Lecture Note 8: Privacy-Enhancing Technologies 2
Zero Knowledge Proof

Rui Song and Yuhuan Liu

March 13, 2023

Summary

In this lecture, we first discuss identification protocols and show how to build a secure signature scheme based on an identification protocol. In particular, we introduce an elegant and efficient identification protocol called the Schnorr's identification protocol, and the Schnorr signature scheme derived from it. The scheme can be proven secure under the DL assumption with a hash function that can be modeled as a random oracle. Subsequently, we further generalize these protocols, introduce the concept of Sigma protocols, and describe some implementations of Sigma protocols. Finally, we present non-interactive zero-knowledge proof systems in a higher dimension and outline SNARK techniques capable of proving arbitrary NP relations using arithmetic circuits.

1 Identification Protocols and Signatures

1.1 Identification/Authentication Paradigm

Identification protocols are used in all scenarios where a prover wants to convince the verifier of its identity. The prover has a secret key sk, while the verifier has a verification key vk to confirm the prover's claim.

Definition 1.1 (Identification Protocol). *An identification protocol is a triple of three algorithms $\mathcal{I} = (G, P, V)$, where:*

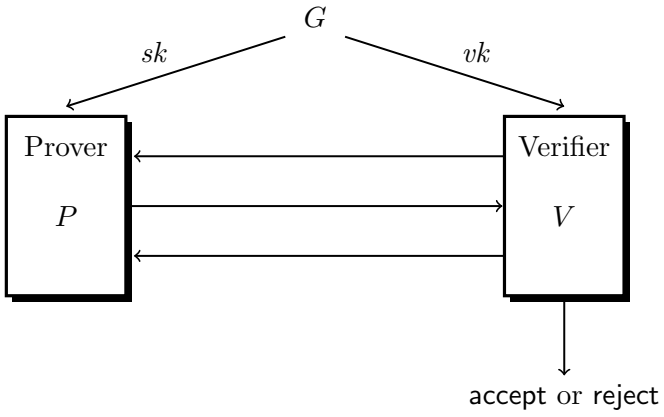

Figure 1: Identification Protocol.

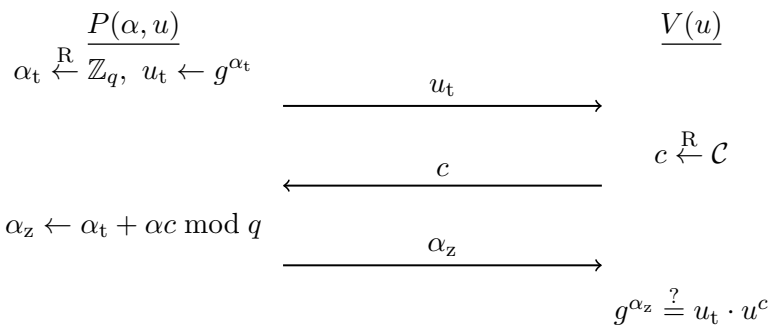

Figure 2: Schnorr Identification Protocol.

- *G is a probabilistic key generation algorithm that outputs a secret key sk and a verification key vk.*
- *P is a prover algorithm which takes a secret key sk as input.*
- *V is a verifier algorithm which takes a verification key vk as input, and outputs either* accept *or* reject*.*

During the identification process, P and V interact with each other. For all valid (vk, sk) generated by G, if P is initialized by sk and V is initialized by vk, V must output accept *with probability* 1 *at the end of the interaction.*

1.2 Schnorr's Identification Protocol

Let $\mathbb{G}$ be a cyclic group of prime order q with generator $g \in \mathbb{G}$. Suppose the prover P has a secret key $sk = \alpha \in \mathbb{Z}_q$, and the corresponding verification key is $vk = u = g^\alpha \in \mathbb{G}$. To convince its identity to a verifier V, P actually wants to convince V that it knows the secret key α.

Schnorr identification protocol is a well-designed protocol that allows P to convince V that it knows the discrete logarithm of $vk = u$, while not directly sending the value $sk = \alpha$ to V [Sch90].

Definition 1.2 (Schnorr's Identification Protocol). *Schnorr's identification protocol is an interactive protocol with a triple of three algorithms $\mathcal{I}_{\text{sch}} = (G, P, V)$, where:*

- *G is a probabilistic key generation algorithm that runs as follows:*

$$\alpha \xleftarrow{R} \mathbb{Z}_q, \quad u \leftarrow g^\alpha.$$

  The verification key is $vk := u$, while the secret key is $sk := \alpha$.

- *P is a prover algorithm which takes a secret key $sk = \alpha$ as input.*
- *V is a verifier algorithm which takes a verification key $vk = u$ as input.*
- *P and V interact with each other as follows:*
  1. *P computes $\alpha_t \xleftarrow{R} \mathbb{Z}_q$, $u_t \leftarrow g^{\alpha_t}$, and sends u_t to V;*
  2. *V generates a challenge $c \xleftarrow{R} \mathcal{C}$ where $\mathcal{C}$ is a subset of $\mathbb{Z}_q$, and sends c to P;*
  3. *P computes a responce $\alpha_z \leftarrow \alpha_t + \alpha c \in \mathbb{Z}_q$ corresponding to the challenge c, and sends α_z to V;*

4. V checks if $g^{\alpha_z} = u_t \cdot u^c$ holds, if it is the case, V outputs accept*, otherwise outputs* reject*.*

Schnorr's identification protocol defined in Theorem 1.2 is secure against *direct* attacks under the DL assumption. As shown in Theorem 1.1, any efficient adversary that can succeed in a direct attack with non-negligible probability can be turned into an algorithm that can efficiently recover the secret α given the verification key u [BS23].

