1 Zero-Knowledge Proofs

A ***proof of knowledge*** is a protocol that enables one party to convince another of the validity of a statement. In a ***zero-knowledge proof***, this is accomplished without revealing any information beyond the legitimacy of the proof. We will examine several examples of zero-knowledge proofs and then formalize a definition. We begin our discussion by looking at the general formulation.

We have two parties, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$. $\mathcal{P}$ must convince $\mathcal{V}$ that she has some knowledge of a statement x without explicitly stating what she knows. We call this knowledge a ***witness*** w. Both parties are aware of a predicate R that will attest to w being a valid witness to x. In general,

- The predicate R is assumed to be polynomial-time computable: given a witness w for a statement x, one can efficiently test that $R(x, w) = 1$.
- The prover $\mathcal{P}$ has R, x, and w such that $R(x, w) = 1$. She wishes to prove possession of w by producing a proof of knowledge π.
- The verifier $\mathcal{V}$ has R, x, and π.

In order for the above protocol to be useful in cryptographic applications, we can make the following assumptions.

- Given R, it is hard to find a corresponding w such that $R(x, w) = 1$.
- The prover $\mathcal{P}$ is reluctant to reveal w; otherwise the solution is trivial.
- The verifier $\mathcal{V}$ can efficiently check the validity of π.

1.1 Examples of Zero-Knowledge Proofs

To demonstrate these concepts, we present two simple examples.

Example (Where's Waldo)**.** In the game *Where's Waldo*, there is a large board depicting an intricate scene of characters, all of whom resemble "Waldo". The objective of the game is to discern Waldo from amongst the look-alikes.

Suppose the old comrades, Alice and Bob decide to play. Alice claims to have found Waldo's exact location, but she does not want to show Bob. After all, antagonizing one's openent makes many games more entertaining.

Here the assumption that Waldo exists is the statement, Waldo's (x, y) coordinates are the witness, and the procedure of receiving (x, y) and verifying that Waldo is indeed there relates to the predicate R.

One possible solution, and admittedly not the only one, is for Alice to cover the board with a large piece of paper with a small, Waldo-sized hole in its center. To prove she has found Waldo, Alice moves the paper so that Waldo, and nothing else, is visible through the hole. Note that for this solution to be effective, the dimensions of the paper must be at least twice those of the board.

Example (Magic Door)**.** After being soundly beaten in *Where's Waldo*, Bob convinces Alice to go spelunking. The two eventually come to a cave as depicted in Figure

1. At the bottom of the cave, there is a magic door that can only be opened using a secret password. Bob proposes a new game to prove to Alice he can open the magic door.

1. Alice stands at Point 1.
2. Bob enters the cave and stands either at Point 3 or 4.

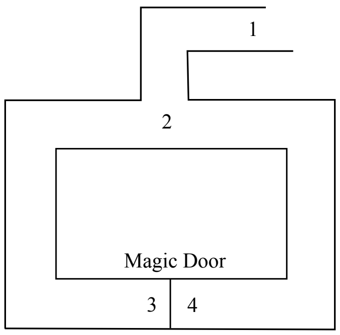

Figure 1: The door between Points 3 and 4 can only be opened using a secret password.

3. After Bob disappears, Alice walks to Point 2.
4. Alice calls to Bob, asking him to come out either the left or the right passage.
5. Bob complies, using the secret password if necessary.
6. Alice and Bob repeat Steps 1-5 k times.

This game illustrates how a probabilistic procedure can be used to model protocols of interest. In particular, after k repetitions, Bob can convince Alice with probability $1 - 1/2^k$ that he knows the magic words.

We now present two practical examples that use proofs of knowledge.

Example. We return to the NP class: the set of all problems for which a candidate solution can be verified in polynomial-time. We call a set of strings a ***language***. Let x denote any problem statement and let R represent a polynomial-time predicate. Then a language L is in NP if

$$\mathsf{L} = \{x \colon R(x,w) = 1 \text{ for some } w\}.$$

Take the language $\mathsf{CLIQUE} = \{\langle G, k\rangle \colon G \text{ is a graph with a clique of size } k\}$. The witnesses are the sets of k verticies forming a clique, and the polynomial-time predicate R verifies that the verticies form a clique.

