Secret Sharing

Noah Stephens-Davidowitz

June 9, 2023

1 Secret Sharing

Imagine that there is some extremely sensitive piece of information $s \in \{0,1\}^\ell$—maybe it's the codes to launch a nuclear missile or the script for the season finale of your favorite TV show. This information might be too sensitive to trust with any one person, but maybe we can trust it to a group of people. E.g., maybe we can give Alice some bit string σ_1, give Bob σ_2, and Charlie σ_3, in such a way that (1) knowledge of σ_1, σ_2, and σ_3 is sufficient to recover s; but (2) knowledge of just σ_i for some fixed i is not sufficient to recover s. We call this a *secret-sharing scheme*, and we call s the *secret* and σ_i the *shares of* s.

In fact, we can do better. We can make it so that σ_1, σ_2, and σ_3 together are enough to recover the secret s, but *any pair* of shares (σ_i, σ_j) reveals *no information* about the secret s.

This abstract discussion is actually rather silly for this particular example because there is such a simple scheme that achieves this. Let $\sigma_1, \sigma_2 \sim \{0,1\}^\ell$ (i.e., Alice and Bob are given uniformly random ℓ-bit strings), and let $\sigma_3 := s \oplus \sigma_1 \oplus \sigma_2 \in \{0,1\}^\ell$. (This scheme might seem strangely asymmetric, since Charlie's share σ_3 was constructed differently than the other two shares. But, it is actually completely symmetric, since we can equivalently describe this scheme as sampling uniformly random $(\sigma_1, \sigma_2, \sigma_3)$ subject to the constraint that $\sigma_1 \oplus \sigma_2 \oplus \sigma_3 = s$.) Then, it is trivial to recover s given σ_1, σ_2, and σ_3, since $s = \sigma_1 \oplus \sigma_2 \oplus \sigma_3$. However, clearly σ_1 and σ_2 alone yield no information whatsoever about s, since they were chosen uniformly random and independently of s. And, a simple analysis shows that (σ_2, σ_3) is also uniformly random and independent of s, as is (σ_1, σ_3).

This is called a 3-out-of-3 secret-sharing scheme because there are three shares, and all three of them are required to reconstruct the secret s. We can generalize this idea to t-out-of-k secret sharing for any $1 \le t \le k$ as follows.

Definition 1.1. *For $1 \le t \le k$, a t-out-of-k secret-sharing scheme with secret space $\mathcal{M}$ is a pair of algorithms* Share *and* $\mathsf{Reconstruct}$ *with the following properties.*

1. **(Correctness.)** *For any secret $s \in \mathcal{M}$ and any t distinct indices $i_1, \ldots, i_t \in [k]$,*

$$\Pr_{(\sigma_1,\ldots,\sigma_k) \leftarrow \mathsf{Share}(s)} [\mathsf{Reconstruct}((i_1, \sigma_{i_1}), \ldots, (i_t, \sigma_{i_t})) = s] = 1 \, .$$

   (In other words, if the $\mathsf{Reconstruct}$ *algorithm is given any t distinct shares—together with the corresponding indices i_j—it can always reconstruct the secret.)*

2. **(Security.)** *For any two secrets $s, s' \in \mathcal{S}$ and any $t-1$ indices $i_1, \ldots, i_{t-1} \in [k]$, the distributions $(\sigma_{i_1}, \ldots, \sigma_{i_t})$ and $(\sigma'_{i_1}, \ldots, \sigma'_{i_{t-1}})$ are identical, where $(\sigma_1, \ldots, \sigma_k) \leftarrow \mathsf{Share}(s)$ and $(\sigma'_1, \ldots, \sigma'_k) \leftarrow \mathsf{Share}(s')$. Equivalently, for any $\beta_1, \ldots, \beta_{t-1}$,*

$$\Pr_{(\sigma_1,\ldots,\sigma_k)\leftarrow\mathsf{Share}(s)}[\forall j,\ \sigma_{i_j} = \beta_j] = \Pr_{(\sigma'_1,\ldots,\sigma'_k)\leftarrow\mathsf{Share}(s')}[\forall j,\ \sigma'_{i_j} = \beta_j] \ .$$

(In other words, the joint distribution of any $t-1$ shares are independent of the secret s.)