Theorem 1.1 (Security against Direct Attacks). *Under the DL assumption for $\mathbb{G}$, and assuming $|\mathcal{C}|$ is super-poly, Schnorr's identification protocol is secure against direct attacks.*

Proof sketch. Suppose $\mathcal{A}$ has advantage ϵ in attacking $\mathcal{I}_{\text{sch}}$. The challenger generates the verification key $u = g^{\alpha}$. In the attacking attempt, $\mathcal{A}$ generates the first transcript u_t arbitrarily. To succeed, $\mathcal{A}$ must respond to the random challenge c with a valid response α_z which satisfies $g^{\alpha_z} = u_t \cdot u^c$. Actually, if $\mathcal{A}$ can generate a valid response to such a challenge with probability ϵ, it should be able to generate a valid response to 2 such challenges with probability ϵ^2.

Then, we can take advantage of $\mathcal{A}$ to compute the discrete logarithm of a random $u \in \mathbb{G}$. Use u as the verification key in $\mathcal{I}_{\text{sch}}$, and let $\mathcal{A}$ generate the first transcript u_t. Then, we feed a random challenge c to $\mathcal{A}$ and hope it can generate a valid response α_z. If this happens, we can *rewind* $\mathcal{A}$'s internal state back to the point when it just finished generating u_t, and feed it with another challenge c', and hope it to generates another valid response u'_t.

If all of the above happens (with probability $\approx \epsilon^2$), we actually obtain 2 valid transcripts (u_t, c, α_z) and (u_t, c', α'_z) with the same verification key u and the first transcript u_t. And since we assume $\mathcal{C}$ is super-poly, we have $c' \neq c$ with overwhelming probability. Then, since either transcript is valid, we have the following equations:

$$g^{\alpha_z} = u_t \cdot u^c, \quad g^{\alpha'_z} = u_t \cdot u^{c'}.$$

By dividing the first one from the second, we can get:

$$g^{\alpha_z - \alpha'_z} = u^{c - c'}.$$

Since $c \neq c'$, $1/(c - c')$ must exists in $\mathbb{Z}_q$. So we can get:

$$g^{(\alpha_z - \alpha'_z)/(c - c')} = u.$$

Thus, we compute the discrete logarithm of u using an efficient adversary $\mathcal{A}$, which can directly attack Schnorr's identification protocol. While we assume the discrete logarithm of $\mathbb{G}$ is hard, no algorithm can solve it efficiently. It comes that Schnorr's identification protocol is secure against direct attacks. □

We showed that any adversary which can successfully perform a direct attack with non-negligible probability can be converted into an algorithm that efficiently recovers the secret key α from the verification key u. For this reason, Schnorr's identification protocol is sometimes referred to as a *proof of knowledge* of discrete logarithms.

Definition 1.3 (Honest Verifier Zero-Knowledge, HVZK). *Let $\mathcal{I} = (G, P, V)$ be an identification protocol. We say that $\mathcal{I}$ is* ***honest verifier zero-knowledge (HVZK)*** *if there is an efficient probabilistic simulation algorithm* Sim *such that for all possible (vk, sk) generated by G, the output of* $\mathsf{Sim}(vk)$ *is indistinguishable with the transcript between $P(sk)$ and $V(vk)$.*

Theorem 1.2. *Schnorr's identification protocol is HVZK.*

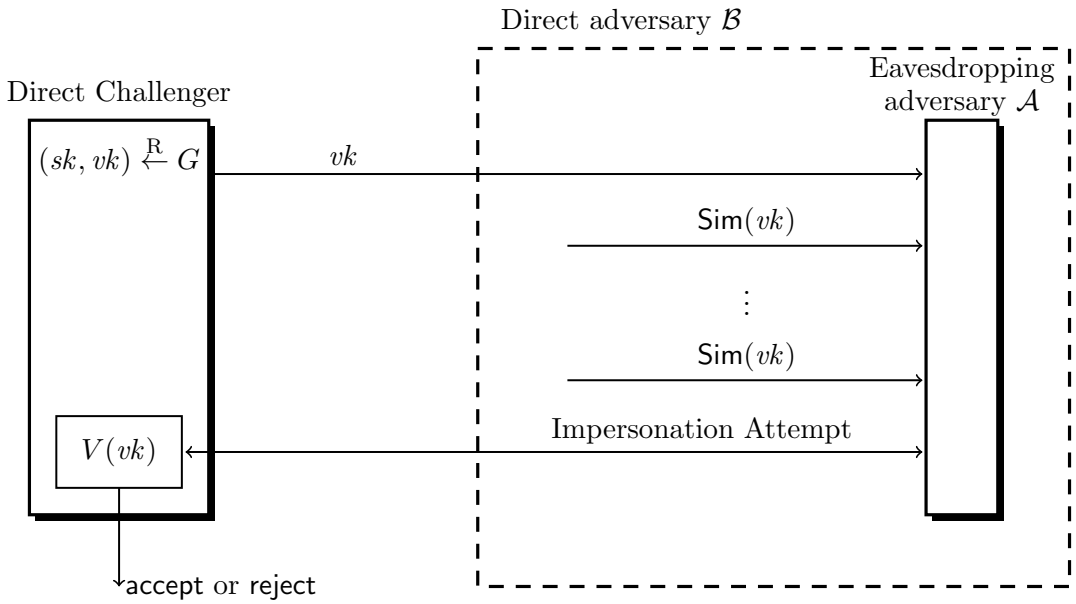

Figure 3: Proof of Theorem 1.3.

Proof. The simulator $\mathsf{Sim}(u)$ generates the messages in a reverse manner. On the input u, Sim performs as follows:

1. randomly chooses $\alpha_z \xleftarrow{R} \mathbb{Z}_q$;
2. randomly chooses $c \xleftarrow{R} \mathcal{C}$;
3. computes $u_t \leftarrow g^{\alpha_z}/u^c$.

Then we want to argue that the transcript generated by $\mathsf{Sim}(u)$ is indistinguishable from the transcript between P and V. Actually, in a real interaction, c and α_z are independent with each other, since $c \in \mathcal{C}$ and $\alpha_z \in \mathbb{Z}_q$ are both uniformly distributed. Upon this base, given c and α_z, the value of u_t is uniquely determined by $u_t = g^{\alpha_z}/u^c$. Thus, the output of the simulator Sim and the transcript between P and V have the same distribution. □

In the following Theorem 1.3, we will show that any identification protocol which is HVZK and secure against direct attacks is also secure against eavesdropping attacks.