Another language is $\mathsf{SAT} = \{\langle \Phi \rangle \colon \Phi \text{ is a satisfiable boolean formula}\}$. One can check in polynomial-time that a set of variables assigned to $\Phi \in SAT$ satisfies Φ. Zero-knowledge proofs can be used to prove a specific element is in CLIQUE or SAT. We will address how in Section 1.5

Example. One key application for zero-knowledge proofs is in user identification schemes. In traditional password mechanisms, an eavesdropping adversary can obtain enough information to gain unauthorized access in a system. In order to allay this problem, suppose the system contains a public directory that assigns a statement of a theorem to each user. In order to gain access to the system, a user must produce a proof of their theorem. Assuming that only an authorized user knows a witness to their proof, a zero-knowledge proof can convince the system of the proofs authenticity. This is directly related to the Schnorr Protocol, which we will discuss in section 1.3.

1.2 Three Basic Properties

Formalizing the definition of a proof of knowledge is a very delicate task. The following definition emerged after fifteen years of work, and has since been deemed an intellectual achievement.

Definition 1.2.1. Let $\langle \mathcal{P}, \mathcal{V} \rangle$ be a pair of interactive programs. Define $\mathsf{out}^{\mathcal{P}}_{\mathcal{P},\mathcal{V}}(x, w, z)$ to be the output of $\mathcal{P}$ when both $\mathcal{P}$ and $\mathcal{V}$ are executed with the public input x and private inputs w and z ($\mathcal{P}$ determines w and $\mathcal{V}$ chooses z); $\mathsf{out}^{\mathcal{V}}_{\mathcal{P},\mathcal{V}}$ is similarly defined for $\mathcal{V}$. The PPT interactive protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ is a ***zero-knowledge proof*** for a language $\mathsf{L} \in NP$ with knowledge error κ and zero-knowledge distance ε if the following properties hold.

- **Completeness**: If $x \in \mathsf{L}$ and $R(x, w) = 1$ for some witness w, then $\mathsf{out}^{\mathcal{V}}_{\mathcal{P},\mathcal{V}}(x, w, z) = 1$ for all strings z with overwhelming probability ν.
- **Soundness**: For any polynomial-time program $\mathcal{P}^*$

  $$\pi_{x,w,z} = \mathsf{Prob}[\mathsf{out}^{\mathcal{V}}_{\mathcal{P}^*,\mathcal{V}}(x, w, z) = 1].$$

  A protocol $\langle \mathcal{P}, \mathcal{V} \rangle$ satisfies soundness if for all $\mathcal{P}^*$ there exists a probabilistic Turing machine (PTM) program K, called a ***knowledge extractor*** with the following property. Suppose that

  $$\widetilde{\pi}_{x,w,z} = \mathsf{Prob}[K(x, w, z) = w' : R(x, w') = 1].$$

  Then it holds that $\pi_{x,w,z}$ is nonnegligible implies that $\widetilde{\pi}_{x,w,z}$ is nonnegligible.
- **(Statistical) Zero-Knowledge** (SZK): For each polynomial-time program $\mathcal{V}^*$, there is a PTM program S, called the ***simulator***, such that for all x, w with $R(x, w) = 1$, the random variables $S(x, z)$ and $\mathsf{out}^{\mathcal{V}^*}_{\mathcal{P},\mathcal{V}^*}(x, w, z)$ are statistically indistinguishable for all strings z:

  $$\forall \mathcal{A} \left| \mathsf{Prob}[\mathcal{A}(S(x, z)) = 1] - \mathsf{Prob}[\mathcal{A}(\mathsf{out}^{\mathcal{V}^*}_{\mathcal{P},\mathcal{V}^*}(x, w, z)) = 1] \right| < \varepsilon.$$

Completeness is very similar to correctness. Assuming both the prover and verifier follow the protocol faithfully, completeness guarantees that the protocol will succeed with a sufficiently high probability.

The intention of soundness ensures that the protocol will fail when executed by a prover using a false witness and an honest verifier. This is a minimal requirement. The formal definition above asserts something must stronger. It guarantees that a knowledge extractor K can derive a valid witness from any convincing prover. This implies K should have some more power than the verifier. In particular, K has access to the program of the prover, something that the verifier does not (the verifier is a program that interacts with the prover, whereas the knowledge extractor is a program that is derived from the program of the prover).

We note that our formulation of soundness is a bit more restrictive (albeit much simpler) than previous formulations in the literature, as it will fail protocols that allow a substantial cheating probability for the prover (e.g., 1/2). In most interesting cases such protocols can be made to satisfy our definition through parallel or sequential repetition.

