# Zero-knowledge Proofs

Luciano Maino

University of Bristol - Advanced Cryptology

## 1 Introduction

Let's consider the following scenario. We have two identical balls, differing only in colour - one is green, and the other is red. Alice, who is red-green color-blind, is skeptical that these balls are truly different. We want to convince her that they are, but without revealing which ball is red. So, how can we achieve this?

We could play the following game. We hand both balls to Alice. She randomly selects one ball and shows it to us. Then, she takes the ball back and either shows us the same one or swaps it for the other. Alice knows whether she has switched the balls or kept the same one.

Our task is to tell her whether the ball she shows is the same as the one she showed before. If we couldn't distinguish between the two balls, we would have to guess randomly, succeeding with probability of $1/2$.

Being fooled with probability $1/2$ is actually not satisfactory for Alice, so she repeats the same game again. This time, if we were trying to cheat, our chance of guessing correctly would drop to $1/4$. After repeating this process $n$ times, the probability of fooling Alice lowers down to $1/2^n$. It doesn't take long to convince Alice they are different.

This is an example of what is in cryptography a *Zero-knowledge proof*. A Zero-knowledge proof is a protocol in which a *prover* needs to convince a *verifier* that a certain statement is true, without revealing any additional information about this statement.

## 2 $\Sigma$ protocols

Let's first introduce some formalism.

**Definition 1 (Informal).** *Let $X$ and $W$ be two non-empty sets. A* relation $\mathcal{R}$ *is a subset of $X \times W$. A language $\mathcal{L}$ for $\mathcal{R}$ is subset of $X$ such that, for each $x \in \mathcal{L}$, there exists a $w \in W$ verifying $(x, w) \in \mathcal{R}$. Given $(x, w) \in X \times W$, we will assume that checking $(x, w) \in \mathcal{R}$ is "efficient", whereas given an $x \in \mathcal{L}$, it is "hard" to compute any $w \in W$ such that $(x, w) \in \mathcal{R}$.*

*Example 2.* Let $E$ be an elliptic curve and let $P$ be a point on $E$ such that the ECDLP is a hard problem (for instance, Curve25519). Also, let $\ell$ denote the order of $P$.

$$\mathcal{R} = \{([n]P, n) \mid n \in \{0, \ell - 1\}\} \qquad \text{and} \qquad \mathcal{L} = \{[n]P \mid n \in \{0, \ell - 1\}\}\,.$$

The scenario we have described in Section 1 is an example of *interactive proof system*. An interactive proof system allows a prover $\mathcal{P}$ to prove to a verifier $\mathcal{V}$ something they know. In cryptography, the notion of "prover and verifier" is usually captured modelling them as probabilistic polynomial time (PPT) algorithms. The most important example of interactive proof system is the $\Sigma$ *protocol*.

In a $\Sigma$ protocol, a prover $\mathcal{P}$ wants to prove the knowledge of $(x, w) \in \mathcal{R}$ to a verifier $\mathcal{V}$. This is achieved engaging in a three-round protocol. The prover $\mathcal{P}$ consists of two PPT algorithms and $P_1, P_2$, whereas the verifier $\mathcal{V}$ is described by the PPT algorithm $V$.

First, the prover $\mathcal{P}$ computes a commitment $\mathsf{com}$ running $P_1$ on input $(x, w)$. After receiving the datum $\mathsf{com}$, the verifier $\mathcal{V}$ samples a random $\mathsf{chl}$ from the challenge space $\Omega$ and sends it over to the prover. The challenge space must be large enough to prevent that an attacker could guess the challenges beforehand.

The prover now runs the algorithm $P_2$ on input $(x, w, \mathsf{com}, \mathsf{chl})$ and obtains a response $\mathsf{rsp}$. The verifier finally runs $V$ on $(x, \mathsf{com}, \mathsf{chl}, \mathsf{rsp})$ and returns 1 if they are convinced about the fact that $\mathcal{P}$ knows a witness for the statement $x$; otherwise they output 0. This interactive protocol is summarised in Figure 1.