Theorem 1.3 (Security against Eavesdropping Attacks). *If an identification protocol $\mathcal{I}$ is secure against direct attacks, and it is HVZK, then the protocol is also secure against eavesdropping attacks.*

Proof sketch. For every eavesdropping adversary $\mathcal{A}$, we can construct another adversary $\mathcal{B}$ who interacts with the challenger in the direct attack game. $\mathcal{B}$ works the same as $\mathcal{A}$, except that it generates the transcripts itself using Sim, and feed them to $\mathcal{A}$. It concludes that the probability of $\mathcal{B}$ winning in the eavesdropping attack game is the same as that of $\mathcal{A}$ winning in the direct attack game. □

Theorem 1.4 (Schnorr's Security). *If Schnorr's identification protocol is secure against direct attacks, then it is also secure against eavesdropping attacks.*

Proof. This theorem comes directly from Theorem 1.2 and Theorem 1.3. □

To summarize, Schnorr's identification protocol has the following three important properties:

1. Completeness: if P and V execute the protocol honestly, V will always output accept.
2. Soundness: If V outputs accept, we can extract a valid witness α effectively.
3. HVZK: we can efficiently simulate valid transcripts even if we do not know the witness α.

1.3 Schnorr Signature

We can convert Schnorr's identification protocol to a signature scheme. This signature scheme can be proven secure in the random oracle model (ROM) under the discrete logarithm assumption.

Recall that in Schnorr's identification protocol $\mathcal{I}_{\text{sch}}$, we need a cyclic group $\mathbb{G}$ of prime order q with generator $g \in \mathbb{G}$, along with a challenge space $\mathcal{C} \subseteq \mathbb{Z}_q$. Now, we need a hash function $H : \mathcal{M} \times \mathbb{G} \to \mathcal{C}$ which can be modeled as a random oracle, where $\mathcal{M}$ is the message space in the signature scheme [Sch91].

Definition 1.4 (Schnorr Signature). *Schnorr signature scheme is a protocol with a triple of three algorithms $\mathcal{S}_{\text{sch}} = (G, S, V)$, where:*

- *G is a probabilistic key generation algorithm that runs as follows:*

$$\alpha \xleftarrow{\text{R}} \mathbb{Z}_q, \quad u \leftarrow g^{\alpha}.$$

  The public key is $pk := u$, while the secret key is $sk := \alpha$.

- *S is a signing algorithm which signs a message $m \in \mathcal{M}$ using a secret key $sk = \alpha$. S runs as follows:*

$$\alpha_{\text{t}} \xleftarrow{\text{R}} \mathbb{Z}_q, \quad u_{\text{t}} \leftarrow g^{\alpha_{\text{t}}}, \quad c \leftarrow H(m, u_{\text{t}}), \quad \alpha_{\text{z}} \leftarrow \alpha_{\text{t}} + \alpha c.$$

  S outputs $\sigma := (u_{\text{t}}, \alpha_{\text{z}})$ as the signature on the message m.

- *V is a verification algorithm which verifies a signature $\sigma = (u_{\text{t}}, \alpha_{\text{z}})$ on a message $m \in \mathcal{M}$ using a public key $pk = u$. To this end, S computes $c \leftarrow H(m, u_{\text{t}})$, and output* accept *if and only if $g^{\alpha_{\text{z}}} = u_{\text{t}} \cdot u^{c}$.*

2 Sigma Protocol

2.1 Definition of Sigma Protocols

Schnorr's identification protocol is a special case of a class of protocols called *Sigma protocols*. Before introducing Sigma protocols, we first declare some useful concepts.

Definition 2.1 (Effective Relation). *An effective relation is a binary relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{R}$ are finite sets. Elements in $\mathcal{Y}$ are called **statements**. If $(x, y) \in \mathcal{R}$, then x is called a **witness** for y.*

Definition 2.2 (The Language of True Statements). *Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ be an effective relation. We say a statement $y \in \mathcal{Y}$ is a **true statement** if $(x, y) \in \mathcal{R}$ holds for some $x \in \mathcal{X}$; otherwise, we say y is a false statement.*

*We denote to $L_{\mathcal{R}}$ the **language** defined by $\mathcal{R}$, i.e., the set of all true statements. Thus we have*

$$L_{\mathcal{R}} = \{y \in \mathcal{Y} \mid \exists x \in \mathcal{X} \text{ s.t. } (x, y) \in \mathcal{R}\}.$$

Then we can define the syntax of a Sigma protocol.

Definition 2.3 (Sigma Protocol). *Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ be an effective relation. A Sigma protocol for $\mathcal{R}$ is a pair of algorithms (P, V), where:*

- *P is an interactive algorithm called the prover which takes a witness $x \in \mathcal{X}$ and a statement $y \in \mathcal{Y}$ as input, where $(x, y) \in \mathcal{R}$.*
- *V is an interactive algorithm called the verifier which takes a statement $y \in \mathcal{Y}$ as input, and outputs either* accept *or* reject*.*
- *P and V interact with each other as follows:*
  - *At the beginning of the protocol, P computes a message t which is called the commitment, and sends t to V;*
  - *Upon receiving t from P, V randomly chooses a challenge c from a finite challenge space $\mathcal{C}$, and sends c back to P;*
  - *Upon receiving c from V, P computes a response z according to c, and sends z back to V;*
  - *Upon receiving z from P, V outputs either* accept *or* reject*. V's computation should be a function of the statement y and the transcript (t, c, z). Particularly, V does not make any random choices other than the selection of the challenge c, i.e., all other computations are strictly deterministic.*

Just like Schnorr's identification protocol, Sigma protocols have the following three properties:

1. Completeness: if P and V execute the protocol honestly, V will always output accept.
2. Soundness: If V outputs accept, we can extract a valid witness x based on the transcript (t, c, z) and (t, c', z') effectively.
3. HVZK: we can efficiently simulate valid transcripts for $y \in \mathcal{Y}$ even if we do not know the witness $x \in \mathcal{X}$.