Intuitively, statistical zero-knowledge is a property that prohibits a verifier from extracting information from an honest prover. If the verifier is able to learn anything, there must be an algorithm that simulates the protocol without access to a witness. Moreover, the execution of the algorithm is indistinguishable from that of the protocol.

A weaker version of zero-knowledge is ***honest-verifier zero-knowledge*** (HVZK). Here it is assumed that the verifier executes the protocol faithfully, but makes additional computations. Specifically, this is captured in the definition above by restricting V^* to simulate the verifier V and in the end, simply output the whole communication transcript. Achieving this much weaker property is sometimes also referred to as *semi-honest* verifier zero-knowledge. Even though this relaxes the SZK specifications, it can be used to obtain zero-knowledge proofs in situations employing generic methods. Proving honest verifier zero-knowledge boils down to producing accepting protocol transcripts that are indistinguishable from the honest prover-verifier transcripts, without the use of a witness.

1.3 The Schnorr Protocol

One classic three-move protocol that exhibits the properties of a zero-knowledge proof is the Schnorr protocol, also know as the Σ-protocol.

$$\begin{tikzcd}
\mathcal{P} \arrow[r, "\textit{offer}"] & \mathcal{V} \\
{} & {} \arrow[l, "\textit{challenge}"'] \\
{} \arrow[r, "\textit{response}"] & {}
\end{tikzcd}$$

The Schnorr protocol operates over a cyclic group $G = \langle g \rangle$ of order m. From our previous discussion, $\mathcal{P}$ and $\mathcal{V}$ have group generators $\langle p, m, g \rangle$. The prover $\mathcal{P}$ chooses a witness $w \in \mathbb{Z}_m$ such that $h = g^w \bmod p$ for some $h \in \langle g \rangle$. The verifier $\mathcal{V}$ is given p, m, g and h, and must confirm that $w = \log_g h$.

This can also be described as a language. Define $\mathsf{DLOG} = \{\langle\langle p, m, g\rangle, h\rangle : h = g^w \bmod p \text{ for some } w \in \mathbb{Z}_m\}$. ($\mathsf{DLOG}$ stands for "discrete logarithm".) Under the Schnorr protocol, there is a very efficient way to prove any statement $\langle\langle p, m, g\rangle, h\rangle$ is in DLOG without revealing $w = \log_g h$.

1. $\mathcal{P}$ choses $t \overset{\mathtt{r}}{\leftarrow} \mathbb{Z}_m$ and sends $y = g^t$ to $\mathcal{V}$.
2. $\mathcal{V}$ chooses a challenge $c \overset{\mathtt{r}}{\leftarrow} \mathbb{Z}_m$ and sends c to $\mathcal{P}$.
3. $\mathcal{P}$ computes $s = t + wc \bmod m$ and sends s to $\mathcal{V}$. $\mathcal{V}$ checks and accepts if and only if $g^s = yh^c$.

If both the prover and the verifier are honest, it holds that

$$g^s = g^{t+wc} = g^t (g^w)^c = yh^c.$$

The Schnorr protocol is therefore complete and can always convince an honest verifier.

This protocol instigates a special case of the soundness property. Before we formalize an extraction algorithm, let us look at how we can obtain information from a convincing prover $\mathcal{P}$.

Suppose we are capable of generating two accepting conversations from $\mathcal{P}$ with the challenge values $c \neq c'$: $\langle y, c, s\rangle$ and $\langle y, c', s'\rangle$. If both s and s' are valid, then $g^s = yh^c$ and $g^{s'} = yh^{c'}$. By solving both equations for y we obtain

$$\begin{aligned}
y = g^s h^{-c} &= g^{s'} h^{-c'} \\
h^{c-c'} &= g^{s-s'} \\
h &= g^{(s-s')/(c-c')}
\end{aligned}$$

While this does not justify how we can reverse-engineer $\mathcal{P}$ to obtain the second conversation, it does show that we can extract the witness as $(s-s')/(c-c') \bmod m$. We will assume we have access to $\mathcal{P}$ in a way that allows us to stop at any given point, return to a previous step, and re-simulate the operation.