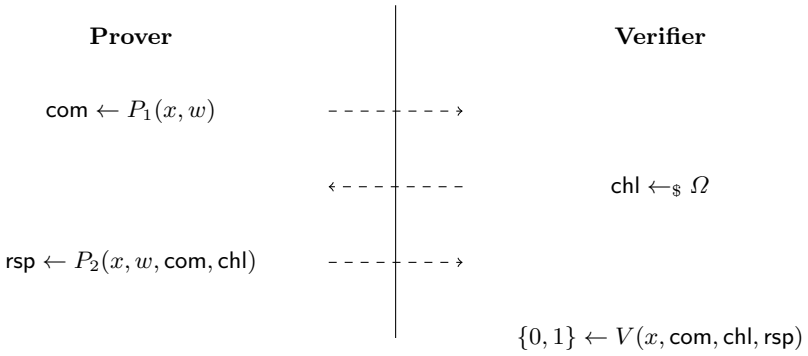


**Fig. 1.** Diagram describing $\Sigma$ protocols.

The tuple $(\mathsf{com}, \mathsf{chl}, \mathsf{rsp})$ is a *transcript* for the interactive protocol. A $\Sigma$ protocol must verify the following properties.

**Completeness:** If the prover $\mathcal{P}$ follows the prescribed protocol, then the verifier $\mathcal{V}$ will output 1 with overwhelming probability.

**Special Soundness:** There exists a PPT algorithm $\mathcal{E}xtr$, called the *extractor*, capable of extracting witnesses from statements as follows. Let $x$ be a statement in $\mathcal{L}$ and let $(\mathsf{com}, \mathsf{chl}, \mathsf{rsp})$ and $(\mathsf{com}, \mathsf{chl}', \mathsf{rsp}')$ be two transcripts such that $\mathsf{chl} \neq \mathsf{chl}'$, $1 \leftarrow V(x, \mathsf{com}, \mathsf{chl}, \mathsf{rsp})$ and $1 \leftarrow V(x, \mathsf{com}, \mathsf{chl}', \mathsf{rsp}')$. Then $\mathcal{E}xtr(x, \mathsf{com}, \mathsf{chl}, \mathsf{rsp}, \mathsf{chl}', \mathsf{rsp}')$ returns a witness $w$ such that $(x, w) \in \mathcal{R}$.

**Honest-Verifier Perfect Zero-Knowledge:** There exists a PPT $\mathcal{S}im$, called the *simulator*, that, on input a statement $x \in \mathcal{L}$, can create a valid transcript $(\mathsf{com}, \mathsf{chl}, \mathsf{rsp})$ for $x$. Moreover, the distribution of its outputs is identical to the distribution of the transcripts obtained via honest interactions between $\mathcal{P}$ and $\mathcal{V}$.

*Remark 3.* It is worth noting that if a $\Sigma$ protocol is Honest Verifier Perfect Zero-Knowledge, it doesn't mean that the simulator $\mathcal{S}im$ is capable of interacting and convincing a verifier $\mathcal{V}$.

The transcripts generated by $\mathcal{S}im$ are generated adaptively. For instance, the $\mathcal{S}im$ could first choose a challenge $\mathsf{chl}$ and then craft an appropriate commitment $\mathsf{com}$.

## 3 Schnorr protocol

We now give a concrete instantiation of a $\Sigma$ protocol, the famous *Schnorr protocol*. Let $E$ be an elliptic curve and let $P$ be a point on $E$ such that the ECDLP is a hard problem. The Schnorr protocol is a $\Sigma$ protocol for the language in Example 2.

Let $(Q, n) \in \mathcal{R}$. The Schnorr protocol is described in Figure 2. We now prove it is complete, special sound and honest-verifier perfect zero-knowledge.