2.2 Cases of Sigma Protocols

2.2.1 Schnorr's Protocol

As introduced before, Schnorr's protocol is a special case of Sigma protocols. In Schnorr's protocol, a prover can convince a verifier that it knows the discrete logarithm of a given group element, while not revealing the value of the very discrete logarithm. If using the definition of relation and language above, we can denote the relation of Schnorr's identification protocol as:

$$\mathcal{R}_{\text{sch}} = \Big\{(\alpha, u) \in \mathbb{Z}_q \times \mathbb{G} : g^{\alpha} = u\Big\}.$$

2.2.2 Okamoto's Protocol

Let $\mathbb{G}$ be a cyclic group of prime order q with generator $g \in \mathbb{G}$. Let some arbitrary group element $h \in \mathbb{G}$ be a system parameter that is generated beforehand and is publicly accessible to all parties.

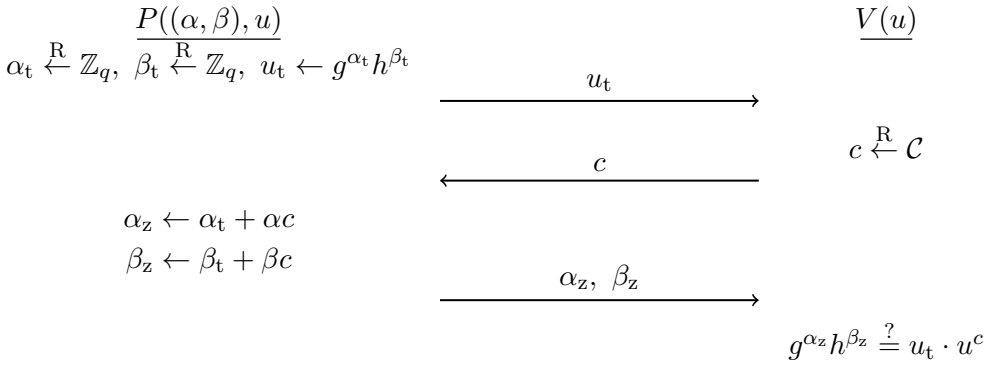

Figure 4: Okamoto's Protocol.

Okamoto's protocol allows a prover to convince a verifier that it knows a representation of a given group element $u \in \mathbb{G}$, but not need to provide the representation to the verifier [Oka92]. Specifically, the relation of Okamoto's protocol is as follows:

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

A witness for the statement $u \in \mathbb{G}$ is $(\alpha,\beta) \in \mathbb{Z}_q^2$ such that $g^\alpha h^\beta = u$. It should be clear that every statement in Okamoto's protocol should have multiple (actually q) witnesses.

Definition 2.4 (Okamoto's Protocol). *Okamoto's protocol is an interactive protocol with a pair of algorithms $\mathcal{I}_{\text{oka}} = (P, V)$, where:*

- *P computes:*
$$\alpha_t \xleftarrow{R} \mathbb{Z}_q, \quad \beta_t \xleftarrow{R} \mathbb{Z}_q, \quad u_t \leftarrow g^{\alpha_t} h^{\beta_t},$$
and sends u_t to V.
- *V generates a challenge $c \xleftarrow{R} \mathcal{C}$ where $\mathcal{C}$ is a subset of $\mathbb{Z}_q$, and sends c to P.*
- *P computes:*
$$\alpha_z \leftarrow \alpha_t + \alpha c \in \mathbb{Z}_q, \quad \beta_z \leftarrow \beta_t + \beta c \in \mathbb{Z}_q,$$
and sends (α_z, β_z) to V.
- *V checks if $g^{\alpha_z} h^{\beta_z} = u_t \cdot u^c$ holds, if it is the case, V outputs* accept*, otherwise outputs* reject*.*

Like Schnorr's identification protocol, Okamoto's protocol has the properties of completeness, soundness, and HVZK.

2.2.3 Chaum-Pedersen Protocol

Definition 2.5 (DH-Triple). *Let $\mathbb{G}$ be a cyclic group of prime order q with generator $g \in \mathbb{G}$. For $\alpha, \beta, \gamma \in \mathbb{Z}_q$, we say $(g^\alpha, g^\beta, g^\gamma)$ is a DH-triple if $\gamma = \alpha\beta$ holds.*

The Chaum-Pedersen protocol allows a prover to convince a verifier that a given triple (u, v, w) is a DH-triple, while not revealing any one of the elements in the DH-triple [CP93]. Specifically, the relation of the Chaum-Pedersen protocol is as follows:

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

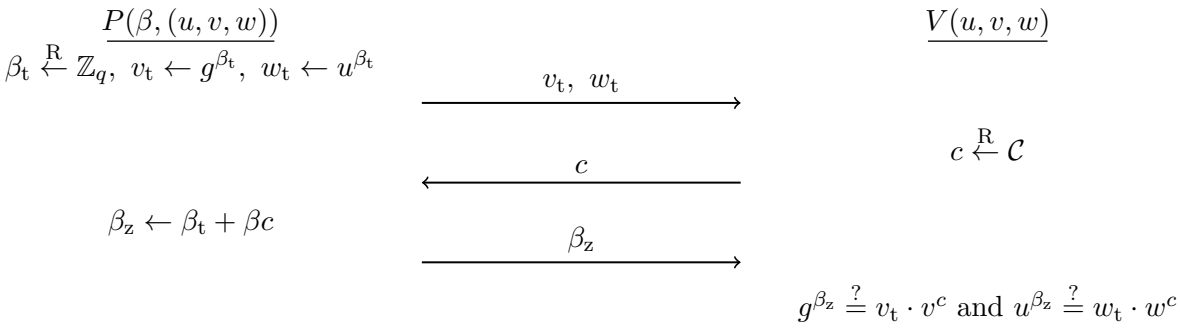

Figure 5: Chaum-Pedersen Protocol.