Let's try to parse this definition. The correctness property is relatively straightforward—it says that any t distinct shares are sufficient to recover the secret, and specifically that this can be done via the $\mathsf{Reconstruct}$ algorithm. The security property is a formal way to say that any $t-1$ shares "reveal no information about the secret." Specifically, for any two secrets $s, s' \in \mathcal{M}$, if the shares were sampled by $\mathsf{Share}(s)$ or $\mathsf{Share}(s')$, the distribution of any $t-1$ shares will be exactly the same.

Notice that this is a purely information-theoretic security definition. We could have instead only asked that the shares $(\sigma_{i_1}, \ldots, \sigma_{i_{t-1}})$ and $(\sigma'_{i_1}, \ldots, \sigma'_{i_{t-1}})$ were computationally indistinguishable, rather than identically distributed. But, since we can get away with a stronger information-theoretic definition, we will stick with this.

Notice also that our XOR-based scheme above did in fact satisfy this definition for $k = t = 3$. More generally, a simple XOR-based scheme works for *any* $k = t$. E.g., to share $s \in \{0,1\}^\ell$ among k parties so that only all k parties together can recover the s, simply take $\sigma_1, \ldots, \sigma_{k-1} \sim \{0,1\}^\ell$ and $\sigma_k := s \oplus \sigma_1 \oplus \cdots \oplus \sigma_{k-1}$.

Another easy example is the case $t = 1$. In this case, the security property is vacuous (since it refers to collections of 0 shares), and the correctness property says that any one share σ_i should be sufficient to recover the secret. So, we can just set $\sigma_i = s$ for all i!

Things get interesting when $1 < t < k$. For example, one can imagine wanting a 2-out-of-k secret-sharing scheme, so that any *two* parties can reconstruct the secret, but no one party can. Or, for odd k, one can imagine a $(k+1)/2$-out-of-k secret-sharing scheme, so that a majority is necessary to reconstruct the secret. But, it is not at all clear how to do this.

2 An apparently unrelated discussion about polynomials modulo primes

Now, let's talk about something seemingly unrelated. Let $g(x) := c_0 + c_1 x + \cdots + c_d x^d \bmod q$ be a polynomial of degree d modulo q, with coefficients $c_i \in \mathbb{Z}_q$, where q is prime and $c_d \neq 0 \bmod q$.

The *roots* of g are the elements $a \in \mathbb{Z}_q$ such that $g(a) = 0 \bmod q$. For example, 2 is a root of the degree-three polynomial $g(x) = x^3 + 2x^2 - 1 \bmod 3$ because $2^3 + 2 \cdot 2^2 - 1 = 15 = 0 \bmod 3$.

We will need the following well-known theorem. Since I happen to know a beautiful proof, we'll prove it :). (Notice that it is important in the theorem that $c_d \neq 0$, since the zero polynomial $g(x) = 0$ has q roots modulo q. But, by definition, the zero polynomial does not have non-negative degree—sometimes we actually define its degree to be $-\infty$.)

Theorem 2.1. *If g is a polynomial with degree $d \geq 0$ modulo a prime q, then g has at most d roots.*

Our proof will use the following claim. In fact, the *claim* works for composite q as well (though the *theorem* itself is false for composite q, as the example $2x^2 + 2x \bmod 4$ shows).

Claim 2.2. *If $a \in \mathbb{Z}_q$ is a root of a polynomial g modulo q with degree $d \geq 1$, then $g(x) = (x-a)g'(x) \bmod q$, where $g'(x) \bmod q$ is a (non-zero) polynomial with degree $d-1$.*