It is helpful to view the probabilistic program $\mathcal{P}$ in two steps:

1. $\mathcal{P}(\mathsf{first}, \langle p, m, g\rangle, h)$ outputs $\langle y, \mathsf{aux}\rangle$
2. $\mathcal{P}(\mathsf{second}, \langle p, m, g\rangle, c, \mathsf{aux})$ outputs $\langle s\rangle$

where aux represents the internal information $\mathcal{P}$ uses, but does not publish. Using this, we can develop a knowledge extractor K with the following structure:

1. Let $\rho_1 \xleftarrow{\mathrm{r}} \{0,1\}^{\lambda_1}$ be the coin tosses required by the first step of $\mathcal{P}$. Fix the randomness of $\mathcal{P}$ with ρ_1 and simulate $\mathcal{P}(\mathsf{first}, \langle p, m, g\rangle, h)$ to obtain y.
2. Choose $c \xleftarrow{\mathrm{r}} \mathbb{Z}_m$.
3. Let $\rho_2 \xleftarrow{\mathrm{r}} \{0,1\}^{\lambda_2}$ be the coin tosses required by the second step of $\mathcal{P}$. Simulate $\mathcal{P}(\mathsf{second}, \langle p, m, g\rangle, c, \mathsf{aux})$ with fixed randomness ρ_2 to obtain s.
4. Choose $c' \xleftarrow{\mathrm{r}} \mathbb{Z}_m$ and $\rho_2' \xleftarrow{\mathrm{r}} \{0,1\}^{\lambda_2}$. Repeat steps 2 and 3 to obtain s'; output $\langle y, c, s\rangle$ and $\langle y, c', s'\rangle$.

If the knowledge extractor obtains two accepting conversations, we can directly reconstruct the witness as previously discussed. It remains to show that the knowledge extractor produces two accepting conversations with an adequate probability. To prove this, we need the following lemma.

Lemma 1.3.1 (Splitting Lemma)**.** *Let X and Y be finite sets. Call $A \subseteq X \times Y$ the set of* ***good elements*** *of $X \times Y$. Suppose there is a lower bound on the number of good elements such that*

$$|A| \geq \alpha\, |X \times Y|\,.$$

Define the set of ***super-good elements*** *A' to be the subset of A such that*

$$A' = \left\{(x,y) \in A \colon k_x > \frac{\alpha}{2}\, |Y|\right\}$$

where k_x is the number of $y \in Y$ such that $(x,y) \in A$ for a fixed x. Then

$$|A'| \geq \frac{\alpha}{2}\, |X \times Y|\,.$$

Proof. We prove this by contradiction. Assume $|A'| \,/\, |X \times Y| < \alpha/2$, so

$$|A| = |A'| + |A \setminus A'| < \frac{\alpha}{2}\, |X \times Y| + |A \setminus A'|\,. \tag{1}$$

For any $(x,y) \in A \setminus A'$, we have that $k_x \leq (\alpha/2)\, |Y|$. Since there are only $|X|$ distinct xs, $|A \setminus A'| \leq (\alpha/2)\, |X|\, |Y|$. From (1) we now obtain

$$|A| < \frac{\alpha}{2}\, |X \times Y| + \frac{\alpha}{2}\, |X|\, |Y|\,.$$

This contradicts the lower bound on $|A|$, so $|A'| \geq \alpha/2\, |X \times Y|$.

■

Returning now to the efficiency of our knowledge extractor K, define

$$X \times Y = \left\{(\rho_1, (c, \rho_2)) \colon \rho_1 \in \{0,1\}^{\lambda_1}\,,\ (c, \rho_2) \in \mathbb{Z}_m \times \{0,1\}^{\lambda_2}\right\}.$$

If the prover is successful with at least probability α, we define A to be the set of $(\rho_1, (c, \rho_2))$ that the verifier accepts. Then $|A| \geq \alpha\, |X \times Y|$. This suggests we can fix a good sequence $(\rho_1, (c, \rho_2))$ in A so that the resulting conversation from K is accepting. By Lemma 1.3.1, K hits a super-good sequence in steps 1 through 3 with probability $\alpha/2$.

Suppose the knowledge extractor does hit a super-good sequence. Then there is again an $\alpha/2$ probability that K hits another super-good sequence when repeating in Step 4. The probability that both conversations are accepting is therefore $\alpha^2/4$. Moreover, there is only a $1/m$ chance that K will generate the same challenge values $c = c'$.

Consider now the following: let S be the event that the knowledge extractor is successful. Next let C be the event that $c \neq c'$ in the second choice, let D be the event that the sequence $(\rho_1, (c, \rho_2))$ is super-good, and let E be the event that the sequence $(\rho_1, (c', \rho_2'))$ is good.

