0.1 Schnorr's Identification Protocol

To better illustrate the Fiat-Shamir Transformation in practice, let us consider one of the most basic Sigma protocols: non-interactive Schnorr Identification Protocol. It is a simple and elegant protocol that allows one party to prove to another party that it knows a discrete logarithm of a given element. It is also quite straightforward to generalize it to a signature scheme.

Let us formalize it using theory from **??**. Suppose $\mathbb{G}$ is a cyclic group of order q with a generator g. Then, the relation and language being considered are:

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

Now, suppose prover $\mathcal{P}$ has a valid statement and a witness $(u, \alpha) \in \mathcal{R}$ and he wants to convince the verifier $\mathcal{V}$ that he knows the witness α to the public statement u (that is, we are building the proof of knowledge). Well, the easiest way how to proceed is simply giving α to $\mathcal{V}$, but this is obviously not what we want. Instead, the Schnorr protocol allows $\mathcal{P}$ to prove the knowledge of α without revealing it.

First, let us start with the interactive version of the protocol.

Definition 0.1. The Schnorr interactive identification protocol $\Pi_{\mathsf{Sch}} = (\mathsf{Gen}, \mathcal{P}, \mathcal{V})$ with a generation function Gen and prover $\mathcal{P}$ and verifier $\mathcal{V}$ is defined as follows:

- $\mathsf{Gen}(1^\lambda)$: As with most public-key cryptosystems, we take $\alpha \xleftarrow{R} \mathbb{Z}_q$ and $u \leftarrow g^\alpha$. We output the *verification key* as $\mathsf{vk} := u$, and the *secret key* as $\mathsf{sk} := \alpha$.
- 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.

This protocol is illustrated in Figure 1.

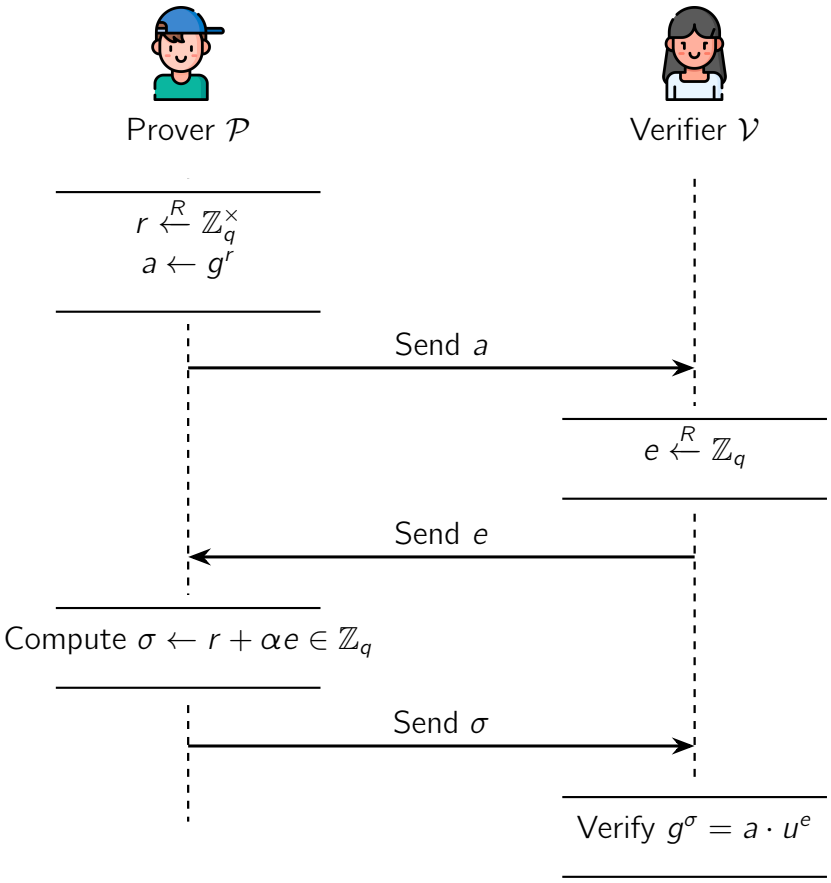

Figure 1: The interactive Schnorr protocol between prover $\mathcal{P}$ and verifier $\mathcal{V}$ for proof of knowledge of discrete logarithm relation.

