Week 7+

Nadav Kohen

August 6, 2025

This brings us to the end of the prework!! You've done it!!!

I will include a few more readings here that would be good to read before we meet in person. I also encourage everyone to take some time to review any material from previous weeks that felt unclear or unresolved.

For the papers listed below, I don't intend to overwhelm you with a bunch of complicated readings to get through, but rather I think it would be productive for you to have some more exposure in advance of our in-person week. As such, I don't expect you to go through these papers as thoroughly as in past weeks, and I encourage you to first read for a higher-level understanding of the general ideas without necessarily getting to bogged down in too many details (unless you'd like to).

Reading 1. *Read about a commonly-used method of organizing complicated security proofs (which will come up in-person) based on using many indistinguishably small transitions in: Victor Shoup - Sequences of Games: A Tool for Taming Complexity in Security Proofs.*

Reading 2. *Read about MuSig2 in:*
Jonas Nick, Tim Ruffing, and Yannick Seurin - MuSig2: Simple Two-Round Schnorr Multi-Signatures.

And lastly, I've included below some exposition on the background for Shamir secret sharing, which is the mathematical primitive underlying most threshold cryptography, after which I've included the FROST paper as a reading.

I will now provide some exposition on polynomial interpolation. All polynomials in this discussion can be thought of as having coefficients (and hence values) in $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. This is the mathematical tool underlying most distributed key generation and multi-party computation protocols. At a high level, the idea is to use the fact that a polynomial, $f(x) = y$, of degree $t - 1$ is uniquely determined by any t of its points so that we can build t-of-n threshold cryptosystems using polynomials to store secrets. For example, a line (degree 1 polynomial) is determined by any two points and a quadratic polynomial is determined by any three of its points. In this context, rather than individual parties holding entire secrets, they only hold secret shares (or shards), which are actually points on a polynomial (that is, pairs (x, y) satisfying $f(x) = y$). Typically, the secret value s is stored as $f(0) = s$. We can think of representing f as a set of n points $(x_1, y_1), \ldots, (x_n, y_n)$ that each satisfy $f(x_i) = y_i$. If anyone knows at least t of these points (where f is a polynomial of degree $t - 1$), then they can reconstruct all of f (as we will discuss below), and in particular, they can compute $f(0) = s$.

One of the reasons to use polynomials (which is known as Shamir secret sharing) in place of, for example, just having everyone XOR secret shares together, is that if we have two shared secrets stored in $f(x) = y$ and $g(x) = y$, then we can add these two secrets together by simply adding corresponding shares together since $(f + g)(x) = f(x) + g(x)$ (where $f + g$ is itself a new polynomial). Similarly, if c is some constant scalar, then $(cf)(x) = c(f(x))$ so that multiplying each share by c results in multiplying the polynomial by c. The fact that we can perform addition and scalar multiplication by simply performing operations on secret shares implies that we can compute values such as Schnorr signatures for shared keys

without needing to construct the keys!

Polynomial interpolation is the actual process by which one computes a polynomial, f, of degree $t-1$ from the input data of t distinct points, $\{(x_i, y_i)\}_{i=1}^{t}$ satisfying $f(x_i) = y_i$. There are multiple methods of performing interpolation, but one popular method relevant to FROST is Lagrange interpolation. The key idea of Lagrange interpolation is that, given distinct $x_1, x_2, \ldots, x_t$, the polynomial,

$$L_j(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_t)}{(x_j-x_1)(x_j-x_2)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_t)} = \prod_{i\neq j}\frac{x-x_i}{x_j-x_i},$$

has the special property that $L_j(x_i)$ is equal to 0 if $i \neq j$ and is equal to 1 if $i = j$. Additionally, $L_j(x)$ is a degree $t-1$ polynomial. Therefore, if I wish to build a polynomial that agrees with f on t given points, I can simply take

$$L(x) = y_1L_1(x) + y_2L_2(x) + \cdots + y_tL_t(x) = \sum_{j=1}^{t} y_tL_t(x) = \sum_{j=1}^{t}\prod_{i\neq j} y_j\frac{x-x_i}{x_j-x_i}.$$

This degree $t-1$ polynomial has the property that $L(x_i) = y_i$ for all $i \in \{1, \ldots, t\}$!

Now that we know how to construct a polynomial that goes through a given t points, we wish to conclude that $L(x) = f(x)$, so we must show that if two degree-at-most $t-1$ polynomials agree on t points, then they must be equal.

Lemma 1. *If f is a non-zero polynomial such that $f(a) = 0$ (in which case we call a a "root" of f), then $f(x)$ can be written as $(x-a)\cdot g(x)$ for some polynomial $g(x)$ of degree one less than f.*

Proof. Expand $f(x+a)$ until it is of the form $c_0 + c_1x + c_2x^2 + \cdots + c_{t-1}x^{t-1}$. Since $f(0+a) = f(a) = 0$, we have that $c_0 = c_0 + c_10 + c_20^2 + \cdots + c_{t-1}0^{t-1} = 0$. Therefore, $f(x+a) = c_1x + c_2x^2 + \cdots + c_{t-1}x^{t-1} = x(c_1 + c_2x + \cdots + c_{t-1}x^{t-2})$, and finally by a change

of variables, $f(x) = (x-a)(c_1 + c_2(x-a) + \cdots + c_{t-1}(x-a)^{t-2})$. □

Lemma 2. *If $f(x)$ is a non-zero degree $t-1$ polynomial, it has at most $t-1$ roots.*

Proof. We prove this lemma by induction on t. When $t = 1$ or $t = 2$, the result is clear. Now suppose f has degree t and the result holds for all polynomials of degree less than t. If f has no roots then the result is true. If a is a root of f, then $f(x) = (x-a)g(x)$ for some polynomial $g(x)$ of degree $t-1$. Therefore, by our inductive hypothesis, $g(x)$ has at most $t-1$ roots so that $f(x)$ has at most t roots, as desired. □

Theorem 3. *If $L(x)$ and $f(x)$ are two polynomials of degree $t-1$ that agree on t distinct values, then $L(x) = f(x)$.*

Proof. Consider they polynomial $(L-f)(x) = L(x) - f(x)$, which is of degree at most $t-1$. This polynomial must have at least t distinct roots, because if $L(x_i) = f(x_i)$, then $(L-f)(x_i) = 0$! Therefore, by the previous lemma, $(L-f)(x)$ must be the zero polynomial, so that $(L-f)(x) = L(x) - f(x) = 0 \Leftrightarrow L(x) = f(x)$. □

At a high level, this theorem is true because Lemma 1 shows us that polynomials can be factored as a product of $(x-a)$'s such that $f(a) = 0$, potentially ending with a factor that has no roots. Thus, a polynomial of degree $t-1$ is determined by any t of its values because the difference of any two polynomials of degree $t-1$ that share t values must be 0 as it cannot be the product of t linear factors (because the degree is bound by $t-1$).

With this background in mind, we can now read about Pedersen's DKG protocol, wherein n parties can construct a random polynomial known to none of them individually, but where each party knows a point on this polynomial. Subsequently, we can read the FROST paper, which uses this DKG as well as some other Lagrange interpolation tricks to achieve threshold Schnorr digital signatures.

Reading 3. *In preparation for reading about FROST, read about Pedersen's trustless distributed key generation protocol in the extended abstract of:*
Torben Pedersen - A Threshold Cryptosystem without a Trusted Party.

Reading 4. *Read about FROST in:*
Chelsea Komlo and Ian Goldberg - FROST: Flexible Round-Optimized Schnorr Threshold Signatures.