A witness for the statement $(u, v, w) \in \mathbb{G}^3$ is $\beta \in \mathbb{Z}_q$ such that $v = g^\beta$ and $w = u^\beta$ hold. It should be clear that a statement (u, v, w) has a witness if and only if it is a DH-triple. Thus, not every statement in this protocol has a witness.

Definition 2.6 (Chaum-Pedersen Protocol). *Chaum-Pedersen protocol is an interactive protocol with a pair of algorithms $\mathcal{I}_{\text{cp}} = (P, V)$, where:*

- *P computes:*
$$\beta_t \xleftarrow{R} \mathbb{Z}_q, \quad v_t \leftarrow g^{\beta_t}, \quad w_t \leftarrow u^{\beta_t},$$
and sends (v_t, w_t) to V.
- *V generates a challenge $c \xleftarrow{R} \mathcal{C}$ where $\mathcal{C}$ is a subset of $\mathbb{Z}_q$, and sends c to P.*
- *P computes:*
$$\beta_z \leftarrow \beta_t + \beta c \in \mathbb{Z}_q,$$
and sends β_z to V.
- *V checks if $g^{\beta_z} = v_t \cdot v^c$ and $u^{\beta_z} = w_t \cdot w^c$ hold, if it is the case, V outputs* accept*, otherwise outputs* reject*.*

Chaum-Pedersen protocol also has the properties of completeness, soundness, and HVZK. Given an input of a triple $(u, v, w) \in \mathbb{G}^3$ and $c \in \mathcal{C}$, the simulator Sim computes:

$$\beta_z \xleftarrow{R} \mathbb{Z}_q, \quad v_t \leftarrow g^{\beta_z}/v^c, \quad w_t \leftarrow u^{\beta_z}/w^c,$$

and outputs $\big((v_t, w_t), \beta_z\big)$. It should be clear that the output of Sim is always valid as a transcript, as required for HVZK.

2.3 Combination of Sigma Protocols

Sigma protocols can be combined to prove more complicated relations. In the AND composition, the prover could convince the verifier that it knows witnesses for multiple statements. While in the OR composition, the prover could convince the verifier that it knows witnesses for one of the given statements.

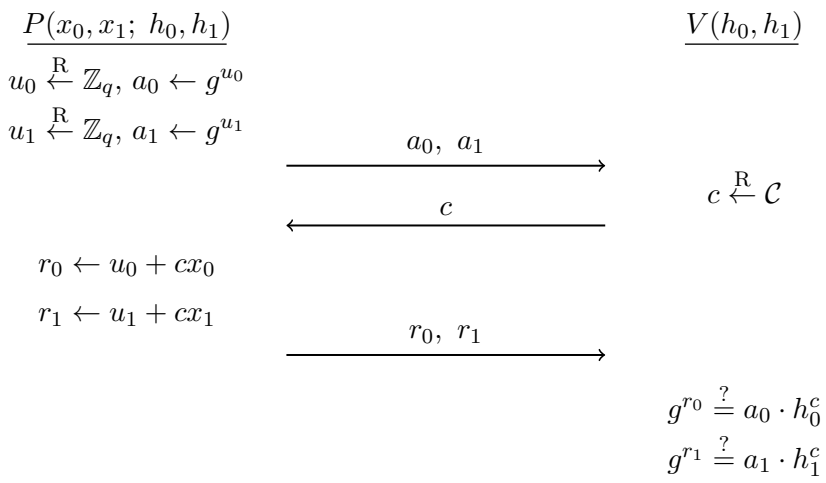

Figure 6: Protocol for AND Composition (for DL Relations).

2.3.1 AND Composition

Definition 2.7 (AND Composition). *Given a Sigma protocol (P_0, V_0) for relation $\mathcal{R}_0 \subseteq \mathcal{X}_0 \times \mathcal{Y}_0$ and a Sigma protocol (P_1, V_1) for relation $\mathcal{R}_1 \subseteq \mathcal{X}_1 \times \mathcal{Y}_1$. Assume that both protocols use the same challenge space $\mathcal{C} \subseteq \mathbb{Z}_q$. The two Sigma protocols can be combined into a new Sigma protocol for relation:*

$$\mathcal{R}_{\text{AND}} = \Big\{ \big((x_0, x_1), (y_0, y_1)\big) \in (\mathcal{X}_0 \times \mathcal{X}_1) \times (\mathcal{Y}_0 \times \mathcal{Y}_1) \ :\ (x_0, y_0) \in \mathcal{R}_0 \ \wedge\ (x_1, y_1) \in \mathcal{R}_1 \Big\}.$$

In the case where $\mathcal{R}_0$ and $\mathcal{R}_1$ are both DL relations, the relation of the AND combination for $\mathcal{R}_0 \wedge \mathcal{R}_1$ is:

$$\mathcal{R}_{\text{AND-Sch}} = \Big\{ \big((x_0, x_1), (h_0, h_1)\big) \in \mathbb{Z}_q^2 \times \mathbb{G}^2 \ :\ h_0 = g^{x_0} \ \wedge\ h_1 = g^{x_1} \Big\}.$$

As shown in Figure 6, the protocol (P, V) for proving the above AND composition relation is as follows:

- P computes:
$$u_0 \xleftarrow{R} \mathbb{Z}_q, \quad u_1 \xleftarrow{R} \mathbb{Z}_q, \quad a_0 \leftarrow g^{u_0}, \quad a_1 \leftarrow g^{u_1},$$
and sends (a_0, a_1) to V.
- V generates a challenge $c \xleftarrow{R} \mathcal{C}$ and sends c to P.
- P computes:
$$r_0 \leftarrow u_0 + cx_0 \in \mathbb{Z}_q, \quad r_1 \leftarrow u_1 + cx_1 \in \mathbb{Z}_q,$$
and sends (r_0, r_1) to V.
- V checks if $g^{r_0} = a_0 \cdot h_0^c$ and $g^{r_1} = a_1 \cdot h_1^c$ hold, if it is the case, V outputs accept, otherwise outputs reject.