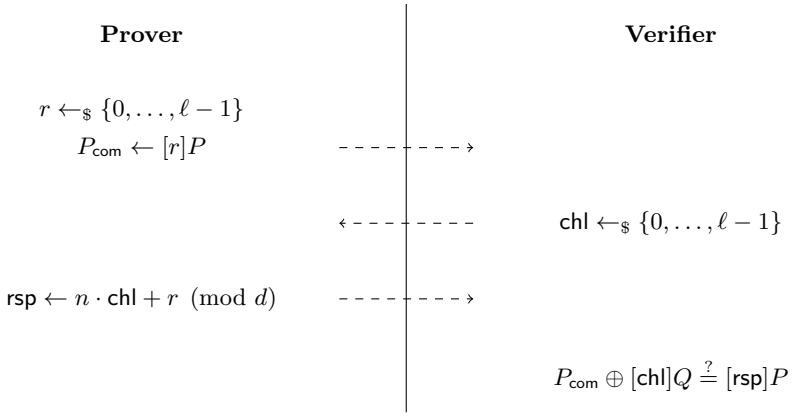


**Fig. 2.** Schnorr protocol

**Completeness:** We will use the same notation as in Figure 2. Given the transcript $(P_{\mathsf{com}}, \mathsf{chl}, \mathsf{rsp})$ we have

$$P_{\mathsf{com}} \oplus [\mathsf{chl}]Q = [r]P \oplus [\mathsf{chl}][n]P = [r + \mathsf{chl} \cdot n]P,$$

from which the correctness follows.

**Special Soundness:** We now have to show the existence of an efficient extractor. Let $(P_{\mathsf{com}}, \mathsf{chl}, \mathsf{rsp})$ and $(P_{\mathsf{com}}, \mathsf{chl}', \mathsf{rsp}')$ be two transcripts for the pair $(Q, n)$ such that $\mathsf{chl} \neq \mathsf{chl}'$.

Recall that to have that the ECDLP is hard, we must have that the order of $P$ is a large prime $\ell$. This means that the quantity $\mathsf{chl} - \mathsf{chl}' \neq 0 \pmod{\ell}$ is invertible modulo $\ell$. We claim that $\frac{\mathsf{rsp}-\mathsf{rsp}'}{\mathsf{chl}-\mathsf{chl}'} \pmod{\ell}$ is a valid witness for $Q$, i.e.

$$\left[\frac{\mathsf{rsp} - \mathsf{rsp}'}{\mathsf{chl} - \mathsf{chl}'} \pmod{\ell}\right] P = Q,$$

which is equivalent to showing that

$$[\mathsf{rsp} - \mathsf{rsp}']\, P = \left[\mathsf{chl} - \mathsf{chl}'\right] Q.$$

We observe that

$$[\mathsf{rsp} - \mathsf{rsp}']\, P = P_{\mathsf{com}} \oplus [\mathsf{chl}]Q \oplus [-1]P_{\mathsf{com}} \oplus [-\mathsf{chl}']Q,$$

from which our claim follows.

To summarise, we have constructed an extractor that, given $(P_{\mathsf{com}}, \mathsf{chl}, \mathsf{rsp})$ and $(P_{\mathsf{com}}, \mathsf{chl}', \mathsf{rsp}')$, returns $\frac{\mathsf{rsp}-\mathsf{rsp}'}{\mathsf{chl}-\mathsf{chl}'} \pmod{\ell}$, which can be done efficiently.

**Honest-Verifier Perfect Zero-Knowledge:** To prove this property, we need to exhibit an efficient simulator. The simulator works in this way. The first step is to uniformly sample a $\mathsf{chl}$ from the challenge space $\Omega$. Then, the simulator also samples a random $z \leftarrow_{\$} \{0, \ldots, \ell - 1\}$. The commitment will be given by $P_{\mathsf{com}} = [z]P \oplus [-\mathsf{chl}]Q$, whereas the response is given by $\mathsf{rsp} = z$.

It is clear that the transcript $(P_{\mathsf{com}}, \mathsf{chl}, \mathsf{rsp})$ will be accepted by the verifier $\mathcal{V}$. We now have to argue that this transcript is indistinguishable from all the other honestly generated transcripts. First, we note that the distribution over the challenges is the same. Also, the distribution of $n \cdot \mathsf{chl} + r \pmod{\ell}$ for a uniformly sampled $r$ is the same as the one of a random $z$ sampled from $\{0, \ldots, \ell - 1\}$. The final step is then to observe that the subgroup generated by $P$ is isomorphic to $\mathbb{Z}/\ell\mathbb{Z}$, from which it derives the claim on the indistinguishability of the transcripts output by our simulator.