Proof. Let $h(x) := g(x) - g(a)$. Since $g(a) = 0 \bmod q$ by definition, we must have $g(x) = h(x) \bmod q$. But,

$$h(x) = c_d(x^d - a^d) + c_{d-1}(x^{d-1} - a^{d-1}) + \cdots + c_1(x-a) \ .$$

Now, notice that $x^r - a^r = (x-a)(x^{r-1} + ax^{r-2} + a^2x^{r-3} + \cdots + a^{r-2}x + a^{r-1})$. Define $p_{r,a}(x) := x^{r-1} + ax^{r-2} + a^2x^{r-3} + \cdots + a^{r-2}x + a^{r-1}$. It follows that

$$\begin{aligned} h(x) &= c_d(x-a)p_{d,a}(x) + c_{d-1}(x-a)p_{d-1,a}(x) + \cdots + c_2(x-a)p_{2,a}(x) + c_1(x-a) \\ &= (x-a) \cdot (c_d p_{d,a}(x) + c_{d-1}p_{d-1,a}(x) + \cdots + c_2 p_{2,a}(x) + c_1) \ . \end{aligned}$$

One can then easily check that the polynomial $g'(x) := c_d p_{d,a}(x) + \cdots + c_2 p_{2,a}(x) + c_1$ has degree $d-1$, and since $g(x) = h(x) \bmod q$, we can write $g(x) = (x-a)g'(x) \bmod q$, as needed. □

Proof of Theorem 2.1. We can now prove the theorem by induction on the degree d. For our base case, notice that the theorem is trivially true for degree-zero polynomials, since by definition these are non-zero constant functions (e.g., $g(x) = 2 \bmod 3$), which can never be zero. (If taking the base case to be the constant functions makes you uncomfortable, you can instead use the degree one polynomials as the base case.)

Now, assume for induction that any polynomial $g'(x) \bmod q$ with degree $d-1$ has at most $d-1$ roots, and let $g(x) \bmod q$ be a polynomial with degree d. If g has no roots, then we are done. Otherwise, let $a \in \mathbb{Z}_q$ be a root of $g(x) \bmod q$. By Claim 2.2, we can write

$$g(x) = (x-a)g'(x) \bmod q \ ,$$

where $g'(x)$ has degree $d-1$.

Suppose that $a' \in \mathbb{Z}_q$ with $a \neq a' \bmod q$ is some other root of $g(x)$, so that $g(x) = (a'-a)g'(a') \bmod q$. Since q is prime and $a - a' \neq 0 \bmod q$, we must have $g'(a') = 0 \bmod q$. In other words, every root of g is either a or one of the at most $d-1$ roots of g'. (a could *also* be a root of g', which is fine.) So, g has at most d roots, as needed. □

Corollary 2.3. *Let $g(x)$ and $h(x)$ be two polynomials modulo q, each with degree at most d. Suppose that $g(a_1) = h(a_1) \bmod q, \ldots, g(a_{d+1}) = h(a_{d+1}) \bmod q$ for* distinct *$a_1, \ldots, a_{d+1} \in \mathbb{Z}_q$. Then, $g(x) = h(x) \bmod q$.*

Proof. Notice that $p(x) := g(x) - h(x) \bmod q$ is a polynomial with degree at most d, with $a_1, \ldots, a_{d+1} \in \mathbb{Z}_q$ as distinct roots. This is a contradiction *unless* p is the zero polynomial (i.e. $p(x) = 0 \bmod q$ for all x), i.e., unless $g = h \bmod q$. □

The result that we will actually use directly is the following. This tells us that *any* $d+1$ equations of the form $g(a_1) = b_1 \bmod q, \ldots, g(a_{d+1}) = b_{d+1} \bmod q$ for distinct $a_i \in \mathbb{Z}_q$ and any $b_i \in \mathbb{Z}_q$ determine a unique polynomial g with degree at most d modulo q and that this polynomial can be found efficiently, given the a_i, b_i.

The algorithm for computing this polynomial is called *Lagrange interpolation*. (Notice that it is crucial that the a_i are distinct. The result is of course false if, e.g., $a_1 = a_2 = \cdots = a_{d+1}$.)

Theorem 2.4. *For any prime q, any* distinct $a_1, \ldots, a_{d+1} \in \mathbb{Z}_q$, *and any (not necessarily distinct)* $b_1, \ldots, b_{d+1} \in \mathbb{Z}_q$, *there exists a* unique *polynomial $g(x)$ mod q with degree at most d such that $g(a_i) = b_i \bmod q$ for all i.*

Furthermore, there is an efficient algorithm that takes as input q and $(a_1, b_1), \ldots, (a_{d+1}, b_{d+1})$ and outputs g.

