

Sigma Protocols

September 3, 2024

Distributed Lab

Plan

Introduction

Schnorr IP

Σ -Protocols

Examples

Coding Time!

Introduction

Recap on Interactive Proofs

- **Interactive proofs** allows practically prover \mathcal{P} to convince the verifier \mathcal{V} that some statement is true.
- **Soundness** ensures that the prover cannot cheat the verifier, while **zero-knowledge** that the verifier learns nothing about the witness.
- **Argument of knowledge** ensures that the prover also “knows” the witness (that is, exists some extractor \mathcal{E} that, acting as an admin, can extract the witness).
- If verifier’s messages are random values, the protocol is **public-coin**.
- Any public-coin protocol can be transformed into a **non-interactive** proof using **Fiat-Shamir heuristic**.

Announcement

Today, we will build and code our first non-interactive proof system using the Fiat-Shamir heuristic based on **Sigma protocols**!

Motivation

In many cases, we need to prove relatively trivial statements without revealing the witness:

- “I know the discrete log of a point $P \in E(\mathbb{F}_p)$ ”.
- “I know the representation of a point $P \in E(\mathbb{F}_p)$, that is $(\alpha, \beta) \in \mathbb{Z}_q^2$ such that $P = [\alpha]G + [\beta]H$ ”.
- “I know the e th modular root w of $x \in \mathbb{Z}_N^\times$ (that is, $w^e = x$)”. [For \$e = 2\$, see previous lecture.](#)
- “I know that $(P, Q, R) \in E(\mathbb{F}_p)^3$ is a Diffie-Hellman triplet”.

Σ -protocols are also fundamentally similar to Bulletproofs!

Note

Everything that has a natural “homomorphic”/discrete-log-like structure can be proven using Sigma (Σ) protocols!

Schnorr IP

Problem Statement

Suppose \mathbb{G} is a cyclic group of order q with a generator g . Then, the relation and language being considered are:

$$\mathcal{R} = \{(u, \alpha) \in \mathbb{G} \times \mathbb{Z}_q : u = g^\alpha\}, \quad \mathcal{L}_{\mathcal{R}} = \{u \in \mathbb{G} : \exists \alpha \in \mathbb{Z}_q : u = g^\alpha\}$$

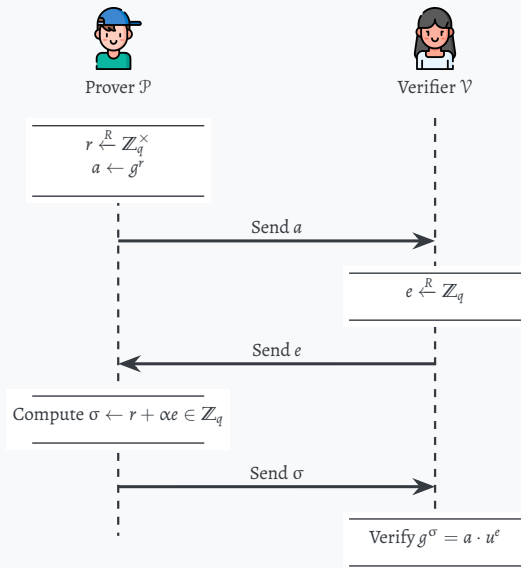
Problem #1

\mathcal{P} wants to convince \mathcal{V} that it knows the discrete log of $u \in \mathcal{L}_{\mathcal{R}}$. That is, he knows α such that $(u, \alpha) \in \mathcal{R}$.

Problem #2

Why cannot we simply send α ? Because we do not want to reveal the witness! That is why we need a zero-knowledge non-interactive argument of knowledge (zk-NARK).

Protocol Flow



Protocol Flow

Definition

The Schnorr interactive identification protocol $\Pi_{\text{Sch}} = (\text{Gen}, \mathcal{P}, \mathcal{V})$ with a generation function Gen and prover \mathcal{P} and verifier \mathcal{V} is defined as:

- $\text{Gen}(1^\lambda)$: Take $\alpha \xleftarrow{R} \mathbb{Z}_q$ and $u \leftarrow g^\alpha$. **Output:** verification key $\text{vk} := u$, and secret key $\text{sk} := \alpha$.
- The protocol between $(\mathcal{P}, \mathcal{V})$ is run as follows:
 - \mathcal{P} computes $r \leftarrow \mathbb{Z}_q^\times$, $a \leftarrow g^r$ and sends a to \mathcal{V} .
 - \mathcal{V} sends a random challenge $e \xleftarrow{R} \mathbb{Z}_q$ to \mathcal{P} .
 - \mathcal{P} computes $\sigma \leftarrow r + \alpha e \in \mathbb{Z}_q$ and sends σ to \mathcal{V} .
 - \mathcal{V} accepts if $g^\sigma = a \cdot u^e$, otherwise it rejects.