It follows that

$$\mathsf{Prob}[S] \geq \mathsf{Prob}[C \wedge D \wedge E] = \mathsf{Prob}[D \wedge E] - \mathsf{Prob}[\neg C] = \frac{\alpha^2}{4} - \frac{1}{m}.$$

This proves that the Schnorr protocol satisfies the soundness property.

Our three-move protocol satisfies honest-verifier zero-knowledge. To show this, we present an algorithm capable of simulating an accepting conversation between an honest prover and a (semi) honest verifier. Suppose that when given the public predicate information $\langle\langle p, m, g\rangle, h\rangle$, the simulator S randomly chooses c and s from $\mathbb{Z}_m$ and outputs $\langle g^s h^{-c}, c, s\rangle$. Recall that the output in the honest model is $\langle g^t, c, t + wc \bmod m\rangle$ where $t, c \overset{r}{\leftarrow} \mathbb{Z}_m$. One can easily verify that the two probability distributions are identical; thus, HVZK holds.

How to go beyond HVZK. While HVZK is a relatively weak property, it is useful in extending protocols to SZK. There are two generic ways to do this employing two kinds of commitment schemes, both of which rely on the apparent importance of the prover choosing y independent from c.

In the first method, the verifier is the first to send a message. Before receiving y, $\mathcal{V}$ chooses c and computes $(c', \sigma) \leftarrow \mathsf{Commit}(c)$. $\mathcal{V}$ then sends c' to the prover. When $\mathcal{P}$ sends y, the verifier returns (c, σ) to open the commitment. If $\mathsf{Verify}(c, c', \sigma) = 0$, the prover stops the protocol; otherwise the protocol is completed using the challenge c.

The commitment scheme satisfies the binding property, so $\mathcal{V}$ cannot change c. Statistical zero-knowledge therefore holds. The hiding property is satisfied by the commitment scheme, so $\mathcal{P}$ cannot obtain any information about c so soundness is not violated. A simulator for the zero-knowledge property must then be able to extract c from c'. This is called the ***extractable*** property for a commitment.

In the second method, the prover is once again the first to send a message. After computing y, $\mathcal{P}$ computes $(y', \sigma) \leftarrow \mathsf{Commit}(y)$ and sends y' to $\mathcal{V}$. Once $\mathcal{V}$ returns c, $\mathcal{P}$ sends (y, σ, s) to open the commitment. If $\mathsf{Verify}(y, y', \sigma) = 0$, the verifier stops the protocol.

Since the commitment scheme satisfies the hiding property, y' contains no useful information about y. $\mathcal{V}$ is therefore forced to pick c independent from y as desired. This scheme satisfies the zero-knowledge property.

Note that soundness is not violated under this scheme because the binding property prohibits $\mathcal{P}$ from opening y in two different ways. In this setting then, a simulator must be able to bypass the binding property on the commitment; that is, if the simulator commits an arbitrary y^*, after receiving the challenge c, it can choose $y = g^s h^{-c}$. Commitments that allow this type of violation are called ***equivocal***.

Observe that the first scheme added a new move to the protocol, whereas the second maintained the standard three-move structure. For this reason, the second scheme is sometimes preferred. Still the above discussion merely reduces the problem to the design of commitment schemes with either the equivocal or the extractable property for the simulator.

1.4 Non-Interactive Zero-Knowledge Proofs

We now introduce a non-interactive version of the Schnorr identification scheme based on what is known as the ***Fiat-Shamir Heuristic***.

To make a proof non-interactive, we use a hash function $H\colon \{0,1\}^* \longrightarrow \mathbb{Z}_m$ such that a conversation $\langle y, c, s\rangle = \langle g^t, H(g^t), t + H(g^t)w \bmod m\rangle$. This mandates that c be selected after y, implicitly relying on the properties of the hash function.

To show that SZK holds, assume H is a random oracle controlled by the simulator. In the Random Oracle Model, a dishonest verifier $\mathcal{V}^*$ may make queries to the random oracle. Figure 2 illustrates how $\mathcal{V}^*$ interacts with H.

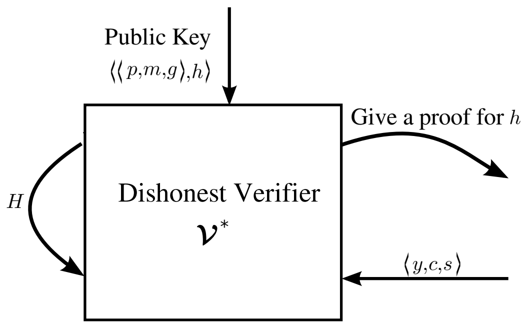

Figure 2: The simulation of a dishonest verifier $\mathcal{V}^*$ in the Random Oracle Model.