Proof. Corollary 2.3 already tells us that, if such a g exists, it must be unique. So, we only need to show that such a g does in fact exist, and how to find such a g efficiently. We do so via *Lagrange interpolation*.

Specifically, suppose that we have polynomials $p_1, \ldots, p_{d+1}$ such that (1) $p_i(a_i) = 1 \bmod q$ for all i; and (2) $p_i(a_j) = 0 \bmod q$ for all $i \neq j$. Then, clearly, we can take $g(x) = b_1 p_1(x) + b_2 p_2(x) + \cdots + b_{d+1} p_{d+1}(x) \bmod q$, and we will have $g(a_i) = b_i$ as needed.

It remains to show that these polynomials p_i exist. Indeed, let $h_i(x) := \prod_{j \neq i}(x - a_j) \bmod q$. By definition, $h_i(a_j) = 0$ for all $i \neq j$, and $h_i(a_i) = \prod_{j \neq i}(a_i - a_j) \bmod q$. Since q is prime and the a_i are distinct, we must have that $h_i(a_i)$ is non-zero, and therefore there must exist some inverse $\alpha_i := h_i(a_i)^{-1} \bmod q$. Then, we can take $p_i(x) := \alpha_i \cdot h_i(x) \bmod q$, which clearly satisfies the desired properties. □

(You might have noticed that the above proof is *very* similar to the proof of the Chinese Remainder Theorem. This is not a coincidence. It turns out that both theorems are really special cases of a more general theorem—the Chinese Remainder Theorem for integral domains. To see the connection, notice that the equation $g(a_i) = b_i$ is equivalent to the equation $g(x) = b_i \bmod (x - a_i)$.)

3 Shamir's secret-sharing scheme

We will now see Adi Shamir's beautiful t-out-of-k secret-sharing scheme [Sha79], which is very natural and is used in many different constructions in cryptography and computer science more broadly. It works as follows for any prime $q > k$ and for secrets $s \in \mathbb{Z}_q$.

- $\mathsf{Share}(s)$: Sample a uniformly random polynomial h over $\mathbb{Z}_q$ with degree at most $d = t - 1$ and coefficients in $\mathbb{Z}_q$ satisfying $h(0) = s$, i.e., $h(x) := s + h_1 x + h_2 x^2 + \cdots + h_{t-1} x^{t-1}$, where $h_i \sim \mathbb{Z}_q$. Set $\sigma_i := h(i) \bmod q$ for $i = 1, \ldots, k$.[1]
- $\mathsf{Reconstruct}((i_1, \sigma_1 := h(i_1) \bmod q), \ldots, (i_t, \sigma_t := h(i_t) \bmod q))$: Use Theorem 2.4 to find the unique polynomial h with degree at most $t - 1$ satisfying $h(i_j) = \sigma_j \bmod q$ for all j. Output $h(0)$ (i.e., the constant term of the polynomial).

It is immediate from Theorem 2.1 that this scheme is correct and can be implemented efficiently. But, why is it secure?

Well, for fixed distinct $i_1, \ldots, i_{t-1} \in \mathbb{Z}_q$ with $i_j \neq 0$ and $\sigma_1, \ldots, \sigma_{t-1}$, consider the set $P^* := \{h \ : \ \deg(h) \leq t - 1, \ h(i_1) = \sigma_1 \bmod q, \ldots, h(i_t) = \sigma_t \bmod q\}$ of polynomials with degree at most $t-1$ and with $h(i_j) = \sigma_j \bmod q$ for all j. Then, we claim that for every $s \in \mathbb{Z}_q$ there exists a unique

[1] It is not strictly necessary to set the secret to be the constant term $h(0)$. One could instead, for example, set the secret to be the coefficient of x^{t-1}, or one could set the secret to be, say, $h(k+1)$. This is simply a particularly convenient choice.