Question

\mathcal{V} only sends a random scalar to \mathcal{P} . How to turn this into a non-interactive proof?

Applying Fiat-Shamir Transformation

Reminder

Suppose prover had messages (m_1, m_2, \dots, m_n) before verifier sends a challenge c . If x is a public statement, it suffices to choose $c \leftarrow H(x, m_1, \dots, m_n)$ without any interaction.

Definition (The Schnorr non-interactive identification protocol)

Define $\Gamma_{\text{Sch}} := (\text{Gen}, \text{Prove}, \text{Verify})$:

- **Gen**(1^λ): **Output** $\alpha \xleftarrow{R} \mathbb{Z}_q$ and $u \leftarrow g^\alpha$.
- **Prove**: on input (u, α) do:
 - Compute $r \leftarrow \mathbb{Z}_q^\times, a \leftarrow g^r$.
 - Compute challenge $e \leftarrow H(u, a)$.
 - Computes $\sigma \leftarrow r + \alpha e$. Output (a, σ) .
- **Verify**: accept iff $g^\sigma = a \cdot u^e$.

Schnorr's Signature Scheme

It easy to turn the non-interactive identification protocol into a signature scheme! Simply regard (u, m) as a public statement with a message m !

Definition

The Schnorr Signature Scheme is $\Sigma_{\text{Sch}} = (\text{Gen}, \text{Sign}, \text{Verify})$, where:

- $\text{Gen}(1^\lambda)$: **Output** $\alpha \xleftarrow{R} \mathbb{Z}_q$ and $u \leftarrow g^\alpha$.
- $\text{Sign}(m, \text{sk})$: The signer computes $r \leftarrow \mathbb{Z}_q^\times$, $a \leftarrow g^r$, $e \leftarrow H(u, m, a)$, $\sigma \leftarrow r + \alpha e$ and outputs (a, σ) .
- $\text{Verify}((a, \sigma), m, \text{pk})$: The verifier checks if $g^\sigma = a \cdot u^e$ for $e \leftarrow H(u, m, a)$.

Note: In **green** we marked the only difference between the identification and signature protocols.

Σ -Protocols

Generalization

Now, can we generalize the Schnorr protocol to any relation \mathcal{R} ?

Well, not for any, but for a large class of relations called **Sigma protocols**!

Definition

Let $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ be an effective relation. A **Sigma protocol** for \mathcal{R} is an interactive protocol $(\mathcal{P}, \mathcal{V})$ that satisfies the following properties:

- In the beginning, \mathcal{P} computes a **commitment** a and sends it to \mathcal{V} .
- \mathcal{V} chooses a random **challenge** $c \in \mathcal{C}$ from the challenge space \mathcal{C} and sends it to \mathcal{P} .
- Upon receiving c , \mathcal{P} computes the response z and sends it to \mathcal{V} .
- \mathcal{V} outputs either accept or reject based on the **conversation** (a, c, z) .

Definition

(a, c, z) is an **accepting conversation** if \mathcal{V} outputs accept on this tuple.

Why Σ ?

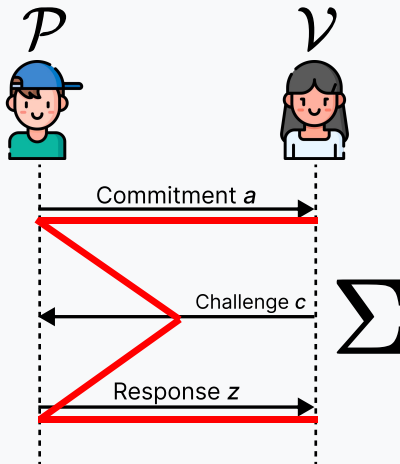


Figure: Why Σ -protocols are called so.

