Tecniche di Sicurezza Informatica dei Dati e delle Reti May 21, 2015

Zero-Knowledge Proofs and Applications

Guest Lecturer: Daniele Venturi *Lecturer: Antonio Villani*

Abstract

The material below covers two lectures on the beautiful and influential concept of *zero-knowledge proofs*. This notion, introduced by Goldwasser, Micali and Rackoff [GMR85] formalizes the idea of a proof that "yields nothing but its validity".[1]

We will start by describing a simple running example, which will allow us to abstract away some basic properties. This will lead to the concept of Σ-protocols, and their application to construct secure identification schemes. Next, we will move to cryptographic applications. In particular we will show how to use zero-knowledge proofs to construct efficient (and provably secure) identification and signature schemes. Finally, we will formalize the definition of zero-knowledge and survey the main results about constructing zero-knowledge proofs for all NP.

The topics covered and the exposition, are inspired by [Dam10, HL10, Ven12]. Comments, questions and corrections can be sent by email to:

Daniele Venturi
Via Salaria 113, 00198 Rome, Italy
website: `http://wwwusers.di.uniroma1.it/~venturi`
email: `venturi@di.uniroma1.it`.

Contents

[1]For this and others influential results, Shafi Goldwasser and Silvio Micali received the prestigious ACM Turing Award in 2012. Read more at `http://www.acm.org/press-room/awards/turing-award-12`.

$$\begin{array}{lcl}
\textbf{Alice}(\mathbb{G}, y, x) & & \textbf{Bob}(\mathbb{G}, y = g^x) \\
\hline
a \leftarrow\!\!\$\; \mathbb{Z}_q; \alpha = g^a & & \\
& \xrightarrow{\alpha} & \\
& & \beta \leftarrow\!\!\$\; \mathbb{Z}_q \\
& \xleftarrow{\beta} & \\
\gamma = \beta \cdot x + a & & \\
& \xrightarrow{\gamma} & \\
& & \text{check } y^\beta \cdot \alpha = g^\gamma
\end{array}$$

Figure 1: The Schnorr protocol for proving "knowledge" of a discrete logarithm.

1 A Running Example

We start by introducing a protocol due to Schnorr [Sch89], and later abstract away several properties that will be useful in the sequel.

1.1 The Schnorr Protocol

Let p be a prime, and take a generator g of a subgroup $\mathbb{G}$ of $\mathbb{Z}_p^*$ of prime order q. We should think of $\mathbb{G}$ as a group where computing discrete logarithms is believed to be hard (for proper choice of the parameters). The Schnorr protocol can be used to "prove knowledge" of a discrete logarithm x of some value $y = g^x \in \mathbb{G}$. The protocol is shown in Fig. 1.

1.2 Basic Properties

We now put forward a series of intuitive requirements we would like to be satisfied by the above protocol. To facilitate generalizing the Schnorr protocol, we will talk about so-called NP relations $\mathcal{R} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ naturally defining a language $\mathcal{L}_\mathcal{R} := \{y : \exists x \text{ s.t. } \mathcal{R}(y,x) = 1\}$. We will refer to Alice as the prover—modelled as a PPT algorithm P—and to Bob as the verifier—modelled as a PPT algorithm V; the prover holds a witness x for a value $y \in \mathcal{L}$, and his goal is to convince the verifier of this fact via an interactive protocol at the end of which the verifier outputs a judgement `yes`/`no`. We will denote the interaction as $\mathsf{P}(y,x) \rightleftarrows \mathsf{V}(y)$; each interaction yields a transcript $\tau \in \{0,1\}^*$, i.e. a sequence of messages (e.g., the values (α, β, γ) in Fig. 1).

Completeness. The first property we would like to be satisfied is a correctness requirement, namely, whenever Alice knows a valid witness she should always be able to convince Bob.

Definition 1 (Completeness)**.** *We say that* (P, V) *satisfies completeness relative to an NP relation* $\mathcal{R}$*, if for all* $y \in \mathcal{L}_\mathcal{R}$ *we have that* $\mathsf{P}(y,x) \rightleftarrows \mathsf{V}(y) = \texttt{yes}$ *with probability one (over the randomness of all involved algorithms).*

The lemma below says that the Schnorr protocol satisfies completeness.

Lemma 1. *The protocol of Fig. 1 satisfies completeness relative to the relation* $\mathcal{R}_{\mathsf{DL}}^{\mathbb{G}}(y,x) = 1$ *iff* $y = g^x$ *(for* g *a generator of* $\mathbb{G}$*).*

Proof.
$$g^\gamma \equiv g^{\beta\cdot x+a} \equiv g^a\cdot(g^x)^\beta = y^\beta\cdot\alpha \pmod p.$$
□

Knowledge soundness. The second natural property is that it should be hard to convince Bob of the validity of a false statement. It turns out that this can be defined in several ways; the definition below aims at capturing the requirement that in order to convince the verifier it is necessary to actually *know* a witness.

Definition 2 (Knowledge soundness). *We say that (P,V) is a proof-of-knowledge (PoK) relative to an NP relation $\mathcal{R}$, if for any (possibly unbounded*[2]*) malicious prover P^* such that $\mathsf{P}^*(y) \rightleftarrows \mathsf{V}(y) = \mathtt{yes}$ with probability larger than ε, there exists a PPT knowledge extractor K (with rewinding black-box access to P^*) such that $\mathsf{K}^{\mathsf{P}^*}(y)$ returns a value x satisfying $\mathcal{R}(y,x)=1$ with probability polynomial in ε.*

The original (slightly more precise) definition is due to Bellare and Goldreich [BG92] (see also [HL10, Definition 6.3.1]). Let us look again at the protocol of Fig. 1 in order to get some intuition. Roughly, if some P^* having sent α could answer two *different* challenges β,β' correctly, this would mean that it could produce γ,γ' such that $g^\gamma = y^\beta\cdot\alpha \bmod p$ and $g^{\gamma'} = y^{\beta'}\cdot\alpha \bmod p$. Dividing one equation by the other, we get that $g^{\gamma-\gamma'} = y^{\beta-\beta'} \bmod p$; since $\beta\neq\beta' \bmod q$ we get that $(\beta-\beta')$ can be inverted modulo q, and thus $y = g^{(\gamma-\gamma')\cdot(\beta-\beta')^{-1}} \bmod p$ yielding $x = \frac{\gamma-\gamma'}{\beta-\beta'} \bmod q$. So loosely speaking, a cheating prover who does not know x can only be able to answer at most one challenge value correctly, since otherwise the argument we just gave would imply that it was in fact able to compute x after all.