When the verifier asks for a proof of $h = g^w$, the simulator randomly chooses c and s to calculate $y = g^s h^{-c}$. It enters (y, c) in *History* and returns $\langle y, c, s\rangle$. The dishonest verifier cannot distinguish between an honest prover and the simulator unless $(y, c') \in History$ with $c \neq c'$. Then $\mathcal{V}^*$ succeeds with probability $(1/m)q_H$, where q_H is the number of random oracle queries.

Next consider soundness in the Random Oracle Model. As with the Schnorr protocol, we want to generate two accepting conversations with the same y and different challenge values. Using these two conversations, we can extract a witness. Note that $c = H(y)$. If a dishonest prover $\mathcal{P}^*$ makes a single query to the random oracle before producing $\langle y, c, s\rangle$, the analysis is same as with the interactive protocol. Problems arise however when $\mathcal{P}^*$ makes more than one query.

Suppose in the initial round $\mathcal{P}^*$ makes q_H queries before outputting a conversation. When a knowledge extractor returns $\mathcal{P}^*$ to a previous step, there is no guarantee that $\mathcal{P}^*$ will again make q_H queries. When $\mathcal{P}^*$ terminates, it could return $\langle y', c', s'\rangle$ with $c' = H(y')$ and $y \neq y'$. This reduces our ability to extract a witness, so we must adjust the probability of obtaining two accepting conversations with the same y.

Assume that after making q_H queries, $\mathcal{P}^*$ chooses one query and uses the corresponding response from the random oracle in its output. Let $\mathsf{Prob}[A] = \alpha$ denote the probability that the resulting conversation is accepting. Let $\mathsf{Prob}[Q_i] = \beta_i$ represent the probability that the dishonest prover completes the ith conversation with $1 \leq i \leq q_H$. Define $\mathsf{Prob}[A \cap Q_i] = \alpha_i$, then

$$\sum_{i=1}^{q_H} \alpha_i = \alpha \qquad \text{and} \qquad \sum_{i=1}^{q_H} \beta_i = 1.$$

Define $\mathsf{Prob}[E]$ to be the probability of extracting a witness from $\mathcal{P}^*$. We then have that

$$\mathsf{Prob}[A \cap Q_i] = \mathsf{Prob}[A \mid Q_i] \cdot \mathsf{Prob}[Q_i]$$

so $\mathsf{Prob}[A \mid Q_i] = \alpha_i/\beta_i$. From our calculations in Section 1.3, we obtain

$$\mathsf{Prob}[E \mid Q_i] \geq \frac{\mathsf{Prob}[A \mid Q_i]^2}{4} - \frac{1}{m} = \frac{\alpha_i^2}{4\beta_i^2} - \frac{1}{m}.$$

The overall probability is calculated as follows.

$$\begin{aligned}
\mathsf{Prob}[E] &= \sum_{i=1}^{q_H} \mathsf{Prob}[E \mid Q_i] \cdot \mathsf{Prob}[Q_i] \\
&\geq \sum_{i=1}^{q_H} \left(\frac{\alpha_i^2}{4\beta_i^2} - \frac{1}{m} \right) \beta_i \\
&= \frac{1}{4} \sum_{i=1}^{q_H} \frac{\alpha_i^2}{\beta_i} - \sum_{i=1}^{q_H} \frac{\beta_i}{m} \\
&= \frac{1}{4} \sum_{i=1}^{q_H} \frac{\alpha_i^2}{\beta_i} - \frac{1}{m} \sum_{i=1}^{q_H} \beta_i \\
&= \frac{1}{4} \sum_{i=1}^{q_H} \frac{\alpha_i^2}{\beta_i} - \frac{1}{m} \\
&\geq \frac{1}{4q_H} \left(\sum_{i=1}^{q_H} \alpha_i \right)^2 - \frac{1}{m} \\
&= \frac{\alpha^2}{4q_H} - \frac{1}{m}
\end{aligned}$$

We can therefore conclude that given a convincing prover, we can extract a witness with probability at least

$$\frac{\alpha^2}{4q_H} - \frac{1}{m}.$$

1.5 Honest-Verifier Zero-Knowledge for all NP

Anything in NP can be proven in a three-move honest-verifier zero-knowledge protocol. Consider the language HC of all Hamiltonian graphs. Recall that a ***Hamiltonian cycle*** π is a path in a graph G that visits each vertex precisely once before returning to the starting point. HC is NP-complete, so a proof of knowledge for HC would provide a proof of knowledge for all NP problems: given any instance of a problem in NP, we can transform it into a graph with a Hamiltonian cycle if and only if the instance is a "Yes" instance. We can then use the HC proof for any NP problem.

