Zero-Knowledge Proofs

Andrew Lovelass & Owen Rusnak (Mentor: Hunter Handley)

The Ohio State University

DEPARTMENT OF MATHEMATICS

Motivation/Cool Idea

- Sometimes you want to show that you know information without giving it away
- Ex: logging in to your bank account, opening a lock, magic tricks, . . .
- In the case of having a lock, you would put in the code without anyone seeing and then show that it is unlocked.
- For more complex scenarios, we need to introduce more complexity. For this, we lean on math and trapdoor functions

Finite Fields as "Language of Computers"

- The environment we work in is a field: a place where you can do your elementary operations (e.g., addition, subtraction, multiplication, and division)
- More formally, a field is a set F with elements $0, 1 \in F$ such that:
  - F is an Abelian group under addition with identity 0 and $F^* := F \setminus \{0\}$ is an Abelian group under multiplication with identity 1
  - The distributive property holds: $a(b + c) = ab + ac$
- Popular infinite fields: Real numbers, complex numbers, & rational numbers

Building a Basic Finite Field

1. Take a prime $p \in \mathbb{Z}$
2. Let $\mathbb{F}_p := \mathbb{Z}/p\mathbb{Z} = \{0, 1, \ldots, p-1\}$
3. Define addition and multiplication by $a + b := (a + b) \bmod p$ and $ab := (ab) \bmod p$
4. Notice for $a \in \mathbb{F}_p^*$, Euclid's algorithm gives $x, y \in \mathbb{Z}$ with $ax + py = 1$, so $a^{-1} = x \bmod p$, giving multiplicative inverses
5. For $\#F = p^n$ with $n > 1$, take $f \in \mathbb{F}_p[x]$ with $n = \deg f$ and $f(a) \neq 0$ for $a \in \mathbb{F}_p$
6. Letting α be a root of f, the set

$$\mathbb{F}_{p^n} := \mathbb{F}_p[x] \bmod f = \left\{ \sum_{i=0}^{n} a_i \cdot \alpha^i : a_i \in \mathbb{F}_p \right\}$$

is a finite field of size p^n

Computing in Finite Fields

- Computations can be tedious and time consuming. Using brute force to solve $a^n \bmod p$, where $a \in \mathbb{F}_p^*$, and $n \in \mathbb{N}$, you need $n - 1$ operations to find the solution
- The following strategies allow us to speed up the process

Fermat's Little Theorem

Fix a prime p. For any $a \in \mathbb{F}_p$, also $a^p = a$ as elements of $\mathbb{F}_p$. Equivalently, for $a \in \mathbb{Z}$, also $a^p \equiv a \bmod p$.

Repeated Squaring

- Solving $a^{2^n} \bmod p$ takes $2^n - 1$ operations via brute force
- But if we know $t := a^{2^{j-1}}$, then $a^{2^j} = t^2$
- Inducting on $j = 1, \ldots, n$ we only use n operations
- For $a^s \bmod p$ with $s \neq 2^n$, use binary to write $s = \sum_0^\ell s_i \cdot 2^i$ with $0 \leq s_i \leq 1$ so

$$a^s = a^{\sum s_i 2^i} = \prod_0^\ell \left(a^{2^i}\right)^{s_i}$$

The Discrete Log Assumption

- We now introduce the idea of a trapdoor function
- Suppose want to find $m \in \mathbb{Z}$ such that $a = b^m$, where a and $b \in \mathbb{F}_p^*$
- Trying to use brute force to solve the discrete logarithm takes forever for p large
- However, if given b and m, one can easily compute $a = b^m$, making this a "trapdoor"– easy one way, hard the other
- **The Assumption:** it is computationally unfeasible to find m with just a and b

The Schnorr Protocol

- We have a verifier and prover agree on a field $\mathbb{F}_q$ and elements $g, h \in \mathbb{F}_q$
- The prover wants to prove knowledge of x such that $h = g^x$
- There are three different steps that happen:
- *The Commitment:* The prover commits to a value and sends it to the verifier
- *The Challenge:* The verifier sends a challenge, depending on the commitment, back to the prover
- *The Response:* The prover uses their commitment, the challenge, and their secret to compute a value that is sent to the verifier, who accepts if correct

