Cryptography and Security Protocols **IST-UL, 2024/2025**

Secret sharing

Lecturer: João Ribeiro

Introduction

Looking a bit ahead, we will be focusing on the problem of *secure computation*. In the 2-party version of this problem, Alice and Bob each have private inputs x and y, respectively, and want to jointly compute a function $f(x, y)$ of their inputs. For example, x and y could be bank balances, and $f(x, y) = 1$ if and only if $x \geq y$. Of course, without privacy constraints Alice and Bob could just share their inputs with the other person, and then each person computes $f(x, y)$ on their own. However, we intuitively require that after running the protocol, Alice learns nothing about y besides what can be inferred from $(x, f(x, y))$, and Bob learns nothing about x besides what can be inferred from $(y, f(x, y))$. This makes the problem much more interesting.

A key tool for secure multiparty computation is *secret sharing*. The usual story goes like this: in a powerful, warmongering country there is a secret code s for launching a barrage of nuclear weapons. We would like to distribute this code among a group of n high-ranking generals. Of course, we could just give s to each general. But then we run the risk of some trigger-happy general unilaterally deciding to launch the missiles. Therefore, we would like to share s among the generals so that s is only revealed if a sufficiently large number t of generals simultaneously agree to launch the missiles, and remains completely hidden otherwise.

Secret sharing is a central tool in cryptography, broadly speaking. It is also a fascinating mathematical notion, with many elegant connections to algebra, combinatorics, geometry, and coding and information theory. And the cherry on top is that we can actually construct secret sharing schemes without relying on computational assumptions!

1 A simple secret sharing scheme

Let's start with one of the simplest versions of this problem. There is a 1-bit secret $s \in \{0, 1\}$ that we wish to secret-share among two parties, so that the two parties can jointly reconstruct s, but a single party learns nothing about s. This can be done by sampling $S_1 \leftarrow \{0, 1\}$ and setting $S_2 = S_1 \oplus s$. Then, the first party's share is S_1, and the second party's share is S_2.

Clearly, the two parties can reconstruct s by computing $S_1 \oplus S_2 = s$. On the other hand, each of S_1 and S_2 is uniformly distributed, independently of s. So, for example, knowledge of S_1 reveals nothing about s (and likewise for s_2). We call this a 2-*out-of*-2 secret sharing scheme.

This technique is called *additive secret sharing*. It can be easily extended to any number n of parties. To share a secret $s \in \{0, 1\}$ among n parties, sample shares $S_1, \ldots, S_{n-1}$ uniformly at random from

$\{0,1\}$, and set $S_n = s \oplus S_1 \oplus \cdots \oplus S_{n-1}$. The i-th party's share is S_i. This is an n-out-of-n secret sharing scheme, in the sense that the n shares reconstruct the secret as $S_1 \oplus S_2 \oplus \cdots \oplus S_n = s$, while any subset of $n-1$ shares is uniformly distributed over $\{0,1\}^{n-1}$, independently of s.

So, this is already good. But what if we want to design, say, a "3-out-of-5" secret sharing scheme?

2 Formalizing threshold secret sharing

Before we proceed to study more general constructions of secret sharing schemes, let's first formally define this notion. Intuitively, we want a "t-out-of-n" secret sharing scheme to have the properties that (1) any subset of t shares allows perfect reconstruction of the secret, while (2) any subset of $t-1$ shares reveals nothing about the secret. We call these *threshold* secret sharing schemes.

Definition 1 (t-out-of-n secret sharing scheme) *For positive integers n and $t \leq n$, a* t-out-of-n secret sharing scheme *is a tuple of algorithms* $(\mathsf{Share}, \mathsf{Rec})$ *with*

- Share *a randomized algorithm that on input a secret s (from some set of secrets $\mathcal{S}$) outputs shares $S_1, \ldots, S_n$, with S_i the share of the i-th party.*
- Rec *a deterministic algorithm that on input a subset $T \subseteq \{1, \ldots, n\}$ of size $|T| \geq t$ and the corresponding shares $(S_i)_{i \in T}$ outputs $s' \in \mathcal{S}$.*

We require the following reconstruction and privacy properties:

- ***Reconstruction:*** *For any subset $T \subseteq \{1, \ldots, n\}$ of size $|T| \geq t$ we have*

$$\Pr[\mathsf{Rec}(T, \mathsf{Share}(s)_T) = s] = 1,$$

  where $\mathsf{Share}(s)_T$ denotes the shares of s corresponding to parties in T.