## 4 Fiat-Shamir Heuristic

A $\Sigma$ protocol is an interactive protocol between two parties, a prover and a verifier. This protocol can be made non-interactive using a transformation called the *Fiat-Shamir transform*. The main idea is to replace the challenge data sent by the verifier with something else, which can be computed without interaction. In practice, this "something else" is a hash function $\mathcal{H}$ that generates challenges depending on the commitment.

It turns out, that any $\Sigma$ protocol verifying the properties above can be transformed into an EUF-CMA digital signature scheme. We briefly recall that a digital signature scheme is existentially unforgeable under an adaptive chosen-message attack if, even

when an attacker can query an oracle to obtain valid signatures for chosen messages, the attacker is still unable to generate a valid signature for any message not previously queried to the oracle.

More formally, let $\mathcal{A}$ be a PPT adversary playing the following game.

1. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathbf{KeyGen()}$;
2. Given $\mathsf{pk}$, the adversary $\mathcal{A}$ chooses $\widetilde{\mathcal{M}} = \{m_1, \ldots, m_n\}$ and queries an oracle which outputs the signatures of the messages in $\widetilde{\mathcal{M}}$.
3. The adversary $\mathcal{A}$ wins the game if they are able to sign a message not in $\widetilde{\mathcal{M}}$.

A signature scheme is EUF-CMA if no PPT adversary $\mathcal{A}$ can win the above game with non-negligible probability.

We now explain how to obtain a digital signature scheme from the Schnorr protocol. Roughly speaking, the secret key is the witness corresponding to a statement, which plays the role of the public key.

**KeyGen()**

1. $\mathsf{sk} \leftarrow_{\$} \{0, \ldots, \ell - 1\}$.
2. $\mathsf{pk} \leftarrow [\mathsf{sk}]P$.
3. **return** $(\mathsf{sk}, \mathsf{pk})$.

To sign a message $m$ contained in a certain space $\mathcal{M}$, we need to have a "robust" hash function $\mathcal{H} \colon \mathcal{M} \times E \to \{0, \ldots, \ell - 1\}$.

**Sign**$(m, \mathsf{sk}, \mathsf{pk})$

1. $r \leftarrow_{\$} \{0, \ldots, \ell - 1\}$.
2. $P_{\mathsf{com}} \leftarrow [r]P$.
3. $\mathsf{chl} \leftarrow \mathcal{H}(m, P_{\mathsf{com}})$.
4. $\mathsf{rsp} \leftarrow \mathsf{sk} \cdot \mathsf{chl} + r$
5. **return** $(P_{\mathsf{com}}, \mathsf{rsp})$.

To verify that a message $m$ has been correctly signed, the verifier has to first generate the challenge themselves and then run the verification process as in the $\Sigma$ protocol.

**Verify**$(m, \mathsf{pk}, P_{\mathsf{com}}, \mathsf{rsp})$

1. $\mathsf{chl}' \leftarrow \mathcal{H}(m, P_{\mathsf{com}})$.
2. **return** $P_{\mathsf{com}} \oplus [\mathsf{chl}']Q \stackrel{?}{=} [\mathsf{rsp}]$.

*Remark 4.* It is worth noting that the Schnorr protocol is *commitment recoverable*. It means that given a correct $(\mathsf{chl}, \mathsf{rsp})$, it is possible to recover $P_{\mathsf{com}}$. Sending $\mathsf{chl}$ usually incur lower bandwidth requirements. The verification can then be modified to have first recover $P_{\mathsf{com}}$ and then check that the hash output coincide with the challenge $\mathsf{chl}$.

## 5 Additional Readings

[2] “Cryptography Made Simple”, Smart; Chapter 21.
[1] "Introduction to Modern Cryptography", Katz and Lindell; Section 12.5.
[3] “Proofs, Arguments, and Zero-Knowledge”, Thaler.

## References

1. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, Second Edition. CRC Press (2014)
2. Smart, N.P.: Cryptography Made Simple. Springer Publishing Company, Incorporated (2015)
3. Thaler, J.: Proofs, Arguments, and Zero-Knowledge, vol. 4. Foundations and Trends® in Privacy and Security (2022)