A graph with n vertices can be represented by an $n \times n$ binary matrix called the graph's ***adjacency matrix***. If the ith vertex is connected to the jth vertex, the ij entry in the matrix is 1; otherwise it is 0. Given a permutation π over $\{1, ..., n\}$ and a graph G defined by its adjacency matrix (a_{ij}), we define the permuted graph G^π as the graph that has the adjacency matrix $(a'_{ij}) = (a_{\pi^{-1}(i)\pi^{-1}(j)})$. A Hamiltonian cycle in a graph can in fact be represented by a permutation π of the vertices with the special property that the graph G^π includes the edges $(1, 2), (2, 3), ..., (n-1, n), (n, 1)$.

If π is a Hamiltonian cycle for a graph G and π' is an arbitrary permutation, then $\pi'^{-1} \circ \pi$ is a Hamiltonian cycle for the graph $G^{\pi'}$.

HVZK proofs are used to verify that a Hamiltonian cycle corresponds to a given graph without revealing the cycle. Figure 3 demonstrates how such an HVZK proof is executed. Note that a dishonest prover can continue unnoticed with probability $\kappa = 1/2$. Suppose that a prover commits to a fake adjacency matrix such that she constructs a Hamiltonian cycle. If the verifier returns $c = 1$, the prover is able to convince the verifier that she knows a Hamiltonian cycle for the graph. But if the prover receives $c = 0$, the verifier learns that the prover did not commit to an accurate permutation of the graph. Through k repetitions however, we can decrease the knowledge error to $\kappa = 1/2^k$ and thus prove the soundness property.

$\mathcal{P}$(private input π)		$\mathcal{V}$
	The graph G	
Pick a random permutation π' of the graph G; commit to the adjacency matrix of $G^{\pi'}$		
	$\begin{bmatrix} \mathsf{com}_{11} & \cdots & \mathsf{com}_{1n} \\ \vdots & \ddots & \vdots \\ \mathsf{com}_{n1} & \cdots & \mathsf{com}_{nn} \end{bmatrix}$ $\longrightarrow$	
		$c \overset{r}{\leftarrow} \{0,1\}$
	$\xleftarrow{c}$	
If $c = 0$, open all commitments and send π'^{-1}; If $c = 1$, tell for each row which commitments need to be opened (depends on π)		
	$\xrightarrow{\text{Open } \mathsf{com} \text{ based on } c}$	
		Verify the commitments: If $c = 0$, check that the matrix equals $G^{\pi'}$; If $c = 1$, check that a cycle was revealed

Figure 3: A proof of knowledge for a Hamiltonian cycle. In addition to committing to the adjacency matrix of $G^{\pi'}$, the prover must make a separate commitment com_{ij} to each entry in the matrix.

1.6 The Conjunction of Two Zero-Knowledge Proofs

There are instances in which a prover wants to validate multiple statements through one interaction, either for efficiency or privacy. This can be done using a single challenge value to preserve the desirable three-move structure. Upon receiving the challenge, the prover combines the offers and responses as is seen in Figure 4.

Theorem 1.6.1. *The conjunction of two honest-verifier zero-knowledge proofs satisfies the soundness and HVZK properties.*

1.7 The Disjunction of Two Zero-Knowledge Proofs

In Section 1.1 we mentioned that some user-identification schemes contain directories cataloging statements of theorems assigned to each user. In such schemes, privacy issues may arise when users wish to gain access without explicitly identifying themselves. In the disjunction of two zero-knowledge proofs, the identification protocol asks the user $\mathcal{P}$ to provide witnesses to two statements. The user obviously knows a witness to one of the statements, but does not need to disclose which one. The system $\mathcal{V}$ sends a single challenge value which the prover uses to fabricate a witness to the second statement. Figure 5 depicts the execution of a disjunction.

