint as it would be possible to crack with enough computations. Another issue is that releasing g^{a_0} reveals information about the secret. With a small enough group, this could be especially detrimental.

6 Chinese Remainder Theorem

There exists a secret sharing scheme based on the Chinese Remainder Theorem, which we will now discuss. The idea is similar to Shamir's procedure in that it is an (n, k) threshold scheme, but rather than using polynomial interpolation and the uniqueness of an interpolant, we use the idea that we can uniquely determine a system of congruences given coprime moduli.

First, we need a few tools from number theory.

6.1 Mathematical Background

Theorem 6.1 (Chinese Remainder Theorem)**.**
Let $m_1, m_2, \ldots, m_k$ be pairwise relatively prime positive integers. Then the system of congruences

$$\begin{cases} x & \equiv a_1 \pmod{m_1} \\ x & \equiv a_2 \pmod{m_2} \\ & \vdots \\ x & \equiv a_k \pmod{m_k} \end{cases}$$

has a unique solution modulo $M = m_1 m_2 \ldots m_k$.

This theorem does not give us a procedure of calculating the solution x, which we will show now.

1. Compute $M = m_1 m_2 \ldots m_k$.
2. Compute
$$M_1 = \frac{M}{m_1},\ M_2 = \frac{M}{m_2}, \ldots, M_k = \frac{M}{m_k}.$$
3. Solve the system of congruences for y_k
$$\begin{cases} M_1 y_1 & \equiv 1 \pmod{m_1} \\ M_2 y_2 & \equiv 1 \pmod{m_2} \\ & \vdots \\ M_k y_k & = 1 \pmod{m_k} \end{cases}$$
4. Then the solution x is given by
$$x \equiv a_1 M_1 y_1 + \cdots + a_k M_k y_k \pmod{M}.$$

6.2 Asmuth-Bloom Scheme

The Asmuth-Bloom secret sharing scheme (ABS) [5] uses the Chinese Remainder theorem and is one of two popular secret sharing procedures that does so (the other being Mignotte's, not discussed here). Asmuth-Bloom was proposed in 1983, four years after SSS. The procedure is as follows:

1. The dealer chooses the number of shares n and the threshold k such that $2 \leq k \leq n$, and constructs a sequence of pairwise coprime positive integers
$$p < m_1 < m_2 < \cdots < m_n$$

satisfying the property that

$$M \coloneqq \prod_{i=1}^{k} m_i > p \prod_{i=1}^{k-1} m_{n-i+1}.$$

2. The dealer defines the secret S to be an integer such that $0 \leq S < p$.
3. The dealer then chooses $\alpha \in \mathbb{Z}$ and computes $y = S + \alpha p$ such that $0 \leq y < M$, where y is kept secret.
4. The n shares (y_i, m_i) are distributed, where each y_i is such that $y_i \equiv y \pmod{m_i}$.
5. Given k shares, shareholders can uniquely determine S from solving their system of congruences using the Chinese Remainder Theorem as

   $$y \equiv S + \alpha p \equiv S \pmod{p}.$$

   That is, shareholders recover y from the Chinese Remainder Theorem and take y modulo p (a public modulus) to recover S.

Asmuth-Bloom has the property, akin to SSS, that $k-1$ or fewer shares will reveal no information about the secret. Suppose only $k-1$ shares are given to a participant. Then the collection of n_i such that $n_i \equiv y \pmod{N}$ where N is the product of $m_1, \ldots, m_{k-1}$ and $n_i \leq M$ cover all congruence classes modulo p. Each class contains at most one more or one less n_i than any other, so there is no information given about the secret S.

6.3 Example: Asmuth-Bloom

Let $n = 4$, $k = 3$, and $S = 2$. We then create the sequence

$$p = 3, m_1 = 11, m_2 = 13, m_3 = 17, m_4 = 19.$$

Note that the sequence is valid because all terms are pairwise coprime and

$$M = \prod_{i=1}^{3} m_i = 2431 > p \prod_{i=1}^{2} m_{n-i+1} = 969.$$

We randomly pick $\alpha = 51$ which defines

$$y = S + \alpha p = 155 < M.$$

The $4 = n$ shares are calculated as

$$\begin{aligned} y_1 &\equiv 1 \pmod{11} \\ y_2 &\equiv 12 \pmod{13} \\ y_3 &\equiv 2 \pmod{17} \\ y_4 &\equiv 3 \pmod{19} \end{aligned}$$

We require $3 = k$ shares to recover the secret, say $\{1, 12, 2\}$. Then we use the procedure in the Chinese Remainder Theorem to determine y modulo N. Here, we determine

$$N = 11 \times 13 \times 17 = 2431,\ N_1 = \frac{N}{m_1} = 221,\ N_2 = \frac{N}{m_2} = 187,\ N_3 = \frac{N}{m_3} = 143.$$

Our system of congruences is then

$$\begin{aligned} N_1 x_1 &\equiv 1 \pmod{m_1} \\ N_2 x_2 &\equiv 1 \pmod{m_2} \\ N_3 x_3 &\equiv 1 \pmod{m_3} \end{aligned} \quad \Rightarrow \quad \begin{aligned} 221 x_1 &\equiv 1 \pmod{11} \\ 187 x_2 &\equiv 1 \pmod{13} \\ 143 x_3 &\equiv 1 \pmod{17} \end{aligned} \quad \Rightarrow \quad \begin{aligned} x_1 &= 1 \\ x_2 &= 8 \\ x_3 &= 5 \end{aligned}$$

Thus, y is

$$y \equiv y_1 x_1 N_1 + y_2 x_2 N_2 + y_3 x_3 N_3 \equiv 19603 \equiv 155 \pmod{N}.$$

Then we determine

$$S \equiv 155 \pmod{3} \quad \Rightarrow \quad S = 2 \pmod{3},$$

as was expected.

BIBLIOGRAPHY

[1] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[2] Timothy Sauer. *Numerical Analysis*. Addison-Wesley Publishing Company, 2011.

[3] Markus Stadler. Publicly verifiable secret sharing. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 190–199. Springer, 1996.

[4] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438. IEEE, 1987.

[5] Charles Asmuth and John Bloom. A modular approach to key safeguarding. *IEEE transactions on information theory*, 29(2):208–210, 1983.