The above discussion leads to the definition of *special soundness*, which is tailored to 3-round interactive protocols.

Definition 3 (Special soundness). *Let (P,V) be a 3-round protocol (for a relation $\mathcal{R}$) with transcripts of the form (α,β,γ), where the prover speaks first. We say that (P,V) satisfies special soundness if the following holds: for any $y\in\mathcal{L}_\mathcal{R}$ and any pair of accepting conversations (α,β,γ) and (α,β',γ') on input y with $\beta\neq\beta'$, we can efficiently compute x such that $\mathcal{R}(y,x)=1$.*

The above discussion yields the following result:

Lemma 2. *The protocol of Fig. 1 satisfies special soundness.*

The lemma below says that, for 3-round protocols, special soundness implies knowledge soundness.

Theorem 1. *Let (P,V) be a 3-round interactive protocol for a relation $\mathcal{R}$, with challenge space $\mathcal{B}$. Then, if (P,V) satisfies special soundness it also satisfies knowledge soundness whenever $|\mathcal{B}| = \omega(\log\kappa)$ for security parameter $\kappa\in\mathbb{N}$.*

Proof. Assume there exists (an unbounded) P^* such that $\mathbb{P}[\mathsf{P}^*(y) \rightleftarrows \mathsf{V}(y) = \mathtt{yes}] \geq \varepsilon$. We construct a PPT K (using P^*) as follows:

<u>Knowledge Extractor K:</u>

1. Choose the coins $r \leftarrow_\$ \{0,1\}^*$ for P^*, and sample $\beta,\beta' \leftarrow_\$ \mathbb{Z}_q$.

[2] If we restrict the malicious prover to be computationally bounded, we speak of *arguments* of knowledge (AoK).

2. Run P^* twice, the first time using β and the second time using β'.
3. In case both runs succeeds, use (α, β, γ) and $(\alpha, \beta', \gamma')$ to recover x (via special soundness, see Definition 3).

Define ε_y to be the probability that P^* succeeds in convincing V upon input y; note that $\varepsilon = \mathbb{E}_y[\varepsilon_y]$. Now fix y and define $\varepsilon_{y,r}$ to be the probability that P^* succeeds (for the fixed y) upon input random coins r. Let $\mathbf{H}$ be the 0/1-matrix with a row for each possible random tape for P^* and one column for each possible challenge value. An entry $\mathbf{H}(r, \beta)$ is 1 if V accepts for random tape r and challenge β, and 0 otherwise. All we know is that $\varepsilon_{y,r}$ is equal to the fraction of 1-entries in $\mathbf{H}$; note that $\varepsilon_y = \mathbb{E}_r[\varepsilon_{y,r}]$.

Letting R be the total number of choices for the randomness of P^*, we can write

$$\begin{aligned}
\mathbb{P}[\mathsf{K} \text{ succeeds on } y] &= \sum_r \frac{1}{R}\varepsilon_{y,r}\left(\varepsilon_{y,r} - \frac{1}{|\mathcal{B}|}\right) && (1)\\
&= \mathbb{E}_r\left[\varepsilon_{y,r}^2\right] - \frac{1}{|\mathcal{B}|}\mathbb{E}_r\left[\varepsilon_{y,r}\right]\\
&\geq \left(\mathbb{E}_r\left[\varepsilon_{y,r}\right]\right)^2 - \frac{\varepsilon_y}{|\mathcal{B}|} && (2)\\
&= \varepsilon_y^2 - \frac{\varepsilon_y}{|\mathcal{B}|}.
\end{aligned}$$

Eq. (1) follows by the fact that $\beta \neq \beta'$ with all but $1/|\mathcal{B}|$ probability, and Eq. (2) follows by Jensen's inequality. Hence, we have obtained

$$\begin{aligned}
\mathbb{P}[\mathsf{K} \text{ succeeds}] &\geq \mathbb{E}_y\left[\varepsilon_y^2 - \frac{\varepsilon_y}{|\mathcal{B}|}\right] && (3)\\
&\geq \varepsilon^2 - \frac{\varepsilon}{|\mathcal{B}|},
\end{aligned}$$

where Eq. (3) follows again by Jensen's inequality. The theorem now follows by observing that the above probability is polynomial in ε whenever $|\mathcal{B}| = \omega(\log \kappa)$. □

Note that the extractor constructed in the proof of the above theorem is not very efficient, in that to ensure success probability roughly 1/2 we need to run the extractor roughly $1/\varepsilon^2$ times. This can be improved to $1/\varepsilon$ runs, but we will not show it here. We refer the interested reader to [HL10, Theorem 6.3.2].

Honest Verifier Zero Knowledge. Until now we did not consider how much information a proof leaks on the witness. In fact, note that any NP relation has a trivial interactive proof which consists of transmitting the witness in the clear. We would like to define formally what it means for a proof to reveal nothing beyond its validity. This will (gradually) lead to the concept of zero knowledge.

Definition 4 (Honest Verifier Zero Knowledge)**.** *We say that* (P, V) *satisfies HVZK if there exists an efficient simulator* S *which can generate a tuple* (α, β, γ) *with the same distribution as the corresponding tuples in a honest protocol run between Alice and Bob.*

Intuitively, the above definition guarantees that *honest* protocol executions do not reveal anything about the witness as their distribution can be emulated without knowing the witness. Sometimes the simulated distribution is not identical to the real ones, but only computationally indistinguishable; this is still fine for applications at the price of a negligible term in the concrete bounds.