Special Soundness

Definition (Special Soundness)

Let $(\mathcal{P}, \mathcal{V})$ be a Σ -protocol for $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$. We say that $(\mathcal{P}, \mathcal{V})$ is **special sound** if there exists a witness extractor \mathcal{E} such that, given statement $x \in \mathcal{X}$ and two accepting conversations (a, c, z) and (a, c', z') (where $c \neq c'$)^a, the extractor can always efficiently compute the witness w such that $(x, w) \in \mathcal{R}$.

^aNotice that initial commitments in both conversations are the same!

Example

The Schnorr protocol is special sound because, given two accepting conversations (a, e, σ) and (a, e', σ') , we can compute the witness α . You can verify that $\alpha = \Delta\sigma/\Delta e$ for $\Delta\sigma = \sigma' - \sigma$ and $\Delta e = e' - e$ suffices.

Examples

Okamoto's Protocol

Again, let \mathbb{G} be a cyclic group of prime order q with a generator $g \in \mathbb{G}$ and let $h \in \mathbb{G}$ an arbitrary group element.

Definition

For $u \in \mathbb{G}$, a **representation** relative to g and h is a pair $(\alpha, \beta) \in \mathbb{Z}_q \times \mathbb{Z}_q$ such that $u = g^\alpha h^\beta$.

Remark

Notice that for the given u there are exactly q representations relative to g and h . Indeed, $\forall \beta \in \mathbb{Z}_q \exists! \alpha \in \mathbb{Z}_q : g^\alpha = uh^{-\beta}$.

Question

How do we actually prove that \mathcal{P} knows the representation of u ?

$$\mathcal{R} = \left\{ (u, (\alpha, \beta)) \in \mathbb{G} \times \mathbb{Z}_q^2 : u = g^\alpha h^\beta \right\}$$

Okamoto's Protocol Flow

Definition (Okamoto's Identification Protocol)

Okamoto's Protocol consists of two algorithms: $(\mathcal{P}, \mathcal{V})$, where the prover is assumed to know $(u, (\alpha, \beta)) \in \mathcal{R}$ defined above. The protocol is defined as follows:

1. \mathcal{P} computes $\alpha_r \xleftarrow{R} \mathbb{Z}_q$, $\beta_r \xleftarrow{R} \mathbb{Z}_q$, $u_r \leftarrow g^{\alpha_r} h^{\beta_r}$ and sends commitment u_r to \mathcal{V} .
2. \mathcal{V} samples the challenge $c \xleftarrow{R} \mathbb{Z}_q$ and sends c to \mathcal{P} .
3. \mathcal{P} computes $\alpha_z \leftarrow \alpha_r + \alpha c$, $\beta_z \leftarrow \beta_r + \beta c$ and sends $\mathbf{z} = (\alpha_z, \beta_z)$.
4. \mathcal{V} checks whether $g^{\alpha_z} h^{\beta_z} = u_r u^c$ and accepts or rejects the proof.

Announcement

We will code the non-interactive Okamoto's protocol in the next section! Stay tuned!

Okamoto's Protocol Correctness

Theorem

Okamoto's Protocol is a Σ -protocol for the relation \mathcal{R} which is Honest-Verifier Zero-Knowledge (HVZK).

Part of the proof. Again, let us show *correctness* and *special soundness* without honest-verifier zero-knowledge properties.

Completeness. Suppose indeed that $(u, (\alpha, \beta)) \in \mathcal{R}$. Then, the verification condition can be written as follows:

$$g^{\alpha_z} h^{\beta_z} = g^{\alpha_r + \alpha_c} h^{\beta_r + \beta_c} = g^{\alpha_r} g^{\alpha_c} h^{\beta_r} h^{\beta_c} = \underbrace{(g^{\alpha_r} h^{\beta_r})}_{=u_r} \cdot \underbrace{(g^{\alpha_c} h^{\beta_c})}_{=u} = u_r u^c$$

Okamoto's Protocol Special Soundness

Special Soundness. Suppose we are given two accepting conversations: $(u_r, c, (\alpha_z, \beta_z))$ and $(u_r, c', (\alpha'_z, \beta'_z))$ and we want to construct an extractor \mathcal{E} which would give us a witness (α, β) . In this case, we have the following holding:

$$g^{\alpha_z} h^{\beta_z} = u_r u^c, \quad g^{\alpha'_z} h^{\beta'_z} = u_r u^{c'}$$

We can divide the former by the latter to obtain:

$$g^{\alpha_z - \alpha'_z} h^{\beta_z - \beta'_z} = u^{c - c'} = g^{\alpha(c - c')} h^{\beta(c - c')},$$

from which the extractor \mathcal{E} can efficiently compute witness as follows:
 $\alpha \leftarrow (\alpha_z - \alpha'_z) / (c - c')$ and $\beta \leftarrow (\beta_z - \beta'_z) / (c - c')$.