- ***Privacy:*** *For any subset $T \subseteq \{1, \ldots, n\}$ of size $|T| < t$ the distribution of $\mathsf{Share}(s)_T$ is independent of s. In other words, for any two distinct secrets $s \neq s'$, the shares $\mathsf{Share}(s)_T$ and $\mathsf{Share}(s')_T$ are identically distributed.*

This notion was first introduced independently by Blakley [Bla79] and Shamir [Sha79].

3 Threshold secret sharing from polynomials

We saw above how to construct n-out-of-n secret sharing schemes, for any number of parties n. We will now work through a very elegant construction of t-out-of-n secret sharing schemes for any n and $t \leq n$, due to Shamir [Sha79].

Shamir's construction is based on the following simple, but very powerful, statement that one may call the **degree mantra**[1]: *A nonzero polynomial of degree t with coefficients in some field (for example, the reals, or $\mathbb{Z}_p$ for p prime) has at most t roots in that field.*

Feel free to ignore the term "field" in the mantra above if you have not seen it before. You have certainly seen the version of the degree mantra for polynomials over the reals (a line has at most one root, a parabola has at most two roots, etc). All that matters for our discussion here is that the degree mantra also holds over $\mathbb{Z}_p$, for any prime p.[2]

It is really surprising how far we can go via clever applications of the degree mantra. I leave some pointers about this for further reading below, if you are curious.

On to Shamir's secret sharing scheme. Fix a number of parties n, a threshold $t \leq n$, and a prime $p > n$. Fix also n distinct "evaluation points" $\alpha_1, \ldots, \alpha_n \in \mathbb{Z}_p \setminus \{0\}$ (this is why we need $p > n$). To share a secret $s \in \mathbb{Z}_p$, we proceed as follows:

1. For $j \in \{1, \ldots, t-1\}$ sample coefficients $c_i \leftarrow \mathbb{Z}_p$ and define the polynomial
$$f(x) = s + \sum_{j=1}^{t-1} c_j x^j \pmod{p}.$$
Note that f has degree at most $t-1$ and that $f(0) = s$. In fact, f is a uniformly random polynomial of degree at most $t-1$ conditioned on $f(0) = s$.
2. The i-th share is $S_i = f(\alpha_i)$.

We will now show that this is a t-out-of-n secret sharing scheme. Along the way we will clarify how reconstruction works.

Claim 1 (Reconstruction) *Fix an arbitrary subset $T \subseteq \{1, \ldots, n\}$ of size $|T| \geq t$. Then, we can efficiently recover $f(0) = s$ from the shares $(y_i = f(\alpha_i))_{i \in T}$.*

Proof: Without loss of generality we may assume that $|T| = t$. Since f has degree $t-1$, we can perfectly reconstruct from its evaluation on t distinct points through "Lagrange interpolation". More precisely, consider the polynomial
$$g(x) = \sum_{i \in T} \prod_{j \in T \setminus \{i\}} \frac{x - \alpha_j}{\alpha_i - \alpha_j} \cdot y_i \pmod{p},$$
where we interpret a fraction a/b as ab^{-1} with b^{-1} the inverse of b modulo p (i.e., $b \cdot b^{-1} = 1 \pmod{p}$). Note that g has degree at most $t-1$ and that $g(\alpha_i) = y_i$ for all $i \in T$ by construction.

This implies that the polynomial $f - g$ has degree at most $t-1$ and t roots at $(\alpha_i)_{i \in T}$. By the degree mantra, this implies that $f - g$ is zero everywhere, and so $g(0) = f(0) = s$. This implicitly defines the reconstruction procedure for Shamir's secret sharing. ∎

[1] The credit for this terminology is not mine. See, e.g., this excellent book on coding theory by Guruswami, Rudra, and Sudan.