Lemma 3. *The protocol of Fig. 1 satisfies perfect HVZK.*

Proof. The simulator S, given $y \in \mathbb{G}$, simply samples $\beta, \gamma \leftarrow\!\!\$\, \mathbb{Z}_q$, and computes $\alpha := g^\gamma \cdot y^{-\beta}$. It is easy to see that such (α, β, γ) has the right distribution. To see this, note that β is uniform as in a real execution. Furthermore, since g^γ is uniform in $\mathbb{G}$ and β is also uniform, we get that $\alpha = g^\gamma \cdot y^{-\beta}$ is uniform as in a real execution. Finally γ is in both cases a deterministic function of α and β, concluding the proof. □

Clearly, HVZK does not say anything about the case where a *malicious* verifier tries to learn something on the witness by deviating from the protocol. This extreme adversarial setting leads to the definition of full-fledged zero knowledge (see Section 5). We remark that the Schnorr protocol is not known to be zero knowledge: there might be some efficient malicious strategy that the verifier may follow which, perhaps after many executions of the protocol, enables it to "steal" the witness.

2 Σ-Protocols

By collecting all notions introduced so far, we arrive at the definition of Σ-protocols which is given below.[3]

Definition 5 (Σ-protocol)**.** *A Σ-protocol for a relation $\mathcal{R}$ is a 3-round interactive protocol with transcripts (α, β, γ), where the prover speaks first, satisfying the following properties:*

Completeness. *The same as in Definition 1.*

Special Soundness. *The same as in Definition 3.*

Special HVZK. *There exists a PPT simulator S that on input $y \in \mathcal{L}_\mathcal{R}$ and for all β outputs a tuple (α, β, γ) with the same distribution as the corresponding tuple in a honest protocol run between Alice and Bob with input y.*

By Theorem 1 we know that every Σ-protocol with super-logarithmic challenge space is also a PoK. Moreover, putting together Lemma 1—3 we also know that the Schnorr protocol is a Σ-protocol.

Note that Special HVZK allows the simulator to also take as input the challenge, which is not allowed in HVZK (see Definition 4). However, most Σ-protocols satisfy special HVZK; in particular the protocol of Fig. 1 does (the proof is left as an exercise). One can also show that any relation in NP has a (computational[4]) Σ-protocol, but we will not prove this.

The proof of the statement below is left as an exercise.

Theorem 2. *Let (P, V) be a Σ-protocol for a relation $\mathcal{R}$. Then:*

[3]The terminology Σ-protocol was introduced for the first time by Ronald Cramer in his PhD thesis [Cra96].

[4]This means that the HVZK property holds only computationally, i.e., the simulator's output distribution is indistinguishable from the distribution of honest executions.

$$\begin{array}{lcl}
\textbf{Alice}(x, y_0, y_1) & & \textbf{Bob}(y_0, y_1) \\
\hline
\alpha_b \leftarrow_{\$} \mathsf{P}(x, y_b); \beta_{1-b} \leftarrow_{\$} \{0,1\}^\ell & & \\
(\alpha_{1-b}, \beta_{1-b}, \gamma_{1-b}) \leftarrow_{\$} \mathsf{S}(y_{1-b}, \beta_{1-b}) & & \\
 & \xrightarrow{\alpha_0, \alpha_1} & \\
 & & \beta \leftarrow_{\$} \{0,1\}^\ell \\
 & \xleftarrow{\beta} & \\
\beta_b = \beta \oplus \beta_{1-b} & & \\
\gamma_b \leftarrow_{\$} \mathsf{P}(x, \alpha_b, \beta_b) & & \\
 & \xrightarrow{\beta_0, \beta_1, \gamma_0, \gamma_1} & \\
 & & \text{check: } \beta = \beta_0 \oplus \beta_1 \\
 & & \qquad \mathsf{V}(y_0, (\alpha_0, \beta_0, \gamma_0)) = \mathtt{yes} \\
 & & \qquad \mathsf{V}(y_1, (\alpha_1, \beta_1, \gamma_1)) = \mathtt{yes}
\end{array}$$

Figure 2: The OR trick.

(i) All properties of (P, V) *are maintained under parallel composition.*

(ii) For any $\ell > 0$ *there exists a* Σ*-protocol with challenges of length* ℓ *bits.*

2.1 OR Proofs

We now explain the OR trick, due to Cramer, Damgård and Shoenmakers [CDS94]. Consider a relation $\mathcal{R}$ (with corresponding language $\mathcal{L} := \mathcal{L}_{\mathcal{R}}$) with a Σ-protocol (P, V) having ℓ-bit challenges. The protocol of Fig. 2 allows a prover—holding $y_0, y_1 \in \mathcal{L}$, together with a witness x for either y_0 or y_1—to convince a verifier that either $\mathcal{R}(y_0, x) = 1$ or $\mathcal{R}(y_1, x) = 1$ *without revealing which is the case.* Define the relation $\mathcal{R}_{\mathsf{OR}}$, such that $\mathcal{R}_{\mathsf{OR}}((y_0, y_1), x)) = 1$ iff either $\mathcal{R}(y_0, x) = 1$ or $\mathcal{R}(y_1, x) = 1$. We show the following theorem.

Theorem 3. *The protocol of Fig. 2 is a* Σ*-protocol for the relation* $\mathcal{R}_{\mathsf{OR}}$*. Moreover, for any (possibly malicious) verifier* V^**, the probability distribution of conversations between* P *and* V^**, where* x *is such that* $\mathcal{R}(y_b, x) = 1$*, is independent of* b*.*

Proof. We proceed to verify that the protocol satisfies all properties as required in Definition 5.

Completeness. This follows by completeness of (P, V), and by the fact that if the prover is honest we always have $\beta = \beta_0 \oplus \beta_1$.

Special soundness. To prove special soundness, consider two accepting conversations on input (y_0, y_1)

$$((\alpha_0, \alpha_1), \beta, (\beta_0, \gamma_0, \beta_1, \gamma_1)), ((\alpha_0, \alpha_1), \beta', (\beta_0', \gamma_0', \beta_1', \gamma_1')) \text{ such that } \beta \neq \beta'.$$

Since $\beta_0 \oplus \beta_1 = \beta \neq \beta' = \beta_0' \oplus \beta_1'$, it follows that either $\beta_0 \neq \beta_0'$ or $\beta_1 \neq \beta_1'$ (or both). Without loss of generality assume that $\beta_0 \neq \beta_0'$. We can now use special soundness of (P, V) to compute x from $(\alpha_0, \beta_0, \beta_0', \gamma_0, \gamma_0')$, such that $\mathcal{R}(y_0, x) = 1$; clearly x is also a valid witness for $\mathcal{R}_{\mathsf{OR}}$.