Definition 0.2. An interaction between $\mathcal{P}$ and $\mathcal{V}$ produces the so-called **conversation** $(a, e, \sigma) \in \mathbb{G} \times \mathbb{Z}_q \times \mathbb{Z}_q$. We call such a conversation an **accepting conversation** if $\mathcal{V}$ accepts the proof.

Example. In case of a Schnorr protocol, the accepting conversation is such that $g^\sigma = a \cdot u^e$.

Now, one can prove the following theorem.

Theorem 0.3. The Schnorr protocol Π_{Sch} is complete, sound, and (honest verifier) zero-knowledge proof of knowledge.

Proof. We are not going to prove the zero-knowledge and soundness properly, but completeness and proof of knowledge are quite straightforward to show.

- **Completeness.** Just observe that $g^\sigma = g^{r+\alpha e} = g^r (g^\alpha)^e = a \cdot u^e$.
- **Proof of Knowledge.** To prove that the protocol is a proof of knowledge, we need to construct an extractor $\mathcal{E}^{\mathcal{P}}$. We construct it as follows:
  1. Extractor runs the prover and gets a, e, and σ as a response.
  2. Extractor rewinds back to the verifier's challenge step, generates a new challenge $e' \xleftarrow{R} \mathbb{Z}_q$ and gets new prover's response σ' (for the same prover's randomness r).

3. Extractor outputs the witness $\alpha \leftarrow (\sigma - \sigma')(e - e')^{-1}$.

The reason why this works is following: notice that $g^{\sigma} = a \cdot u^{e}, g^{\sigma'} = a \cdot u^{e'}$. Therefore, by dividing former by latter, we obtain $g^{\sigma-\sigma'} = u^{e-e'} = g^{\alpha(e-e')}$. It immediately follows that $\alpha = (\sigma - \sigma')(e - e')^{-1}$.

Remark. Before considering how to make such protocol non-interactive correctly, suppose that we instead do the following: after interaction with the verifier, the prover publishes the conversation as a proof of knowledge. Would that be a valid non-interactive proof? In other words, can we convince the independent observer of the interaction that the prover knows the witness? The answer is no (and it is generally so for any interactive protocol). The reason why is that the prover can first sample randomly $e, \sigma \xleftarrow{R} \mathbb{Z}_q$, compute $a \leftarrow g^{\sigma}/u^{e}$ and simply publish (a, e, σ) as a proof. This is a valid conversation since $g^{\sigma} = a \cdot u^{e} = (g^{\sigma}/u^{e}) \cdot u^{e}$ and thus the observer would be convinced that the prover knows the witness. However, the prover might not know the witness at all!
Therefore, either (1) the prover needs to get a challenge e **before** he commits to the value σ, or (2) challenge must be randomized. Otherwise, he can precompute σ and publish it as a proof (or simply make a deal with the verifier to fool the observer).

Now, notice that the provided protocol is a public-coin protocol. Therefore, we can apply the Fiat-Shamir transformation to make it non-interactive. Suppose we have a random oracle $\mathcal{O}_{\mathsf{R}} : \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q$:

1. The prover $\mathcal{P}$ computes $r \leftarrow \mathbb{Z}_q^{\times}$, $a \leftarrow g^{r}$ and sends a to the Fiat-Shamir Channel.
2. The Fiat-Shamir channel responds with the challenge $e \leftarrow \mathcal{O}_{\mathsf{R}}(u, a)$.
3. The prover $\mathcal{P}$ computes $\sigma \leftarrow r + \alpha e$ and sends σ to the Fiat-Shamir Channel.
4. The Fiat-Shamir channel outputs the proof $\pi = (a, e, \sigma)$, which the verifier can check via previously mentioned equation $g^{\sigma} = a \cdot u^{e}$.

Now, notice that e might not be included in the proof since the verifier can compute it by himself. Therefore, the final proof π can be reduced to $(a, \sigma) \in \mathbb{G} \times \mathbb{Z}_q$ and its computation does not need any interaction with the verifier. Moreover, it is still complete, sound, and proof of knowledge due to the Fiat-Shamir transformation. It is also (not easy to prove) zero-knowledge.

0.2 Schnorr's Signature Scheme

Now, turning the Schnorr's Identification Protocol into a signature scheme is quite straightforward. The only modification to the non-interactive proof described in the previous section is that we include the message $m \in \mathcal{M}$ instead of our statement $u \in \mathbb{G}$ in the computation of the challenge e. Additionally, suppose we use the hash function H as a random oracle from the previous section. Now, let us give a formal definition.