The Schnorr Protocol

	Alice	Public	Bob
Knows:	Secret x	$g, h \in \mathbb{F}_q$	
1) Commitment:	Random $t \in \mathbb{F}_q$, computes $r := g^t$	$\xrightarrow{r}$	
2) Challenge:		$\xleftarrow{c}$	Random $c \in \mathbb{F}_q$
3) Response:	$z := t + xc$	$\xrightarrow{z}$	Accepts if $g^z = rh^c$

- To be zero-knowledge, however, it must hold the following three properties
- **Completeness:** If a prover supplies a correct answer in their response, then an honest verifier will always accept
- Using the Schnorr protocol, one can see that

$$g^z = g^{t+xc} = g^t(g^{xc}) = r(h^c),$$

showing that an honest prover will have their response accepted
- **Soundness:** If an answer is accepted, then the prover *must* know the secret
- According to the discrete log assumption, a malicious prover can guess challenge value correctly probability $1/q$
- To make this "soundness error" as close to zero as possible we use large primes to construct our finite fields
- **Zero-Knowledge:** The verifier, upon accepting a correct response, has learned nothing about the provers secret
- This property protects the prover from a malicious verifier, who wants to extract the secret
- If we can go backwards and simulate a transcript (r, c, z) without x that looks just as random a real transcript, we are good!
- Choose $c, z \in \mathbb{F}_q$ randomly and compute $r := g^z h^{-c}$ to get (r, c, z). This satisfies the protocol, but has the same probabilistic distribution as a legit (r, c, z), whilst not using x at all

Improvements

Pederson Commitments

- Pederson Commitments allow a prover to efficiently hide multiple things at once or show they correctly performed arithmetic
- Via Pederson, the verifier can see that the prover did the math correctly without getting the primes, so then they can pay and the prover can reveal
- Suppose the verifier gave prover $N \in \mathbb{Z}$ that is a product of two primes, but they do not know *which* primes– maybe N is huge

Pederson Protocol for $p_1 p_2 = N$

	Alice	Public	Bob
Knows:	Secret p_1, p_2	$g_i, h \in \mathbb{F}_q$ $N \in \mathbb{Z}$	
1) Commitment:	Random $r, t_i \in \mathbb{F}_q$, computes $\rho_1 := g_3^{t_1 t_2}$ $\rho_2 := g_1^{t_1} g_2^{t_2} g_3^{t_1 p_2 + t_2 p_1} h^{t_3}$ $C := g_1^{p_1} g_2^{p_2} g_3^{p_1 p_2} h^r$	$\xrightarrow{\rho_1, \rho_2, C}$	
2) Challenge:		$\xleftarrow{c}$	Random $c \in \mathbb{F}_q$
3) Response:	$z_1 := t_1 + cp_1$ $z_2 := t_2 + cp_2$ $z_3 := t_3 + cr$	$\xrightarrow{z_i}$	Accepts if $g_3^{z_1 z_2}(g_1^{z_1} g_2^{z_2} h^{z_3})^c = \rho_2^c \rho_1 C^{c^2}$

Fiat-Shamir Non-Interactive Zero Knowledge (NIZK)

- Schnorr requires a prover and verifier to send multiple messages back and forth, taking needless time and energy
- An improvement to this is a non-interactive proof
- **HASH function:** A function that transforms the input data into a fixed-length size where collisions are unlikely to occur and even small changes in input will produce a different output. Thence the output looks random.

Fiat-Shamir non-interactive Schnorr (NIZK)

	Alice	Public	Bob
Knows:	Secret x	$g, h \in \mathbb{F}_q$ Hash fn HASH(X)	
1) Commitment: and challenge	Random $t \in \mathbb{F}_q$, computes $r := g^t$ $c = \text{Hash}(g, h, q, r)$ $z = t + xc$	$\xrightarrow{r, c, z}$	Accepts if $c = \text{Hash}(g, h, q, r)$ and $g^z = rh^c$

References

RJ McEliece, *Finite fields for scientists and engineers*, Springer, 1987

V Shoup, *A computational introduction to number theory and algebra*, Cambridge University Press, 2005

ZKDocs, Trail of Bits, url: https://www.zkdocs.com/