[2] For readers familiar with these concepts, all that we say here applies equally well over any finite field.

Claim 2 (Privacy) *Fix an arbitrary subset $T \subseteq \{1, \ldots, n\}$ of size $|T| < t$. Then, the distribution of $(y_i = f(\alpha_i))_{i \in T}$ is independent of s.*

Proof: Without loss of generality we may assume that $|T| = t - 1$. Note that for each $s' \in \mathbb{Z}_p$ there is exactly one polynomial g of degree at most $t - 1$ such that $g(0) = s'$ and $g(\alpha_i) = y_i$ for all $i \in T$. This follows from Lagrange interpolation as above, because a degree-$(t-1)$ polynomial is completely determined by its evaluation on t distinct points.

Fix any secret s'. Then, the above means that the probability of observing shares $(y_i)_{i \in T}$ is exactly $p^{-(t-1)}$ (i.e., it is the probability of sampling the unique consistent polynomial). This does not depend on s'. ■

3.1 Some nice properties of Shamir secret sharing

Shamir's secret sharing is one of my favorite cryptographic constructions. It is extremely elegant, mathematically rich, and highly impactful.

Linearity. Shamir's secret sharing is *linear*, in the following sense: if we have shares $S_1, \ldots, S_n$ of a secret s and shares $S'_1, \ldots, S'_n$ of another secret s', then $(S_1 + S'_1 \pmod p, \ldots, S_n + S'_n \pmod p)$ are shares of $s + s' \pmod p$. Note that such a transformation can be performed *locally* by the parties, simply by having each party add their two shares.

Linear secret sharing schemes are an extremely important sub-family of secret sharing schemes, in particular for secure computation.

Robustness. Consider the scenario where some of the parties lie and give wrong shares for the reconstruction. Shamir's secret sharing is quite robust in this setting, allowing us to "correct" these corrupted shares (so long as there are not too many) and recover the correct secret. For readers with some knowledge of coding theory, an illuminating way to see this is to note that Shamir's secret sharing is actually a Reed-Solomon code in disguise.

4 General access structures

Shamir's secret sharing allows us to design t-out-of-n secret sharing schemes, for any number of parties n and threshold t. However, what if we would like to design "non-threshold" secret sharing schemes? For example, consider a scenario where we want to share a secret among n parties so that reconstructing the secret requires shares from some t parties, *plus the share of a special party* $P^\star$. Or a scenario where parties are divided into two disjoint subcommittees of sizes n_1 and n_2, respectively, and reconstructing the secret requires at least t_1 parties from the first subcommittee *and* at least t_2 parties from the second subcommittee?

More generally, call a collection Γ of subsets of $\{1, \ldots, n\}$ *monotone* if for any $A \in \Gamma$ and any B such that $A \subseteq B$ we also have $B \in \Gamma$. We say that a secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ *realizes* Γ if

the subsets in Γ are exactly the authorized subsets for this scheme. More precisely,

- if $T \in \Gamma$, then $\Pr[\mathsf{Rec}(T, \mathsf{Share}(s)_T) = s] = 1$ for any secret s;
- if $T \notin \Gamma$, then for any two distinct secrets $s \neq s'$ the distributions $\mathsf{Share}(s)_T$ and $\mathsf{Share}(s')_T$ are identically distributed.

Note that a secret sharing can only realize monotone collections Γ, because if a subset of T allows reconstructing s, then the shares corresponding to T also allow for reconstructing s. We also call such monotone Γ the *access structure* of this secret sharing scheme. For example, the access structure for a t-out-of-n secret sharing scheme is $\Gamma = \{T \subseteq \{1, \ldots, n\} : |T| \geq t\}$.

Can we realize every access structure Γ? It turns out that the answer is "yes!", if you are willing to forego efficient sharing and reconstruction. We briefly sketch here a particularly simple approach for proving this, due to Benaloh and Leichter [BL90].