Definition 0.4. The Schnorr Signature Scheme is $\Sigma_{\mathsf{Sch}} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, where:

- $\mathsf{Gen}(1^\lambda)$: We take $\alpha \xleftarrow{R} \mathbb{Z}_q$ and $u \leftarrow g^\alpha$. The *public key* is $\mathsf{pk} := u$, while the *secret key* as $\mathsf{sk} := \alpha$.
- $\mathsf{Sign}(m, \mathsf{sk})$: The signer computes $r \leftarrow \mathbb{Z}_q^\times, a \leftarrow g^r, e \leftarrow H(m, a), \sigma \leftarrow r + \alpha e$ and outputs the signature (a, σ).
- $\mathsf{Verify}((a, \sigma), m, \mathsf{pk})$: The verifier checks if $g^\sigma = a \cdot u^e$ for $e \leftarrow H(m, a)$.

Remark. Typically, one also uses a so-called "*key-prefixed*" variant of the scheme, where the challenge e is computed as $e \leftarrow H(\mathsf{pk}, m, a)$ for a random oracle $H : \mathbb{G} \times \mathcal{M} \times \mathbb{G} \to \mathbb{Z}_q$. It was argued that such variant has a better multi-user security bound than the classical one.

0.3 Sigma Protocols

Now, the Schnorr Protocol is just one of the many examples of a so-called **Sigma Protocol**. Sigma protocols are a class of interactive proof systems that are used to prove the knowledge of a witness to a statement. They are quite general and can be used to prove the knowledge of a witness to any effective relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$, where $\mathcal{X}$ is the set of public statements and $\mathcal{W}$ is the set of witnesses. Let us define them formally.

Definition 0.5. 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 transcript (a, c, z).

Remark. The name "Sigma" protocol comes from the fact that the "shape" of the message flow vaguely resembles the Greek letter Σ: see Figure 2.

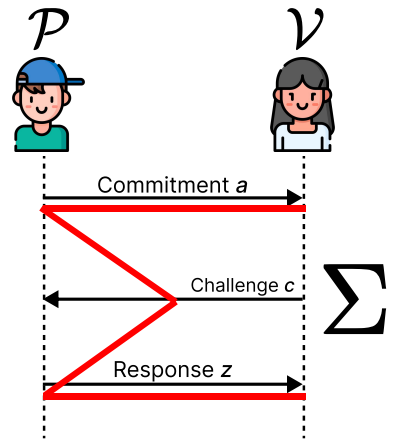

Figure 2: Sigma Protocol Illustration: the flow of messages between prover $\mathcal{P}$ and verifier $\mathcal{V}$ closely resembles the Greek letter Σ, which is marked in **red** in the Figure.

Example. In particular, for the Schnorr Protocol, the Sigma protocol is defined over the relation $\mathcal{R} \subset \mathcal{X} \times \mathcal{W}$ where:

$$\mathcal{X} = \mathbb{G},\ \mathcal{W} = \mathbb{Z}_q,\ \mathcal{R} = \{(u, \alpha) \in \mathbb{G} \times \mathbb{Z}_q : u = g^\alpha\}$$

Here, the challenge space $\mathcal{C}$ is a subset of $\mathbb{Z}_q$ (or, typically, the whole set).

Similarly to interactive protocols, Sigma protocols also have a property called *soundness*. However, there is an additional property called *special soundness* that simplifies the general notion of soundness.

Definition 0.6 (Special Soundness)**.** Let $(\mathcal{P}, \mathcal{V})$ be a Σ-protocol for $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$. We that 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 $(x, w) \in \mathcal{R}$.

[a] Notice that initial commitments in both conversations are the same!

Example. In case of the Schnorr Protocol, the special soundness property is satisfied by the extractor $\mathcal{E}$ that we have constructed in the proof of knowledge. In other words, we can extract the discrete logarithm $\alpha = \mathsf{DLog}_{\mathbb{G}}(u)$ given two accepting conversations (a, e, σ) and (a', e', σ').

Now, let us consider some more examples of Sigma protocols.

0.4 More Sigma Protocol Examples

0.4.1 Okamoto's Protocol for Representations

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 (for example, it might be yet another group generator). While considering Pedersen Commitments, you already encountered form $g^\alpha h^\beta$. Now, let us generalize this concept a bit.