Special HVZK. Given β and (y_0, y_1) the simulator S_{OR} simply chooses β_0, β_1 at random subject to $\beta_0 \oplus \beta_1 = \beta$ and hence runs the simulator S of the underlying Σ-protocol twice, on inputs (y_0, β_0) and (y_1, β_1).

It remains to prove the second part of the statement. Towards this, consider an arbitrary verifier V^* interacting with (honest) P in the protocol of Fig. 2. We note that the distribution of conversations has the form $((\alpha_0, \alpha_1), \beta, (\beta_0, \gamma_0, \beta_1, \gamma_1))$, where (α_0, α_1) are distributed as an honest prover in (P, V) would choose them (this follows from HVZK of the underlying protocol). Then β has whatever distribution V^* outputs, given $y_0, y_1, \alpha_0, \alpha_1$; moreover β_0, β_1 are random subject to $\beta = \beta_0 \oplus \beta_1$. Finally, γ_0 has whatever distribution the honest prover in (P, V) outputs, given that the input was y_0 and the first part of the conversation was α_0, β_0; a similar conclusion holds for γ_1. This is trivial for γ_b and follows from HVZK of the underlying protocol for γ_{1-b}. We conclude that the distribution is independent of b, as desired. □

The last property is also known as *witness indistinguishability* (WI). One can prove that every Σ-protocol satisfies WI, but we will not do this. We refer the reader, e.g., to [Ven12, Lemma 13.7]. The OR trick described above can be generalized to the case where there are polynomially many relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$ and $\mathcal{R}_{\mathsf{OR}} := \mathcal{R}_1 \vee \cdots \vee \mathcal{R}_n$. It is also possible to give a Σ-protocol for $\mathcal{R}_{\mathsf{AND}} := \mathcal{R}_1 \wedge \cdots \wedge \mathcal{R}_n$. The two things together yield Σ-protocols for any monotone function.

2.2 Witness Hiding

As our next step, we want to formalize what it means for an interactive proof to "hide the witness" This will require Σ-protocols for so-called *hard* relations, which we introduce next.

Definition 6 (Hard relation)**.** *A relation $\mathcal{R}$ is said to be hard if there exists a PPT algorithm* Gen *that takes as input a security parameter $\kappa \in \mathbb{N}$ and outputs a pair (y, x) such that $\mathcal{R}(y, x) = 1$. Moreover, for all PPT adversaries* A*, there exists a negligible*[5] *function $\nu : \mathbb{N} \to [0, 1]$ such that*

$$\mathbb{P}\left[\mathcal{R}(y, x^*) = 1 : \; x^* \leftarrow \mathsf{A}(y); (y, x) \leftarrow \mathsf{Gen}(1^\kappa)\right] \leq \nu(\kappa).$$

Intuitively the above captures the requirement that it should be hard to find a witness for a valid statement $y \in \mathcal{L}_{\mathcal{R}}$. Assuming that computing discrete logs in $\mathbb{G}$ is hard, for instance, the relation $\mathcal{R}_{\mathsf{DL}}$ introduced in Section 1 is hard.

We now argue that the protocol of Fig. 2 has a property known as *witness hiding* (WH), whenever the underlying relation is hard. Intuitively we want to say that even a cheating verifier V^* cannot, after running the protocol with an honest prover, learn sufficient information in order to recover a valid witness.

Definition 7 (Witness hiding)**.** *Let $\mathcal{R}$ be a hard relation with Σ-protocol* (P, V)*. We say that* (P, V) *is witness hiding if for all PPT* V^* *there exists a negligible function $\nu : \mathbb{N} \to [0, 1]$ such that* $\mathbb{P}[\mathsf{V}^* \text{ wins}] \leq \nu(\kappa)$ *in the following game:*

- *Run* $(y, x) \leftarrow \mathsf{Gen}(1^\kappa)$ *and give y to* V^**.*
- V^* *can run polynomially many executions* $\mathsf{P}(y, x) \rightleftarrows \mathsf{V}^*(y)$*, and learn the corresponding transcripts.*

[5] A function $\nu : \mathbb{N} \to [0, 1]$ is negligible in κ if it decreases faster than any inverse polynomial in κ, i.e. $\nu(\kappa) = \kappa^{-\omega(1)}$.

- *Finally* V^* *outputs* x^* *and it is said to win if* $\mathcal{R}(y, x^*) = 1$.

Theorem 4. *Let* $\mathcal{R}$ *be a hard relation. The protocol* $(\mathsf{P}_{\mathsf{OR}}, \mathsf{V}_{\mathsf{OR}})$ *of Fig. 2 is WH for* $\mathcal{R}_{\mathsf{OR}}$.

Proof. Assume there exists a PPT V^* that wins with non-negligible probability in the game below:

- For a random bit $b \leftarrow\$ \{0,1\}$, run $(y_0, x_0), (y_1, x_1) \leftarrow \mathsf{Gen}(1^\kappa)$, give (y_0, y_1) to V^*, and keep x_b.
- V^* can run polynomially many executions $\mathsf{P}_{\mathsf{OR}}(x_b, y_0, y_1) \rightleftarrows \mathsf{V}^*(y_0, y_1)$, and learn the corresponding transcripts.
- Finally V^* outputs x^* and it is said to win if $\mathcal{R}_{\mathsf{OR}}((y_0, y_1), x^*) = 1$ (i.e., either $\mathcal{R}(y_0, x^*) = 1$ or $\mathcal{R}(y_1, x^*) = 1$).