$h^* \in P^*$ such that $h^*(0) = s \bmod q$. Notice that this implies the result, since it implies that for any $s \in \mathbb{Z}_q$,

$$\Pr_{(\sigma'_1,\ldots,\sigma'_k)\leftarrow\mathsf{Share}(s)}[\forall j \in \{1,\ldots,t-1\},\ \sigma'_{i_j} = \sigma_j] = 1/q^{t-1}\ ,$$

independent of the secret s.

Indeed, Theorem 2.1 immediately implies that there is a unique $h^* \in P^*$ such that $h^*(0) = s \bmod q$. So, our scheme is secure (and, we have elegantly proven both correctness and security using the same theorem).

3.1 The Vandermonde matrix

Shamir's secret-sharing scheme can be written succinctly in matrix notation. In particular, let

$$\boldsymbol{A}_{k,t} := \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 4 & \cdots & 2^{t-1} \bmod q \\ 1 & 3 & 9 & \cdots & 3^{t-1} \bmod q \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & k \bmod q & k^2 \bmod q & \cdots & k^{t-1} \bmod q \end{pmatrix} \in \mathbb{Z}_q^{k\times t}\ .$$

Matrices of this form are called Vandermonde matrices (or sometimes just "the Vandermonde matrix"). If we view the secret $s \in \mathbb{Z}_q$ and the randomness $h_1,\ldots,h_t \in \mathbb{Z}_q$ sampled by the Share algorithm as a vector

$$\boldsymbol{h} := \begin{pmatrix} s \\ h_1 \\ h_2 \\ \vdots \\ h_{t-1} \end{pmatrix} \in \mathbb{Z}_q^t\ ,$$

and similarly view the resulting shares

$$\boldsymbol{r} := \begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_k \end{pmatrix} \in \mathbb{Z}_q^k$$

as a vector, then we see that

$$\boldsymbol{r} = \boldsymbol{A}_{k,t}\boldsymbol{h} \bmod q\ .$$

This in particular means that Shamir's scheme is *linear*. So, if we have shares $\sigma_{0,1},\ldots,\sigma_{0,k} \in \mathbb{Z}_q$ of some secret s_0 and shares $\sigma_{1,1},\ldots,\sigma_{1,k} \in \mathbb{Z}_q$, then the shares $\sigma_{+,1} := \sigma_{0,1}+\sigma_{1,1} \bmod q,\ldots,\sigma_{+,k} := \sigma_{0,k}+\sigma_{1,k} \bmod q$ are shares of the secret $s_+ := s_0 + s_1 \bmod q$. This can be very useful for applications.

In fact, *any* full-rank matrix $\boldsymbol{A} \in \mathbb{Z}_q^{k\times t}$ yields a linear secret-sharing scheme. However, Shamir's scheme was discovered first and is arguably the most elegant such scheme. (It also has additional properties that make it extremely useful, though we will likely not see those in this course.)

4 Secret sharing for more general access structures

The notion of t-out-of-k secret sharing that we studied above is also known as *threshold* secret sharing, since a group of parties can recover the secret if and only if the size of the group is larger than some threshold t. One can generalize threshold secret sharing to a much larger class of primitives. For example, suppose that s is the code necessary to launch a nuclear missile. Suppose we want to share it between, say, the President, the Vice President, all of the members of Congress, and the Secretary of Defense in such a way that

1. the President can reconstruct s if she is joined by *either* (1) the Vice President; (2) half the members of Congress; or (3) the Secretary of Defense; and
2. the Vice President, all the members of Congress, and the Secretary of Defense can together reconstruct s; but
3. no smaller group can "learn any information about s."

This scheme cannot be described as a t-out-of-k secret-sharing scheme. However, we can still build it!

To define this formally, we need to first define an *access structure*. An access structure $\mathcal{S}$ is simply a collection of sets of the parties, which specifies which subsets of the parties can recover the message. Intuitively, any set of parties in the access structure $\mathcal{S}$ should be able to recover the secret, and any set that is *not* in $\mathcal{S}$ should be unable to learn *anything* about S. Here is the formal definition.