Definition 0.7. 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, $\forall \beta \in \mathbb{Z}_q\, \exists! \alpha \in \mathbb{Z}_q : g^\alpha = u h^{-\beta}$.

Now, the *Okamoto's Protocol* is a Sigma protocol that allows one party to prove the knowledge of a representation of a given $u \in \mathbb{G}$ relative to g and h. In other words, we are working with the relation

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

Now, let us describe the protocol.

Definition 0.8 (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 $\mathbf{z} = (\alpha_z, \beta_z)$ to $\mathcal{V}$.
4. $\mathcal{V}$ checks whether $g^{\alpha_z} h^{\beta_z} = u_r u^c$ and accepts or rejects the proof accordingly.

Theorem 0.9. Okamoto's Protocol is a Σ-protocol for the relation $\mathcal{R}$ which is Honest-Verifier Zero-Knowledge.

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:

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

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:

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

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 $\beta \leftarrow (\beta_z - \beta'_z)/(c - c')$.

0.4.2 Chaum-Pedersen protocol for DH-triplets

As with previous examples, suppose we are given the cyclic group $\mathbb{G}$ or prime order q and generator $g \in \mathbb{G}$. Recall that *the Diffie-Hellman Triple* (or, *DH-triple*) is a triple $(g^\alpha, g^\beta, g^\gamma)$ with $\gamma = \alpha\beta$. Now, this definition is not really convenient for us, so we will reformulate the DH-triple using the proposition below.

Proposition 0.10 (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:

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

In other words, here we have a witness $\beta \in \mathbb{Z}_q$, while the statement is a triplet $(u, v, w) \in \mathbb{G}^3$. Again, we want to convert this into a Sigma protocol. We do it as follows.

Definition 0.11 (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 commitment (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 0.12. Chaum-Pedersen Protocol is a Σ-protocol for the relation $\mathcal{R}$ which is Honest-Verifier Zero-Knowledge.

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

Correctness. Again, consider the expression g^{β_z} more closely:

$$g^{\beta_z} = g^{\beta_r + \beta c} = g^{\beta_r} g^{\beta c} = \underbrace{g^{\beta_r}}_{=v_r} (\underbrace{g^{\beta}}_{=v})^c = v_r v^c$$

The similar reasoning can be applied to the second verification condition: indeed, here we have $u^{\beta_z} = u^{\beta_r}(u^{\beta})^c = w_r w^c$

Special Soundness. Suppose we are given two accepting conversations: $((u_r, w_r), c, \beta_z)$ and $((u_r, w_r), c', \beta'_z)$ and we want to construct an extractor $\mathcal{E}$ which would give us a witness β. Notice that the following equations hold:

$$g^{\beta_z} = v_r v^c,\ g^{\beta'_z} = v_r v^{c'},$$
$$u^{\beta_z} = w_r w^c,\ u^{\beta'_z} = w_r w^{c'}.$$

Divide left equations by the right ones to obtain:

$$g^{\beta_z - \beta'_z} = v^{c - c'},\ u^{\beta_z - \beta'_z} = w^{c - c'}.$$

Consider the first equation. Since $v = g^{\beta}$ we derive $(\beta_z - \beta'_z) = \beta(c - c')$, from which $\mathcal{E}$ outputs $\beta = \frac{\beta_z - \beta'_z}{c - c'}$. The same value can be extracted from the second equation.

0.5 Generalizing Sigma Protocols

Now, the most interesting part! Probably, you have noticed, that all protocols above (Schnorr, Okamoto, Chaum-Pedersen) have a similar structure. So is there any way to generalize them? The answer is yes and moreover, this done in a very elegant way.