We construct a PPT A using V^* to break the hardness of $\mathcal{R}$ as follows:

Reduction A:

- At the beginning A receives y, such that $(y, x) \leftarrow \mathsf{Gen}(1^\kappa)$, from its own challenger.
- Run $(\bar{y}, \bar{x}) \leftarrow \mathsf{Gen}(1^\kappa)$, sample a random bit $b \leftarrow\$ \{0,1\}$, and define $y_b := y$ and $y_{1-b} := \bar{y}$. Return (y_0, y_1) to V^*.
- In case V^* wants to interact with the prover in P_{OR}, simply emulate the interaction honestly (using knowledge of $\bar{x}$).
- Whenever V^* terminates and outputs x^*, output x^*.

Notice that, by WI of $(\mathsf{P}_{\mathsf{OR}}, \mathsf{V}_{\mathsf{OR}})$, we have that V^* has no information on our choice of b. Since b was chosen at random, we obtain that $\mathcal{R}(y, x^*) = 1$ with probability $1/2$. It follows that A succeeds with non-negligible probability $\nu/2$, a contradiction. This concludes the proof. $\square$

3 Secure Identification

We now sketch an application of what we have seen so far to the context of secure identification (ID) schemes.

3.1 Detour I: Passive/Active Security

An ID scheme is a tuple of algorithms $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ specified as follows: (i) Algorithm Gen takes as input the security parameter and outputs a key pair (pk, sk); (ii) (P, V) defines an interactive protocol—where the prover holds (pk, sk) and the verifier holds pk—at the end of which the verifier outputs a judgement $\texttt{yes}/\texttt{no}$. As usual, we say that (P, V) satisfies correctness if $\mathsf{P}(pk, sk) \rightleftarrows \mathsf{V}(pk) = \texttt{yes}$ with probability one over the choice of $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$ and over the randomness of all involved algorithms.

The most basic form of security for an ID scheme is called *passive security*, and intuitively captures the requirement that it should be hard to convince the verifier without knowing the secret key (even after observing polynomially many *honest* executions of the ID scheme).

Definition 8 (Passive security)**.** *We say* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *is a passively secure ID-scheme if for all PPT adversaries* A *there exists a negligible function* $\nu : \mathbb{N} \to [0,1]$ *such that* $\mathbb{P}[\mathsf{A}\ \textit{wins}] \leq \nu(\kappa)$ *in the following game:*

1. *Run* $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$*, and give* pk *to* A*.*
2. *The adversary has access to an oracle returning polynomially many transcripts* τ *of honest executions* $\mathsf{P}(pk, sk) \rightleftarrows \mathsf{V}(pk)$*.*
3. *The adversary interacts with the verifier, and is said to win iff* $\mathsf{A}(pk) \rightleftarrows \mathsf{V}(pk) = \mathtt{yes}$*.*

In *active security*, it should be hard to impersonate the prover even if the adversary can previously replace the verifier in polynomially many protocol executions.

Definition 9 (Active security)**.** *We say* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *is an actively secure ID-scheme if for all PPT adversaries* A *there exists a negligible function* $\nu : \mathbb{N} \to [0,1]$ *such that* $\mathbb{P}[\mathsf{A}\ \textit{wins}] \leq \nu(\kappa)$ *in the following game:*

1. *Run* $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$*, and give* pk *to* A*.*
2. *The adversary can replace the verifier in polynomially many executions of the protocol, yielding transcripts* τ *corresponding to* $\mathsf{P}(pk, sk) \rightleftarrows \mathsf{A}(pk)$*.*
3. *The adversary interacts with the verifier, and is said to win iff* $\mathsf{A}(pk) \rightleftarrows \mathsf{V}(pk) = \mathtt{yes}$*.*

3.2 Constructions from Σ-Protocols

A way to construct an ID scheme, is to use directly a Σ-protocol for a hard relation $\mathcal{R}$. The idea is to generate a pair $(y, x) \leftarrow \mathsf{Gen}(1^\kappa)$ and to define the public key of a user to be $pk := y$, while the secret key is $sk := x$. An execution of the protocol is then defined by running the underlying Σ-protocol.

Theorem 5. *Let* (P, V) *be a Σ-protocol for a hard relation* $\mathcal{R}$*, with challenge space of size* $|\mathcal{B}| = \omega(\log \kappa)$*. Then* (P, V) *is a passively secure ID scheme.*

Proof. Denote by $\mathbf{G}$ the security experiment of Definition 8. Consider a modified game $\mathbf{G}'$ where the transcripts $\tau := (\alpha, \beta, \gamma)$ corresponding to honest protocol executions are emulated by running the HVZK simulator S of the underlying Σ-protocol.

By (perfect) HVZK we have that $\mathbf{G}$ and $\mathbf{G}'$ are identically distributed, in particular $\mathbb{P}[\mathsf{A}$ wins in $\mathbf{G}'] = \mathbb{P}[\mathsf{A}$ wins in $\mathbf{G}]$. We now show that for all PPT A the probability that A wins in $\mathbf{G}'$ is negligible. By contradiction, assume there exists an adversary A such that $\mathbb{P}[\mathsf{A}$ wins in $\mathbf{G}']$ is non-negligible. We construct an adversary B (using A) breaking hardness of the underlying relation.

Adversary B:

- Receive the value y (for the hard relation), set $pk := y$ and give pk to A.
- Run the knowledge extractor $\mathsf{K}^{\mathsf{A}}(y)$; denote with x^* the output of the extractor.
  - While running the extractor, whenever A asks to see a honest execution of the protocol, run $\tau := (\alpha, \beta, \gamma) \leftarrow \mathsf{S}(y)$ and return τ to A.

- Output x^*.

By knowledge soundness (which follows by special soundness as $|\mathcal{B}| = \omega(\log \kappa)$), we get that B outputs a value such that $\mathcal{R}(y, x^*) = 1$ with non-negligible probability. This concludes the proof. □

In case the Σ-protocol additionally satisfies the WH property, it turns out the ID scheme actually achieves *active security.* The intuition here is that if an adversary A could break active security with non-negligible probability, by knowledge soundness we could extract a secret key sk^* that is consistent with pk with non-negligible probability; this contradicts the WH property.

Theorem 6. *Let* (P, V) *be a* Σ*-protocol (with challenge space of size* $|\mathcal{B}| = \omega(\log \kappa)$*) for a hard relation* $\mathcal{R}$*, that additionally satisfies the witness hiding property. Then* (P, V) *is an actively secure ID scheme.*

Proof. Assume that there exists an adversary A that wins the active security game with non-negligible probability. We construct an adversary B (using A) that breaks the witness hiding property of the underlying Σ-protocol.