Definition 4.1. *For any access structure $\mathcal{S}$ (consisting of subsets of parties), a $\mathcal{S}$-secret-sharing scheme with secret space $\mathcal{M}$ is a pair of algorithms* Share *and* $\mathsf{Reconstruct}$ *with the following properties.*

1. **(Correctness.)** *For any secret $s \in \mathcal{M}$ and any $T = \{i_1, \ldots, i_\ell\} \in \mathcal{S}$,*

$$\Pr_{(\sigma_1,\ldots,\sigma_k)\leftarrow\mathsf{Share}(s)}[\mathsf{Reconstruct}((i_1,\sigma_{i_1}),\ldots,(i_\ell,\sigma_{i_\ell})) = s] = 1 \ .$$

   (In other words, if the $\mathsf{Reconstruct}$ *algorithm is given any shares corresponding to a set in the access structure, it can always reconstruct the secret.)*

2. **(Security.)** *For any two secrets $s, s' \in \mathcal{S}$ and any $T = \{i_1, \ldots, i_\ell\} \notin \mathcal{S}$, the distributions $(\sigma_{i_1}, \ldots, \sigma_{i_\ell})$ and $(\sigma'_{i_1}, \ldots, \sigma'_{i_\ell})$ are identical, where $(\sigma_1, \ldots, \sigma_k) \leftarrow \mathsf{Share}(s)$ and $(\sigma'_1, \ldots, \sigma'_k) \leftarrow \mathsf{Share}(s')$. Equivalently, for any $\beta_1, \ldots, \beta_\ell$,*

$$\Pr_{(\sigma_1,\ldots,\sigma_k)\leftarrow\mathsf{Share}(s)}[\forall j,\ \sigma_{i_j} = \beta_j] = \Pr_{(\sigma'_1,\ldots,\sigma'_k)\leftarrow\mathsf{Share}(s')}[\forall j,\ \sigma'_{i_j} = \beta_j] \ .$$

   (In other words, the joint distribution of any shares corresponding to a set not in the access structure are independent of the secret s.)

Notice, however, that some access structures $\mathcal{S}$ are clearly impossible to achieve. E.g., if $T \in \mathcal{S}$ and $T \subseteq T'$, then it must be the case that $T' \in \mathcal{S}$ as well. Otherwise, the above definition would say that the shares in T are sufficient to recover the secret, but the larger collection of shares in T' are completely independent of the secret. This cannot possibly be true. E.g., it cannot possibly

be the case that the President can recover the codes when she is joined by half the members of Congress but that she is unable to do so when she is joined by all members of Congress.

We call an access structure $\mathcal{S}$ *monotone* if whenever $T \in \mathcal{S}$, we also have $T' \in \mathcal{S}$ for any superset T' of T. Perhaps surprisingly, *any* monotone access structure can be achieved. However, we pay a lot in efficiency. In particular, some access structures seem to require shares whose size is exponential in k. (Notice that Shamir's secret-sharing scheme only required shares of size $O(\log k)$.) It is still an open problem to determine the optimal share size of a $\mathcal{S}$-secret-sharing scheme for arbitrary monotone $\mathcal{S}$.

Theorem 4.2. *For* any monotone *access structure $\mathcal{S}$, there exists a $\mathcal{S}$-secret-sharing scheme.*

We will not see the proof of the above theorem in this course, but it is actually not too difficult. It follows from these two results.

1. Given a $\mathcal{S}_0$-secret-sharing scheme for an access and a $\mathcal{S}_1'$-secret-sharing-scheme, it is possible to construct a $\mathcal{S}_{\mathsf{AND}}$-secret-sharing scheme and a $\mathcal{S}_{\mathsf{OR}}$-secret-sharing scheme, where $\mathcal{S}_{\mathsf{AND}} := \mathcal{S}_0 \cap \mathcal{S}_1$ and $\mathcal{S}_{\mathsf{OR}} := \mathcal{S}_0 \cup \mathcal{S}_1$.

2. Any monotone access structure can be constructed by taking intersections and unions of "simple" access structure (e.g., access structures consisting of all sets containing one party).

References

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. 4