Let $(\mathbb{H}, \oplus)$ and $(\mathbb{T}, \otimes)$ be two finite abelian groups and suppose we have some concrete homomorphism $\psi : \mathbb{H} \to \mathbb{T}$. Moreover, we require that given $t \in \mathbb{T}$, finding the pre-image of t (meaning, finding some $h \in \mathbb{H}$ such that $\psi(h) = t$) is computationally hard. Suppose $\mathcal{F}$ is a set of all homomorphisms from $\mathbb{H}$ to $\mathbb{T}$ (sometimes denoted as $\mathsf{Hom}(\mathbb{H}, \mathbb{T})$). Now, define the following relation:

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

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

Proposition 0.13. 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 $\psi : \mathbb{Z}_q \to \mathbb{G}$ 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 $\psi : \mathbb{Z}_q^2 \to \mathbb{G}$ is defined as $\psi(\alpha, \beta) = g^\alpha h^\beta$. It is also quite easy to see that ψ is a homomorphism:

  $$\psi((\alpha,\beta)+(\alpha',\beta')) = \psi(\alpha+\alpha', \beta+\beta') = g^{\alpha+\alpha'} h^{\beta+\beta'} = g^\alpha h^\beta g^{\alpha'} h^{\beta'} = \psi(\alpha,\beta)\psi(\alpha',\beta')$$

- **Chaum-Pedersen Protocol:** Here we have $\mathbb{H} = \mathbb{Z}_q$, $\mathbb{T} = \mathbb{G}^2$, and $\psi : \mathbb{Z}_q \to \mathbb{G}^2$ is defined as $\psi(\beta) = (g^\beta, u^\beta)$. Again, it is easy to see that ψ is a homomorphism.

Now, we formulate the general Sigma protocol for the relation $\mathcal{R}$ over homomorphism.

Definition 0.14 (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 $h_z \leftarrow h_r \oplus h \cdot c$ and sends h_z to $\mathcal{V}$.
4. $\mathcal{V}$ checks whether $\psi(h_z) = t_r \otimes t^c$, and accepts or rejects the proof accordingly.

0.6 Combining Sigma Protocols

Now, suppose we have the Sigma interactive protocol $(\mathcal{P}_0, \mathcal{V}_0)$ for one relation $\mathcal{R}_0 \subseteq \mathcal{W}_0 \times \mathcal{X}_0$ and another Sigma interactive protocol $(\mathcal{P}_1, \mathcal{V}_1)$ for another relation $\mathcal{R}_1 \subseteq \mathcal{W}_1 \times \mathcal{X}_1$. Now, we want to combine these two protocols into a single one. Namely, we want our prover to be able to convince the verifier that:

1. He knows the witnesses w_0, w_1 to both statements x_0, x_1.
2. He knows the witness $w \in \mathcal{W}_0 \cup \mathcal{W}_1$ to either statement x_0 or x_1.

Among two, the second one is a bit more interesting since it allows us to prove the knowledge of a witness to either of the statements. This is called the *OR-composition* of Sigma protocols.

0.6.1 The AND Sigma Protocol

Now, let the prover $\mathcal{P}$ prove the witness knowledge of the following relation:

$$\mathcal{R}_{\text{AND}} = \{((x_0, x_1), (w_0, w_1)) \in (\mathcal{X}_0 \times \mathcal{X}_1) \times (\mathcal{W}_0 \times \mathcal{W}_1) : (w_0, x_0) \in \mathcal{R}_0 \wedge (w_1, x_1) \in \mathcal{R}_1\}$$

We define the following protocol.

Definition 0.15 (The AND Sigma Protocol)**.** Define a pair of algorithms $(\mathcal{P}, \mathcal{V})$ which are run as follows:

1. The prover $\mathcal{P}$ runs $\mathcal{P}_0(w_0, x_0)$ to get commitment a_0 and runs $\mathcal{P}_1(w_1, x_1)$ to get a_1 and sends the pair $\mathbf{a} = (a_0, a_1)$ to $\mathcal{V}$.
2. The verifier computes the challenge $c \xleftarrow{R} \mathcal{C}$ and sends it to $\mathcal{P}$.
3. The prover feeds provers $\mathcal{P}_0(w_0, x_0)$ and $\mathcal{P}_1(w_1, x_1)$ with the challenge to get responses z_0 and z_1, respectively. He then sends $\mathbf{z} = (z_0, z_1)$ to $\mathcal{V}$.
4. The verifier checks whether both $\mathcal{V}_0(a_0, c, z_0)$ and $\mathcal{V}_1(a_1, c, z_1)$ pass.

However, such protocol is not very interesting since what we did essentially is just running two protocols separately: one for $(\mathcal{P}_0, \mathcal{V}_0)$, and the other for $(\mathcal{P}_1, \mathcal{V}_1)$. The only difference is that we use the single challenge for both protocols.