Adversary B:

- Receive the value y (for the hard relation), set $pk := y$ and give pk to A.
- Run the knowledge extractor $\mathsf{K}^{\mathsf{A}}(y)$; denote with x^* the output of the extractor.
  - While running the extractor, whenever A wants to play the role of the verifier in an execution of the ID scheme, forward A's messages to the challenger and relay back the challenger's messages to A.
- Output x^*.

By knowledge soundness (which follows by special soundness as $|\mathcal{B}| = \omega(\log \kappa)$), we get that B outputs a value such that $\mathcal{R}(y, x^*) = 1$ with non-negligible probability. This concludes the proof. □

As a corollary, the OR proof from Section 2.1 directly yields an actively secure ID scheme.

3.3 Beyond Active Security

In practice, active security might not be enough. In particular an adversary could play a *man-in-the-middle* between some user and a honest verifier, and just relay communication in order to impersonate the user to that verifier. In order to avoid this, we need to ensure that not only provers, but also verifiers, hold certified public keys. See [Dam10, Section 7] for a more in-depth discussion how this can be achieved.

As a final note, we remark that there are other ways to construct secure ID schemes. We refer the reader, e.g., to [Ven12, Chapter 11] for a survey. One popular such way is to use a signature scheme: The verifier challenges the user on a randomly chosen message, and the prover provides a signature on that message. On the positive side, this approach requires only two rounds (and not three as for Σ-protocols). On the negative side, a signature could be used later on as a proof of the fact that a given user talked to the verifier. This problem does not appear when using Σ-protocols: A verifier can always simulate a transcript of the protocol only given the public key, so this cannot be used as evidence that it really talked to some user.

4 The Fiat-Shamir Heuristic

In this section we show a powerful application of Σ-protocols to the construction of efficient and provably secure signature schemes. The result is in the random oracle model [BR93] (ROM), which assumes the existence of a random hash function $h \leftarrow \mathcal{H}$ that answers to every query with a (truly) random response chosen uniformly from its output domain; if a query is repeated it responds the same way every time that query is submitted.

4.1 Detour II: Universal Unforgeability

We take a detour and recall the standard security notion for signature schemes. A signature scheme $\mathcal{SS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is triple of algorithms defined as follows: (i) Algorithm KGen takes as input a security parameter, and outputs a pair (vk, sk) where vk is the public (verification) key and sk is the secret (signing) key; (ii) Algorithm Sign is randomized, takes as input the signing key sk, a message $m \in \{0,1\}^*$ and random coins r, and returns a signature $\sigma := \mathsf{Sign}(m; r)$; (iii) Algorithm Vrfy takes as input vk, m, σ and outputs a decision bit.

We say that $\mathcal{SS}$ satisfies correctness if for all $m \in \{0,1\}^*$, and for all $(vk, sk) \leftarrow_{\$} \mathsf{KGen}(1^\kappa)$, we have that $\mathsf{Vrfy}(vk, (m, \mathsf{Sign}(sk, m))) = 1$ with probability one.

As for security, we require that it should be hard to forge a signature on a "fresh" message even given access to an oracle that signs polynomially many (and adaptively chosen) messages.

Definition 10 (Unforgeability)**.** *We say that $\mathcal{SS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is unforgeable against chosen-message attacks if for all PPT forgers F there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that $\mathbb{P}[\mathsf{F}\ wins] \le \nu(\kappa)$ in the following game:*

1. *Sample $(vk, sk) \leftarrow_{\$} \mathsf{KGen}(1^\kappa)$ and give vk to F.*
2. *The forger can specify polynomially many signing queries of the form $m \in \{0,1\}^*$; upon each query it receives back $\sigma \leftarrow_{\$} \mathsf{Sign}(sk, m)$.*
3. *At some point F outputs a pair (m^*, σ^*) and is said to win if $\mathsf{Vrfy}(vk, (m^*, \sigma^*)) = 1$ and m^* was not asked as a signing query.*

4.2 The Fiat-Shamir Transform

The Fiat-Shamir transform [FS86] allows to turn every Σ-protocol (for a hard relation $\mathcal{R}$) into a signature scheme. The transformation is depicted in Fig. 3, and can be cast as follows: (i) Algorithm KGen samples $h \leftarrow_{\$} \mathcal{H} := \{h : \{0,1\}^* \to \{0,1\}^\ell\}$, runs the generation algorithm of the underlying hard relation to obtain a pair (y, x) such that $\mathcal{R}(y, x) = 1$, and defines $vk := (y, h)$ and $sk := x$; (ii) Algorithm Sign emulates a run of the Σ-protocol by computing the challenge β as hash of the values (y, m, α), and returns (α, γ) as signature on m; (iii) Algorithm Vrfy first recovers β computing $h(y, m, \alpha)$, and finally checks that (α, β, γ) is valid.

The proof below is facilitated by assuming (P, V) is a passively secure ID scheme (which is the case whenever the relation $\mathcal{R}$ is hard). Recall that this means that no PPT adversary P^* (not knowing a witness) can make the verifier accept, even after observing polynomially many honest executions of the protocol.

$$\begin{array}{lcl}
\textbf{Alice}(vk := (y,h), sk := x) & & \textbf{Bob}(vk := (y,h)) \\
\hline
\alpha \leftarrow_{\$} \mathsf{P}(x,y) & & \\
\beta := h(m,\alpha,y) & & \\
\gamma \leftarrow_{\$} \mathsf{P}(x,\alpha,\beta) & & \\
 & \xrightarrow{\sigma:=(\alpha,\gamma)} & \\
 & & \text{check: } \beta = h(m,\alpha,y) \\
 & & \qquad \mathsf{V}(y,(\alpha,\beta,\gamma)) = \mathtt{yes}
\end{array}$$

Figure 3: The Fiat-Shamir transformation, turning a Σ-protocol (P,V) into a signature scheme $\mathcal{SS}_{\mathsf{FS}}$.