Diffie-Hellman Triplets

Suppose we are given the cyclic group \mathbb{G} of prime order q and generator $g \in \mathbb{G}$.

Definition

A triplet $(u, v, w) \in \mathbb{G}^3$ is a **Diffie-Hellman triplet** if $\exists \alpha, \beta \in \mathbb{Z}_q : u = g^\alpha, v = g^\beta, w = g^{\alpha\beta}$.

Alternative DH-triple Definition

(u, v, w) is a DH-triplet iff $\exists \beta \in \mathbb{Z}_q : v = g^\beta, w = u^\beta$.

Now, this makes it easier to define the relation \mathcal{R} for the Chaum-Pedersen protocol:

$$\mathcal{R} = \{((u, v, w), \beta) \in \mathbb{G}^3 \times \mathbb{Z}_q : v = g^\beta \wedge w = u^\beta\}$$

Chaum-Pedersen Protocol

Definition (Chaum-Pedersen Protocol)

Chaum-Pedersen Protocol consists of two algorithms: $(\mathcal{P}, \mathcal{V})$, where the prover is assumed to know $(\beta, (u, v, w)) \in \mathcal{R}$ defined above. The protocol is defined as follows:

1. \mathcal{P} computes $\beta_r \xleftarrow{R} \mathbb{Z}_q$, $v_r \xleftarrow{R} g^{\beta_r}$, $w_r \leftarrow u^{\beta_r}$ and sends (u_r, w_r) to \mathcal{V} .
2. \mathcal{V} samples the challenge $c \xleftarrow{R} \mathbb{Z}_q$ and sends c to \mathcal{P} .
3. \mathcal{P} computes $\beta_z \leftarrow \beta_r + \beta c$ and sends β_z to \mathcal{V} .
4. \mathcal{V} checks whether two conditions hold: $g^{\beta_z} = v_r v^c$ and $u^{\beta_z} = w_r w^c$, and accepts or rejects the proof accordingly.

Theorem

Chaum-Pedersen Protocol is a Σ -protocol for the relation \mathcal{R} which is Honest-Verifier Zero-Knowledge (HVZK).

Homomorphism

Let us formulate the core objects that we will use in this section:

- $(\mathbb{H}, +)$ is a finite abelian input group.
- (\mathbb{T}, \times) is a finite abelian output group.
- $\psi : \mathbb{H} \rightarrow \mathbb{T}$ is a hard-to-invert homomorphism.
- $\mathcal{F} = \text{Hom}(\mathbb{H}, \mathbb{T})$ is a set of all homomorphisms from \mathbb{H} to \mathbb{T} .

Reminder

Homomorphism $\psi : \mathbb{H} \rightarrow \mathbb{T}$ is a function, satisfying the following property:

$$\forall h_1, h_2 \in \mathbb{H} : \psi(h_1 + h_2) = \psi(h_1)\psi(h_2)$$

Note

If between input and output we have an easy-to-compute and hard-to-invert homomorphism, we can use Sigma protocols to prove pre-images of this homomorphism!

Problem Statement

Define the following relation:

$$\mathcal{R} = \{((t, \psi), h) \in (\mathbb{T} \times \mathcal{F}) \times \mathbb{H} : \psi(h) = t\}$$

\mathcal{P} wants to convince \mathcal{V} that he knows witness h to the statement (t, ψ) .

Example

Now, why does this generalize the previous protocols? Well, let us consider all previous examples:

- **Schnorr Protocol:** Here we have $\mathbb{H} = \mathbb{Z}_q$, $\mathbb{T} = \mathbb{G}$, and $\psi : \mathbb{Z}_q \rightarrow \mathbb{G}$ is defined as $\psi(\alpha) = g^\alpha$. Moreover, here ψ is an isomorphism!
- **Okamoto Protocol:** Here we have $\mathbb{H} = \mathbb{Z}_q^2$, $\mathbb{T} = \mathbb{G}$, and $\psi : \mathbb{Z}_q^2 \rightarrow \mathbb{G}$ is defined as $\psi(\alpha, \beta) = g^\alpha h^\beta$.
- **Chaum-Pedersen Protocol:** Here we have $\mathbb{H} = \mathbb{Z}_q$, $\mathbb{T} = \mathbb{G}^2$, and $\psi : \mathbb{Z}_q \rightarrow \mathbb{G}^2$ is defined as $\psi(\beta) = (g^\beta, u^\beta)$.