The key idea is the following:

- It is easy to come up with secret sharing schemes realizing the access structures $\Gamma^{(i)} = \{T \subseteq \{1, \ldots, n\} : i \in T\}$ for $i \in \{1, \ldots, n\}$ and the trivial access structures $\varnothing$ and $2^{\{1,\ldots,n\}}$, where 2^S denotes the powerset of a set S.
- Given secret sharing schemes realizing access structures Γ_0 and Γ_1, is also not difficult to come up with secret sharing schemes realizing $\Gamma_0 \cap \Gamma_1$ and $\Gamma_0 \cup \Gamma_1$.
- Every access structure Γ can be written by a "tree of conjunctions and disjunctions" of the base access structures $\Gamma^{(1)}, \ldots, \Gamma^{(n)}$.

  More precisely, we may identify subsets $T \subseteq \{1, \ldots, n\}$ with binary vectors $x_T \in \{0,1\}^n$ such that $(x_T)_i = 1$ if and only if $i \in T$, and we may also identify Γ with the boolean function $f_\Gamma : \{0,1\}^n \to \{0,1\}$ such that $f_\Gamma(x_T) = 1$ if and only if $T \in \Gamma$. Then, every f_Γ can be computed by a *monotone formula*.[3]

4.1 Limitations of this approach and a nice open problem

The approach of Benaloh and Leichter above shows that we can realize any access structure. However, the resulting secret sharing scheme may require extremely long shares. In the worst case the length of a share may be $2^{n-o(n)}$, where n is the number of parties, even if we only want to share a 1-bit secret.

This raises the natural question of whether there exist access structures that can only be realized by secret sharing schemes with exponentially long shares. We still know surprisingly little about this!

Using a connection to matroid theory, Csirmaz [Csi97] showed that there exist access structures requiring shares of bitlength $\Omega(n/\log n)$. This is still the best lower bound on the share size for

[3] There are various ways of defining monotone formulas. For example, we may define them inductively: the "base" monotone formulas are the constants 0 and 1 and the functions $f(x) = x_i$ for $i \in \{1, \ldots, n\}$, and, if f_0 and f_1 are monotone formulas, then $f_0 \wedge f_1$ and $f_0 \vee f_1$ are also monotone formulas.

general access structures we know today. Regarding upper bounds, the $2^{n-o(n)}$ upper bound was long conjectured to be optimal. This was surprisingly disproved by Liu and Vaikuntanathan in fairly recent work [LV18]. A line of work has chipped away at the upper bound, and the state-of-the-art is now 1.5^n [AN21], still very far away from the best known $\Omega(n/\log n)$ lower bound!

For the special case of *linear* secret sharing schemes, we do know exponential lower bounds on the share size, via the equivalence to objects called *monotone span programs* [RPRC16, PR17].

5 Further reading

If you are interested in learning more about secret sharing, Amos Beimel's survey is an excellent starting point.

Behind Shamir's secret sharing lies a very simple but ingenious application of the degree mantra. More clever applications of the degree mantra and related properties of polynomials fall into what is usually called the *polynomial method*. One of my favorite applications of the polynomial method is Zeev Dvir's extremely slick proof of the finite field Kakeya conjecture. If you want to learn more about this, see this blog post by Terence Tao.

References

[AN21] Benny Applebaum and Oded Nir. Upslices, downslices, and secret-sharing with complexity of 1.5^n. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 627–655, Cham, 2021. Springer International Publishing.

[BL90] Josh Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, pages 27–35, New York, NY, 1990. Springer New York.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979.

[Csi97] László Csirmaz. The size of a share must be large. *J. Cryptol.*, 10(4):223–231, 1997.

[LV18] Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 699–708, New York, NY, USA, 2018. Association for Computing Machinery.

[PR17] Toniann Pitassi and Robert Robere. Strongly exponential lower bounds for monotone computation. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 1246–1255, New York, NY, USA, 2017. Association for Computing Machinery.

[RPRC16] Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential lower bounds for monotone span programs. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 406–415, 2016.

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