Theorem 7. *Let (P,V) be a 3-round passively secure ID scheme, such that the first message α has conditional min-entropy*[6] *$\omega(\log\kappa)$ given the public key y. Then the signature scheme $\mathcal{SS}_{\mathsf{FS}}$ is universally unforgeable against chosen-message attacks in the random oracle model.*

Proof. Correctness follows by the completeness property of the underlying Σ-protocol. As for unforgeability, consider a forger F with non-negligible advantage in the unforgeability game. We assume F makes q_s signature queries $(m_i)_{i\in[q_s]}$, and q_h random oracle queries $(\hat{m}_i,\hat{\alpha}_i,y)_{i\in[q_h]}$. Wlog. we can assume that if F outputs $(m^*,\sigma^* := (\alpha^*,\beta^*,\gamma^*))$, then it must have queried the random oracle on (m^*,α^*,y); in particular, $(\hat{m}_{i^*},\hat{\alpha}_{i^*}) = (m^*,\alpha^*)$ for some $i^* \in [q_h]$. We construct an adversary P^* (using F) breaking passive security of the underlying ID scheme.

<u>Reduction P^*:</u>

- Receive the public key $y \in \mathcal{L}$, sample $j \leftarrow_{\$} \{1,\ldots,q_h\}$, and forward y to F; observe q_s honest executions of $\mathsf{P}(y,x) \rightleftarrows \mathsf{V}(y)$, yielding transcripts $(\alpha_i,\beta_i,\gamma_i)_{i\in[q_s]}$.
- Upon input a query $(\hat{m}_i,\hat{\alpha}_i,y)$ to the random oracle (where $i \neq j$), if $h(\hat{m}_i,\hat{\alpha}_i,y)$ is already defined return this value; else return $\hat{\beta}_i \leftarrow_{\$} \{0,1\}^{\ell}$.
- Upon input a signature query on message m_i, program the random oracle at $h(\alpha_i,m_i,y) = \beta_i$; if the random oracle is already defined at that point, abort.
- Upon input query $(\hat{m}_j,\hat{\alpha}_j,y)$ to the random oracle, forward $\hat{\alpha}_j$ to the challenger, receiving a challenge $\hat{\beta}_j$; program the random oracle at $h(\hat{\alpha}_j,\hat{m}_j,y) = \hat{\beta}_j$.
- Whenever F outputs $m^*,(\alpha^*,\beta^*,\gamma^*)$, if $(m^*,\alpha^*) \neq (\hat{m}_j,\hat{\alpha}_j)$ abort; else, forward γ^* to the challenger.

Define the event `Abort` to be the event that P^* aborts in the third step of the above reduction; by our assumption on the min-entropy of α we know that $\mathbb{P}[\mathtt{Abort}] \leq \nu'(\kappa)$ for some negligible function ν'. Moreover let `Guess` be the event that the reduction guesses correctly the index $j = i^*$. We have,

$$\begin{aligned}
\mathbb{P}[\mathsf{P}^* \text{ wins}] &\geq \mathbb{P}[\mathsf{F} \text{ wins} \wedge \mathtt{Guess} \wedge \neg\mathtt{Abort}] \\
&= (1-\mathbb{P}[\mathtt{Abort}]) \cdot \mathbb{P}[\mathsf{F} \text{ wins} \wedge \mathtt{Guess} | \neg\mathtt{Abort}] \\
&\geq \frac{1}{q_h} \cdot \mathbb{P}[\mathsf{F} \text{ wins}] - \nu(\kappa),
\end{aligned}$$

a non-negligible quantity. □

[6]The min-entropy of a random variable X is defined as $\mathbb{H}_\infty(X) := -\log\max_x \mathbb{P}[X = x]$.

4.3 The Random Oracle Controversy

Do random oracles exist in the real world? Not that we know. Clearly, a proof is never better than the assumption from which a statement is proven. So what is a random oracle proof useful for? The answer to this question is often debated in the community.

On the negative side, there exists (albeit contrived) protocols that are secure in the random oracle model but can never be secure for any possible instantiation of the hash function. For instance, Goldwasser and Kalai [GK03] proved that there exists a 3-round *argument* such that the Fiat-Shamir transformation is always insecure when used with that argument. While this still leaves the possibility that the Fiat-Shamir transformation could work for all 3-round *proofs*, [BDG$^+$13] have shown that this is also "implausible".[7]

On the positive side, a proof in the random oracle is much better than no proof at all! It also means that a cryptographic scheme has no fundamental flaws, and that, if a vulnerability ever exists, it can only regard the hash function. Said that, if available, a standard model proof is usually preferred.

5 Zero-Knowledge Proofs

As we have seen, HVZK means that *honest* protocol executions do not reveal anything on the witness (in the sense that they can be simulated efficiently without knowing a witness). WI (or more in general WH), instead, require that a *malicious* verifier cannot distinguish protocol executions with either of two witnesses (or learn anything useful which allows to *compute* a valid witness). Roughly speaking, full-fledged zero knowledge means that even for *malicious* verifiers protocol executions do not reveal anything on the witness.

Definition 11 (Zero knowledge)**.** *Let* (P, V) *be an interactive protocol for a relation* $\mathcal{R}$*. We say that* (P, V) *is zero knowledge if for all PPT* V^* *there exists a PPT simulator* S *such that*

$$\{\mathsf{P}(y, x) \rightleftarrows \mathsf{V}^*(y)\}_{y \in \mathcal{L}_\mathcal{R}} \approx \{\mathsf{S}^{\mathsf{V}^*}(y)\}_{y \in \mathcal{L}_\mathcal{R}},$$

where "$\approx$" can mean computational, statistical, or perfect indistinguishability.

Typically, the definition requires that V^* and S are also given some auxiliary input (which might possibly depend on x); this is important when ZK protocols are used as sub-protocols in larger protocols.