Sigma Protocol

Definition (Sigma Protocol for the pre-image of a homomorphism)

The protocol consists of two algorithms: $(\mathcal{P}, \mathcal{V})$, where the prover is assumed to know the witness $h \in \mathbb{H}$ defined above. The protocol is defined as follows:

1. \mathcal{P} computes $h_r \xleftarrow{R} \mathbb{H}$, $t_r \leftarrow \psi(h_r) \in \mathbb{T}$ and sends t_r to the verifier \mathcal{V} .
2. \mathcal{V} samples the challenge $c \xleftarrow{R} \mathcal{C} \subset \mathbb{Z}$ from the challenge space and sends c to \mathcal{P} .
3. \mathcal{P} computes $h_z \leftarrow h_r + h \cdot c$ and sends h_z to \mathcal{V} .
4. \mathcal{V} checks whether $\psi(h_z) = t_r t^c$, and accepts or rejects the proof.

Theorem

Such protocol is a Σ -protocol for the relation \mathcal{R} which is Honest-Verifier Zero-Knowledge (HVZK).

Combining Σ -Protocols

One of the features (which we are not going to delve into) is the ability to combine Σ -protocols to prove more complex statements. Namely,

- Given two relations \mathcal{R}_0 and \mathcal{R}_1 , we can prove that the prover knows witnesses for both relations.
- Given two relations \mathcal{R}_0 and \mathcal{R}_1 , we can prove that the prover knows a witness for at least one of the relations.

Example

\mathcal{P} can prove that he either knows the discrete log of u or the representation of u relative to g and h . Moreover, \mathcal{V} does not know which of the two statements \mathcal{P} is proving.

Coding Time!

Methodology

Reminder

Suppose prover had messages (m_1, m_2, \dots, m_n) before verifier sends a challenge c . If x is a public statement, it suffices to choose $c \leftarrow H(x, m_1, \dots, m_n)$ without any interaction.

Let us turn **Okamoto's Protocol** into a non-interactive proof using the Fiat-Shamir heuristic!

Reminder: Okamoto's Identification Protocol

1. \mathcal{P} computes $\alpha_r \xleftarrow{R} \mathbb{Z}_q$, $\beta_r \xleftarrow{R} \mathbb{Z}_q$, $u_r \leftarrow g^{\alpha_r} h^{\beta_r}$ and sends commitment u_r to \mathcal{V} .
2. \mathcal{V} samples the challenge $c \xleftarrow{R} \mathbb{Z}_q$ and sends c to \mathcal{P} .
3. \mathcal{P} computes $\alpha_z \leftarrow \alpha_r + \alpha c$, $\beta_z \leftarrow \beta_r + \beta c$ and sends $\mathbf{z} = (\alpha_z, \beta_z)$.
4. \mathcal{V} checks whether $g^{\alpha_z} h^{\beta_z} = u_r u^c$ and accepts or rejects the proof.

Non-Interactive Okamoto Protocol

Okamoto's Non-Interactive Identification Protocol

Now, we apply the *Fiat-Shamir Transformation*.

- $\text{Gen}(1^\lambda)$: On input $(u, (\alpha, \beta)) \in \mathbb{G} \times \mathbb{Z}_q^2$,
 1. Sample $\alpha_r, \beta_r \xleftarrow{R} \mathbb{Z}_q$ and compute $u_r \leftarrow g^{\alpha_r} h^{\beta_r}$.
 2. Using the hash function $H : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{C}$, compute $c \leftarrow H(u, u_r)$.
 3. Compute $\alpha_z \leftarrow \alpha_r + \alpha c, \beta_z \leftarrow \beta_r + \beta c$ and publish (u_r, α_z, β_z) as a proof π .
- Verify: Upon receiving statement u and a proof $\pi = (u_r, \alpha_z, \beta_z)$, the verifier:
 1. Recomputes the challenge c using the hash function.
 2. Accepts if and only if $g^{\alpha_z} h^{\beta_z} = u_r u^c$.

<https://github.com/ZKDL-Camp/lecture-7-sigma>

Thank you for your attention!