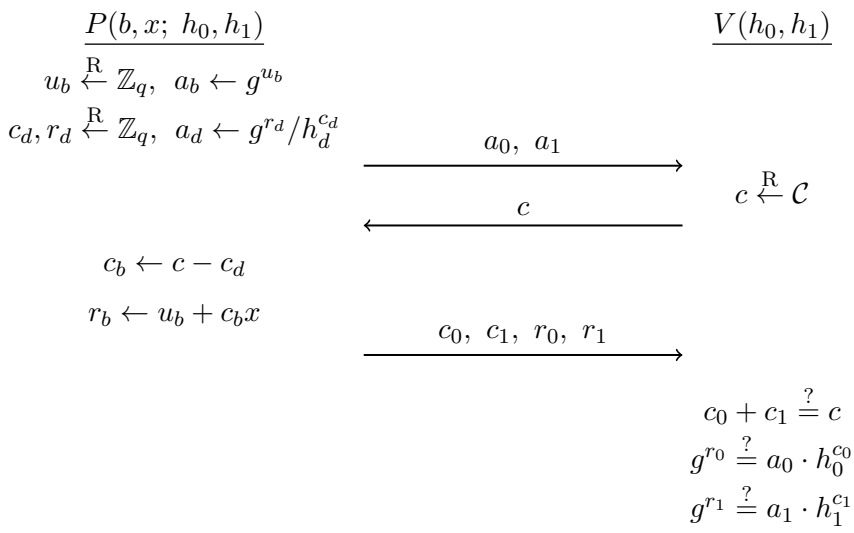

Figure 7: Protocol for OR Composition (for DL Relations).

2.3.2 OR Composition

Definition 2.8 (OR Composition). *Given a Sigma protocol (P_0, V_0) for relation $\mathcal{R}_0 \subseteq \mathcal{X}_0 \times \mathcal{Y}_0$ and a Sigma protocol (P_1, V_1) for relation $\mathcal{R}_1 \subseteq \mathcal{X}_1 \times \mathcal{Y}_1$. Assume that both protocols use the same challenge space $\mathcal{C} \subseteq \mathbb{Z}_q$. The two Sigma protocols can be combined into a new Sigma protocol for relation:*

$$\mathcal{R}_{\text{OR}} = \Big\{ \big((b, x), (y_0, y_1)\big) \in \big(\{0,1\} \times (\mathcal{X}_0 \cup \mathcal{X}_1)\big) \times (\mathcal{Y}_0 \times \mathcal{Y}_1) \ : \ (x, y_b) \in \mathcal{R}_b \Big\}.$$

In the case where $\mathcal{R}_0$ and $\mathcal{R}_1$ are both DL relations, the relation of the OR combination for $\mathcal{R}_0 \vee \mathcal{R}_1$ is:

$$\mathcal{R}_{\text{OR-Sch}} = \Big\{ \big((b, x), (h_0, h_1)\big) \in \big(\{0,1\} \times \mathbb{Z}_q\big) \times \mathbb{G}^2 \ : \ h_b = g^x \Big\}.$$

As shown in Figure 7, the protocol (P, V) for proving the above OR composition relation is as follows:

- P computes:

  $$c_d \xleftarrow{R} \mathbb{Z}_q, \quad r_d \xleftarrow{R} \mathbb{Z}_q, \quad u_b \xleftarrow{R} \mathbb{Z}_q, \quad a_d \leftarrow g^{r_d}/h_d^{c_d}, \quad a_b \leftarrow g^{u_b},$$

  where $d = 1 - b$, and sends (a_0, a_1) to V.

- V generates a challenge $c \xleftarrow{R} \mathcal{C}$ and sends c to P.

- P computes:

  $$c_b \leftarrow c - c_d, \quad r_b \leftarrow u_b + c_b x \in \mathbb{Z}_q,$$

  and sends (c_0, c_1, r_0, r_1) to V.

- V checks if $c_0 + c_1 \stackrel{?}{=} c$, $g^{r_0} = a_0 \cdot h_0^{c_0}$ and $g^{r_1} = a_1 \cdot h_1^{c_1}$ hold, if it is the case, V outputs accept, otherwise outputs reject.

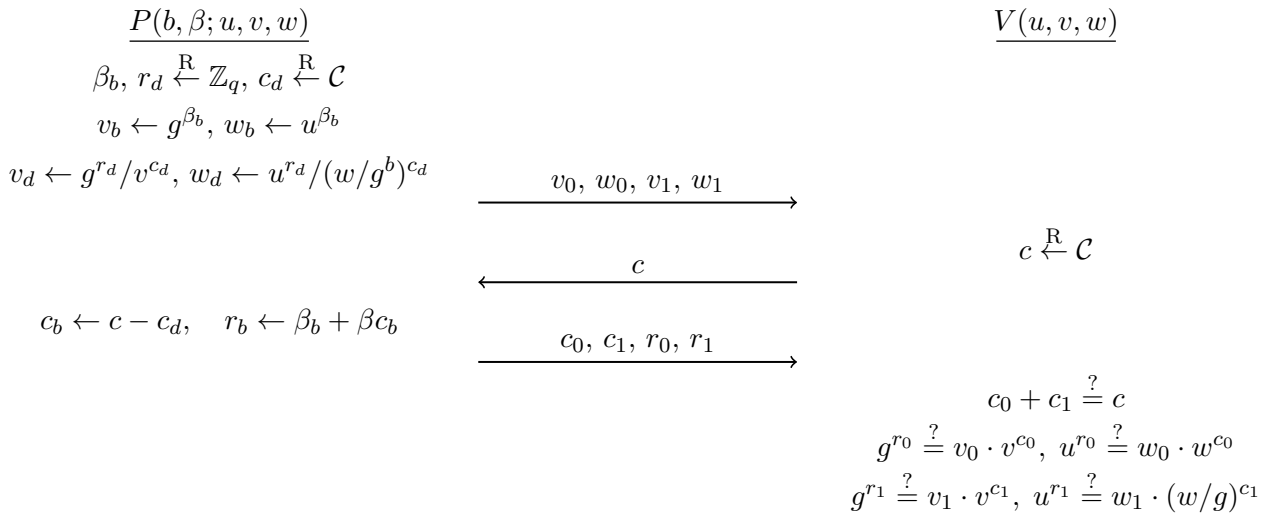

Figure 8: Protocol for OR Composition (for DH-Triple Relations).