0.6.2 The OR Sigma Protocol

The less trivial example is the following: define the relation

$$\mathcal{R}_{\text{OR}} = \{((x_0, x_1), (w, b)) \in (\mathcal{X}_0 \times \mathcal{X}_1) \times ((\mathcal{W}_0 \cup \mathcal{W}_1) \times \{0, 1\}) : (x, w_b) \in \mathcal{R}_b\}$$

Here, the statement is x_0 and x_1, but the witness is the witness w to either x_0 or x_1, and the bit $b \in \{0, 1\}$, marking to which of the statement w belongs to. That being said, w might be from either set $\mathcal{W}_0$ or $\mathcal{W}_1$: that is why we say that $w \in \mathcal{W}_0 \cup \mathcal{W}_1$.

To make the interactive protocol work, we add one more assumption about both relations $\mathcal{R}_0$ and $\mathcal{R}_1$. Suppose that the challenge space $\mathcal{C} \subseteq \{0, 1\}^\ell$. This assumption is not very strong as typically $\mathcal{C}$ is some subspace of integers and thus decomposing some $c \in \mathcal{C}$ into the fixed-length bit representation is a trivial task.

Now, we describe the algorithm.

Definition 0.16 (The OR Sigma Protocol)**.** Define a pair of algorithms $(\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}_{\text{OR}}$ with $b^* := 1 - b$ as follows:

1. The prover chooses a random challenge $c_{b^*} \xleftarrow{R} \mathcal{C}$ and generates random commitment and response (a_{b^*}, z_{b^*}) that form a valid accepting conversation $(a_{b^*}, c_{b^*}, z_{b^*})$ (essentially, the prover runs the simulator $(a_{b^*}, z_{b^*}) \xleftarrow{R} \mathsf{Sim}_{b^*}(x_{b^*}, c_{b^*})$). Then, $\mathcal{P}$ also runs $\mathcal{P}_b(x_b, w)$ to get a valid commitment a_b and sends (a_0, a_1) to $\mathcal{V}$.
2. The verifier sends a random challenge $c \xleftarrow{R} \mathcal{C} \subseteq \{0, 1\}^\ell$.
3. The prover XORs both challenges: $c_b \leftarrow c \oplus c_{b^*}$. Then it feeds the challenge c_b to the prover $\mathcal{P}_b(x_b, w)$ to get the responses z_b ($b \in \{0, 1\}$) and sends (c_0, z_0, z_1) to $\mathcal{V}$.
4. Verifier computes $c_1 \leftarrow c \oplus c_0$ and checks that both verifications $\mathcal{V}_0(a_0, c_0, z_0)$ and $\mathcal{V}_1(a_1, c_1, z_1)$ pass.

0.7 Exercises

Exercises 1-5. In search of correct Schnorr's Identification Protocol...

You are given the protocol and five ways to implement it. Most of them lack the crucial properties. For each attempt, you need to determine whether the protocol is correct and, if not, specify which of the properties are violated.

Recall, that given the cyclic group $\mathbb{G}$ of order q, the prover wants to convince the verifier that he knows the discrete logarithm α of $h \in \mathbb{G}$ with respect to the generator $g \in \mathbb{G}$ (so that $g^\alpha = h$).
Here are five attempts to construct the protocol:

Attempt 1. Prover sends witness α to the verifier. Verifier checks whether $h = g^\alpha$.

Attempt 2. Prover chooses random $r \xleftarrow{R} \mathbb{Z}_q$ and sends $a \leftarrow \alpha + r$ to the verifier. Verifier checks whether $h = g^a$.

Attempt 3. Prover chooses random $r \xleftarrow{R} \mathbb{Z}_q$, calculates $a \leftarrow \alpha + r$ and sends both (a, r) to the verifier. Verifier checks whether $g^r h = g^a$.

Attempt 4. Prover chooses random $r \xleftarrow{R} \mathbb{Z}_q$, calculates $a \leftarrow g^r, z \leftarrow \alpha + r$ and sends (a, z) to the verifier. Verifier checks whether $a \cdot h = g^z$.

Attempt 5. Prover chooses random $r \xleftarrow{R} \mathbb{Z}_q$, calculates $a \leftarrow g^r$, and sends a to the verifier. Verifier chooses $e \xleftarrow{R} \mathbb{Z}_q$ and sends to the prover. Prover calculates $z \leftarrow \alpha e + r$ and sends to the prover. Verifier checks whether $a \cdot h^e = g^z$.