Theorem 1.7.1. *The disjunction of two honest-verifier zero-knowledge proofs satisfies the soundness and HVZK properties.*

Notes by S. Pehlivanoglu, J. Todd, & H.S. Zhou

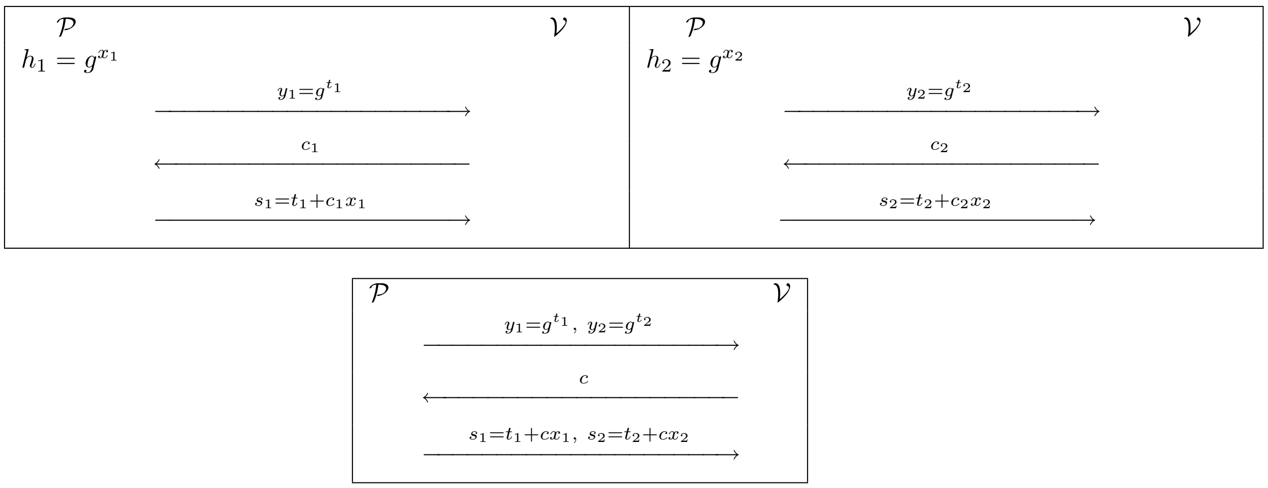

Figure 4: The conjunction of two zero-knowledge proofs for the discrete logarithm.

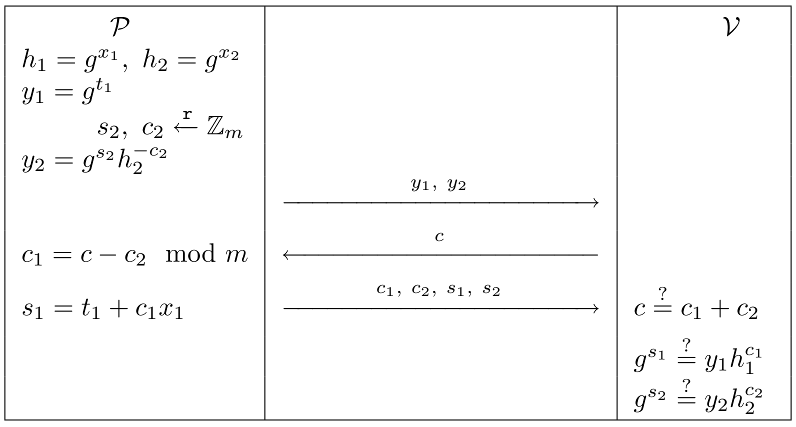
$\mathcal{P}$
$\mathcal{V}$
$h_1 = g^{x_1},\ h_2 = g^{x_2}$
$y_1 = g^{t_1}$
$s_2,\ c_2 \xleftarrow{\mathtt{r}} \mathbb{Z}_m$
$y_2 = g^{s_2} h_2^{-c_2}$
$y_1,\ y_2$
c
$c_1 = c - c_2 \mod m$
$c_1,\ c_2,\ s_1,\ s_2$
$s_1 = t_1 + c_1 x_1$
$c \overset{?}{=} c_1 + c_2$
$g^{s_1} \overset{?}{=} y_1 h_1^{c_1}$
$g^{s_2} \overset{?}{=} y_2 h_2^{c_2}$

Figure 5: The disjunction of two zero-knowledge proofs for the discrete logarithm showing how the prover can show the verifier he knows one of the two discrete logarithms of h_1, h_2 (without revealing which one). In this case the prover $\mathcal{P}$ knows a witness for h_1.