In another case where $\mathcal{R}_0$ and $\mathcal{R}_1$ are the relations of DH-triples, the relation of the OR composition for $\mathcal{R}_0 \vee \mathcal{R}_1$ is:

$$\mathcal{R}_{\mathrm{OR-CP}} = \Big\{ \big((b, \beta), (u, v, w)\big) \in \big(\{0, 1\} \times \mathbb{Z}_q\big) \times \mathbb{G}^3 \ : \ v = g^{\beta} \ \wedge \ w = u^{\beta} \cdot g^b \Big\}.$$

The above relation indicates that (u, v, w) is the ElGamal encryption of 0 or 1 if and only if (u, v, w) or $(u, v, w/g)$ is a DH-triple.

As shown in Figure 8, the protocol (P, V) for proving the above OR composition relation is as follows:

- P computes:
$$\beta_b \xleftarrow{\mathrm{R}} \mathbb{Z}_q, \quad v_b \leftarrow g^{\beta_b}, \quad w_b \leftarrow u^{\beta_b},$$
$$c_d \xleftarrow{\mathrm{R}} \mathbb{Z}_q, \quad r_d \xleftarrow{\mathrm{R}} \mathbb{Z}_q, \quad v_d \leftarrow g^{r_d}/v^{c_d}, \quad w_d \leftarrow u^{r_d}/(w/g)^{c_d},$$
where $d = 1 - b$, and sends (v_0, w_0, v_1, w_1) to V.
- V generates a challenge $c \xleftarrow{\mathrm{R}} \mathcal{C}$ and sends c to P.
- P computes:
$$c_b \leftarrow c - c_d, \quad r_b \leftarrow \beta_b + \beta c_b \in \mathbb{Z}_q,$$
and sends (c_0, c_1, r_0, r_1) to V.
- V checks if $c_0 + c_1 \stackrel{?}{=} c$, $g^{r_0} = v_0 \cdot v^{c_0}$, $u^{r_0} = w_0 \cdot w^{c_0}$, $g^{r_1} = v_1 \cdot v^{c_1}$ and $u^{r_1} = w_1 \cdot (w/g)^{c_1}$ hold, if it is the case, V outputs accept, otherwise outputs reject.

The above protocol could be used in encrypting a bit using ElGamal encryption. Suppose one encodes a bit b as g^b and then encrypts it using the receiver's public key $u \in \mathbb{G}$, generating a ciphertext $(v, w) = (g^{\beta}, u^{\beta} \cdot g^b)$. One could use the protocol to convince the verifier that (v, w) is derived from a bit (not other stuff, e.g., a number other than 0/1), while revealing nothing else.

3 Zero-Knowledge Proofs

3.1 Zero-Knowledge Proof System

Zero-knowledge proofs are extensions of Sigma protocols. As all other Sigma protocols, for language L, zero-knowledge proof systems have the following three properties:

- Completeness: if $y \in L$, and both P and V interact with each other honestly, V must output accept at the end of the interaction.
- Soundness: if $y \notin L$, for any computational-bounded prover P, V outputs accept with negligible probability.
- Zero-knowledge: any V who does not know the witness x can efficiently simulate valid transcripts for any statement $y \in L$.

Given a NP language $L_{\mathcal{R}}$ for a relation $\mathcal{R}$, a prover P initialized by $(x, y) \in \mathcal{R}$ could prove to a verifier that $y \in L$ holds. The above three properties imply the following propositions:

- if $(x, y) \in \mathcal{R}$ holds for some $x \in \mathcal{X}$, verifier V will output accept with overwhelming probability.
- if $(x, y) \notin \mathcal{R}$ holds for all $x \in \mathcal{X}$, p.p.t. verifier V will output accept with negligible probability.
- Any verifier V could learn nothing about the witness $x \in \mathcal{X}$ during the interaction.

According to Goldreich, Micali and Wigderson [GMW86], we have the following theorem:

Theorem 3.1. *If there exists a secure probabilistic encryption, then every language in NP has a zero-knowledge interactive proof system in which the prover is a probabilistic polynomial-time machine that gets an NP proof as an auxiliary input.*

3.2 Non-Interactive Zero-Knowledge (NIZK)

Zero-knowledge systems based on Sigma protocols need multiple rounds of interaction. Actually, any Sigma protocol could be converted into a non-interactive proof system using Fiat-Shamir transform. The basic idea is not relying on the verifier to randomly generate the challenges, while taking advantage of a hash function H as a random oracle.

Definition 3.1 (Non-Interactive Proof System). *Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$ be an relation. A non-interactive proof system for $\mathcal{R}$ is a pair of algorithms (P, V), where:*

- *P is an efficient probabilistic algorithm which takes as input a witness $x \in \mathcal{X}$ and a statement $y \in \mathcal{Y}$ where $(x, y) \in \mathcal{R}$, and outputs a proof $\pi \in \mathcal{PS}$.*
- *V is an efficient deterministic algorithm that takes as input a statement $y \in \mathcal{Y}$ and a proof $\pi \in \mathcal{PS}$, and outputs either* accept *or* reject.

The following introduced Fiat-Shamir transform is a technique that can convert a Sigma protocol into a non-interactive proof protocol [FS86].