Below, mark whether the properties of *completeness*, *soundness*, and *zero-knowledge* hold for each attempt.

Attempt #	**1**	**2**	**3**	**4**	**5**
Completeness holds?	✓/✗	✓/✗	✓/✗	✓/✗	✓/✗
Soundness holds?	✓/✗	✓/✗	✓/✗	✓/✗	✓/✗
Zero-Knowledge holds?	✓/✗	✓/✗	✓/✗	✓/✗	✓/✗

Exercises 6-10. Non-Interactive Chaum-Pedersen Protocol.

This section explores how to make the previously considered Chaum-Pedersen protocol non-interactive. Fill in the gaps in the following text with the correct statements.

Recall that the Chaum-Pedersen protocol allows the prover $\mathcal{P}$ to convince the skeptical verifier $\mathcal{V}$ that the given triplet $(u, v, w) \in \mathbb{G}^3$ is a Diffie-Hellman (DH) triplet in the cyclic group $\mathbb{G}$ of prime order q with a generator $g \in \mathbb{G}$, meaning that $u = g^\alpha$, $v = g^\beta$, $w = g^{\alpha\beta}$ for some $\alpha, \beta \in \mathbb{Z}_q$. However, instead of making (α, β) as a witness, observe that β is sufficient. Indeed, if $u = g^\alpha$, $v = g^\beta$, then $w =$ **6**. Thus, the relation is:

$$\mathcal{R} = \left\{ ((u, v, w), \beta) \in \mathbb{G}^3 \times \mathbb{Z}_q : \boxed{7} \right\}$$

Now, we apply the *Fiat-Shamir Transformation*. Recall that prover, instead of getting the random challenge $c \xleftarrow{R} \mathcal{C} \subset \mathbb{Z}_q$ from the verifier interactively, calculates it as the hash function from the public statement (u, v, w) and the prover's commitment. For that reason, define the non-interactive proof system $\Phi = (\mathsf{Gen}, \mathsf{Verify})$ as follows:

- Gen: On input $(u, v, w) \in \mathbb{G}^3$,
  1. Sample $\beta_r \xleftarrow{R} \mathbb{Z}_q$ and compute the commitment **8**.
  2. Use the hash function **9** to get the challenge $c \leftarrow$ **10**.
  3. Compute response $\beta_z \leftarrow \beta_r + \beta c$ and output commitment (v_r, w_r) and β_z as a proof π.
- Verify: Upon receiving statement (u, v, w) and a proof $\pi = (v_r, w_r, \beta_z)$, the verifier:
  1. Recomputes the challenge c using the hash function.
  2. Accepts if and only if $g^{\beta_z} = v_r v^c$ and $u^{\beta_z} = w_r w^c$.

Exercise 6.

A) v^β
B) u^β
C) vu
D) v^u
E) $v^\beta u$

Exercise 7.

A) $v = g^\beta$ and $w = vu$
B) $v = g^\beta$ and $w = v^\beta$
C) $v = g^\beta$ and $w = u^\beta$
D) $u = g^\beta$ and $w = u^\beta$
E) $u/w = g^\beta$

Exercise 8.

A) $(v_r, w_r) = (g^{\beta_r}, g^{\beta_r\beta})$
B) $(v_r, w_r) = (g^{\beta_r}, w^{\beta_r})$
C) $(v_r, w_r) = (g^{\beta_r}, u^{\beta_r})$
D) $(v_r, w_r) = (g^\beta, g^{\beta_r})$
E) $(v_r, w_r) = (g^\beta, g^{\beta_r} g^\beta)$

Exercise 9.

A) $H : \mathbb{G}^3 \times \mathbb{G}^2 \to \mathcal{C}$
B) $H : \mathbb{G}^3 \times (\mathbb{G} \times \mathbb{Z}_q) \to \mathcal{C}$
C) $H : \mathbb{G}^3 \to \mathcal{C}$
D) $H : \mathbb{G}^3 \times \mathbb{Z}_q \to \mathcal{C}$
E) $H : \mathbb{G}^2 \times \mathbb{Z}_q \to \mathcal{C}$

Exercise 10.

A) $H((u, v, w), (v_r, w_r))$
B) $H((u, v, w), (v_r, \beta_r))$
C) $H(u, v, w)$
D) $H((u, v, w), \beta_r)$
E) $H((v_r, w_r), \beta_r)$