Are Σ-protocols ZK? It seems hard to prove, at least if the challenge space is $\kappa^{\omega(1)}$. In order to see this, consider again the Schnorr protocol (see Fig. 1) and the following malicious verifier V^*: At the beginning V^* receives the value $\alpha = g^a$ (for random a) from the prover, and then it defines $\hat{\beta} = H(\alpha)$ where $H(\cdot)$ is a collision resistant hash function; let $\hat{\gamma} = \hat{\beta} \cdot x + a$ be the prover's last message. The simulator S has to simulate a conservation $(\alpha, \hat{\beta}, \hat{\gamma})$ such that $\hat{\beta} = H(\alpha)$, and moreover $g^{\hat{\gamma}} = \alpha \cdot y^{\hat{\beta}}$. It is not clear how to do this. A first possibility would be to choose $\hat{\beta}$ and $\hat{\gamma}$, and then hope that the value α satisfies $H(\alpha) = \hat{\beta}$; this would violate the hash function property. A second possibility would be to choose α (which determines $\hat{\beta}$); but then finding $\hat{\gamma}$ would require to compute a discrete logarithm.

[7] As positive result, [BLV06] put forward a simple property of the hash function that allows to prove soundness of Fiat-Shamir in the standard model; we do not know any hash function satisfying this property though.

The good news is that every Σ-protocol can be transformed into a protocol that is full-fledged zero knowledge (and in fact, even into a ZK-PoK). However, we won't have time to explain how this can be done. The interested reader is referred, e.g., to [HL10, Section 6.5].

5.1 On the Role of Randomness and Interaction

One can show that both prover's randomness and interaction are necessary in order to construct ZK proofs. In particular, without making further assumptions, non-interactive ZK (NIZK) is possible only for "trivial languages".[8] The same holds for the case of deterministic provers. We refer the reader, e.g., to [Ven12, Section 13.2] for a proof.

Nevertheless, NIZK is possible in the random oracle model; in particular the FS transformation of Section 4 can be shown to yield a NIZK proof.

5.2 Zero Knowledge for NP

Many interesting relations have ZK proofs, but we won't have time to study this. For instance, there is a simple ZK proof for the problem of graph isomorphism. Let $G = (V, E)$ be a graph, where V denotes the set of the vertices and E the set of edges. Two graphs $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ are isomorphic if there exists a map $\varphi : V_0 \to V_1$ such that $(v, v') \in E_0$ iff $(\varphi(v), \varphi(v')) \in E_1$. The underlying language here is the set of all isomorphic graphs, the map φ being the corresponding witness.

Does any language in NP have a ZK proof? The answer to this question is positive (although we won't have time to show this). In particular it is sufficient to construct a ZK proof for any NP-complete problem. This was done for the first time by Goldreich, Micali and Widgerson [GMW86] assuming that one-way functions exist.

6 Further Reading

We conclude with a brief collection of pointers to other interesting applications of zero-knowledge proofs.

- Concrete Σ-protocols exist for many other "hard problems" beyond discrete logarithms. These includes the RSA assumption [GQ88], factoring [FF02], and learning parity with noise [JKPT12]. A famous generalization of the Schnorr protocol is due to Okamoto [Oka92].
- The definitions of zero knowledge we have studied are for the so-called "standalone setting" where protocols are run in isolation. In practice protocols are not standalone, but are run as sub-protocols in larger protocols. Security in this setting is often called *concurrent security*, and has been studied in several important works. See, e.g., [GMY06, BPS06].
- Non-interactive zero-knowledge is also possible in the common reference string model, where one assumes a trusted party produces a reference string that is given as input to all parties. In this setting, the famous Groth-Sahai proof system [GS08, GOS12] allows to generate very efficient proofs for concrete "algebraic" statements. Zero knowledge for "non-algebraic" statements is also an interesting question [JKO13].

[8] Trivial languages are languages that can be decided without the need of an interactive proof.

- An important application of non-interactive zero knowledge, is to the setting of public-key encryption (PKE): Seminal work of Naor-Yung [NY90] and Dolev-Dwork-Naor [DDN03] showed that it is possible to use non-interactive zero-knowledge proofs for generically transforming a PKE with weak security (IND-CPA) into a PKE with very strong security (IND-CCA).
- Secure multi-party computation (MPC) allows a set of mutually distrusting parties to compute an *arbitrary* function of their inputs, without revealing more than what the output already reveals. Zero knowledge is one of the main tools for constructing MPC protocols with active security; this was shown for the first time by [GMW87] and has been extended in several directions in subsequent work.
- Zero knowledge finds also many applications in the construction and analysis of concrete cryptographic protocols, including electronic voting [KZZ15], electronic cash [CPST15], and anonymous credentials [CL01].

References

[BDG+13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why “Fiat-Shamir for proofs” lacks a proof. In *TCC*, pages 182–201, 2013.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, pages 390–420, 1992.

[BLV06] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006.

[BPS06] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *FOCS*, pages 345–354, 2006.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, pages 62–73, 1993.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001.

[CPST15] Sébastien Canard, David Pointcheval, Olivier Sanders, and Jacques Traoré. Divisible e-cash made practical. In *PKC*, pages 77–100, 2015.

[Cra96] Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, CWI and University of Amsterdam, 1996.

[Dam10] Ivan Damgård. On Σ-protocols, 2010. Available at `http://www.cs.au.dk/~ivan/Sigma.pdf`.

[DDN03] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM Review*, 45(4):727–784, 2003.

[FF02] Marc Fischlin and Roger Fischlin. The representation problem based on factoring. In *CT-RSA*, pages 96–113, 2002.

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

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *FOCS*, pages 102–113, 2003.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, pages 291–304, 1985.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *FOCS*, pages 174–187, 1986.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *J. Cryptology*, 19(2):169–209, 2006.

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for non-interactive zero-knowledge. *J. ACM*, 59(3):11, 2012.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *EUROCRYPT*, pages 123–128, 1988.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols*. Springer Verlang Berlin Heidelberg, 2010.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *CCS*, pages 955–966, 2013.

[JKPT12] Abhishek Jain, Stephan Krenn, Krzysztof Pietrzak, and Aris Tentes. Commitments and efficient zero-knowledge proofs from learning parity with noise. In *ASIACRYPT*, pages 663–680, 2012.

[KZZ15] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *EUROCRYPT*, pages 468–498, 2015.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

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

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989.

[Ven12] Daniele Venturi. *Crittografia nel Paese delle Meraviglie*. Springer Verlang Italy, 2012.