Definition 3.2 (Fiat-Shamir Transform). *Let $\Pi = (P, V)$ be a Sigma protocol for a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{Y}$. Assume that transcripts (t, c, z) for Π belongs to the space $\mathcal{T} \times \mathcal{C} \times \mathcal{Z}$. Let $H : \mathcal{Y} \times \mathcal{T} \to \mathcal{C}$ be a hash function. We define the Fiat-Shamir non-interactive proof system $\Pi_{\text{fs}} = (P', V')$ with proof space $\mathcal{PS} = \mathcal{T} \times \mathcal{Z}$:*

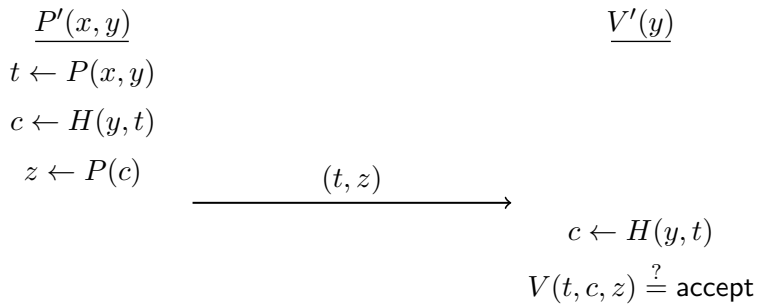

Figure 9: Non-Interactive Proof System based on Fiat-Shamir Transform.

- *On input $(x,y) \in \mathcal{R}$, P' runs $P(x,y)$ to obtain a commitment $t \in \mathcal{T}$. It then feeds the challenge $c \leftarrow H(y,t)$ to $P(x,y)$, and obtains a response $z \in \mathcal{Z}$. It finally outputs $(t,z) \in \mathcal{T} \times \mathcal{Z}$ as the proof π.*
- *On input $(y,\pi) \in \mathcal{Y} \times \mathcal{PS}$, V' parses π as (t,z) and verifies if (t,c,z) is a valid transcript for y, where $c = H(y,t)$ can be computed by V' itself.*

It could be proven that given the underlying Sigma protocol is special HVZK and has unpredictable commitments, the Fiat-Shamir transform could always yield a NIZK.

3.3 Succinct Non-Interactive Arguments of Knowledge (SNARKs)

SNARK is such a zero-knowledge technique that can be used in arbitrary logic and circuits, and yield succinct proofs. In 2016, Groth built a zkSNARK scheme based on quadratic arithmetic programs (QAP), which is now used in various systems [Gro16].

Definition 3.3 (zkSNARKs). *Let $\mathcal{R}$ be an effective relation. For $(x,y) \in \mathcal{R}$, a non-interactive argument of knowledge is a triple of algorithms $\Pi_{\text{snark}} = (G,P,V)$, where:*

- *G is a probabilistic setup algorithm that takes as input the relation $\mathcal{R}$ and a security parameter λ, and outputs a common reference string σ.*
- *P is a probabilistic prover algorithm which takes as input a common reference string σ, a witness $x \in \mathcal{X}$ and a statement $y \in \mathcal{Y}$, outputs an argument π.*
- *V is a deterministic verifier algorithm that takes as input a common reference string σ, an argument π and a statement y, and outputs either* accept *or* reject.

*We say a proof system Π_{snark} is a **zero-knowledge succinct non-interactive argument of knowledge (zkSNARK)** if it has the properties of completeness, knowledge soundness, zero-knowledge and succinctness defined below.*

Definition 3.4 (Completeness). *We say an argument Π_{snark} is **complete** if for all $\lambda \in \mathbb{N}$ and $(x,y) \in \mathcal{R}$, we have:*

$$\Pr\left[\begin{matrix}\sigma \xleftarrow{R} G(\mathcal{R}) \\ \pi \xleftarrow{R} P(\sigma,x,y)\end{matrix} \;:\; V(\sigma,y,\pi) = 1\right] = 1.$$

Definition 3.5 (Knowledge Soundness). *For all non-uniform polynomial-time adversaries $\mathcal{A}$, we say an argument Π_{snark} is* ***knowledge sound*** *if there exists a non-uniform polynomial-time extractor* Ext *such that:*

$$\Pr\left[\begin{array}{rcl} \sigma \xleftarrow{\text{R}} G(\mathcal{R}) & : & (x,y) \notin \mathcal{R} \ \wedge \\ \big((y,\pi);x\big) \leftarrow (\mathcal{A} \,\|\, \mathsf{Ext})(\mathcal{R},\sigma) & & V(\sigma,y,\pi)=1 \end{array}\right] \approx 0.$$

Definition 3.6 (Zero-Knowledge). *For all $\lambda \in \mathbb{N}$, all $(x,y) \in \mathcal{R}$ and all adversaries $\mathcal{A}$, we say an argument Π_{snark} is* ***zero-knowledge*** *if these exists a polynomial algorithm* Sim *such that:*

$$\Pr\left[\begin{array}{c} \sigma \xleftarrow{\text{R}} G(\mathcal{R}) \\ \pi \xleftarrow{\text{R}} P(\sigma,x,y) \end{array} : \mathcal{A}(\mathcal{R},\sigma,\pi)=1\right] = \Pr\left[\begin{array}{c} \sigma \xleftarrow{\text{R}} G(\mathcal{R}) \\ \pi \xleftarrow{\text{R}} \mathsf{Sim}(\mathcal{R},y) \end{array} : \mathcal{A}(\mathcal{R},\sigma,\pi)=1\right].$$

Definition 3.7 (Succinctness). *We say an argument Π_{snark} is* ***succinct*** *if the verifier time is a polynomial of $\lambda + |y|$ and the proof size is a polynomial of λ. Upon this, if the size of the common reference string $|\sigma|$ is also a polynomial of λ, we say the argument is* ***fully succinct****.*

References

[BS23] Dan Boneh and Victor Shoup. A graduate course in applied cryptography. *Draft 0.6*, 2023.

[CP93] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology—CRYPTO'92: 12th Annual International Cryptology Conference Santa Barbara, California, USA August 16–20, 1992 Proceedings 12*, pages 89–105. Springer, 1993.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto*, volume 86, pages 186–194. Springer, 1986.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *27th Annual Symposium on Foundations of Computer Science (FCS 1986)*, pages 174–187. IEEE Computer Society, 1986.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326. Springer, 2016.

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Crypto*, volume 92, pages 31–53. Springer, 1992.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology—CRYPTO'89 Proceedings 9*, pages 239–252. Springer, 1990.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4:161–174, 1991.