An Efficient Transform from Sigma Protocols to NIZK with a CRS and Non-Programmable Random Oracle[*]

Yehuda Lindell

Dept. of Computer Science
Bar-Ilan University, Israel
lindell@biu.ac.il

January 7, 2015

Abstract

In this short paper, we present a Fiat-Shamir type transform that takes any Sigma protocol for a relation R and outputs a non-interactive zero-knowledge proof (not of knowledge) for the associated language L_R, in the common reference string model. As in the Fiat-Shamir transform, we use a hash function H. However, zero-knowledge is achieved under standard assumptions in the common reference string model (without any random oracle), and soundness is achieved in the *non-programmable* random oracle model. The concrete computational complexity of the transform is only slightly higher than the original Fiat-Shamir transform.

Keywords: zero knowledge, Sigma protocols, Fiat-Shamir, random oracles, programmability, common reference string

[*] This work was funded by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC consolidators grant agreement n. 615172 (HIPS), and under the European Union's Seventh Framework Program (FP7/2007-2013) grant agreement n. 609611 (PRACTICE).

1 Introduction

Concretely efficient zero knowledge. Zero knowledge proofs [21, 17] play an important role in many fields of cryptography. In secure multiparty computation, zero-knowledge proofs are used to force parties to behave semi-honestly, and as such are a crucial tool in achieving security in the presence of malicious adversaries [18]. This use of zero-knowledge proofs is not only for proving feasibility as in [18], and efficient zero-knowledge proofs are used widely in protocols for achieving concretely efficient multiparty computation with security in the presence of malicious adversaries; see [30, 23, 27, 31, 24, 25] for just a few examples. Efficient zero knowledge is also widely used in protocols for specific problems like voting, auctions, anonymous credentials, and more.

Most efficient zero knowledge proofs known are based on Sigma protocols [8] and [22, Ch. 7]. Informally, Sigma protocols are honest-verifier perfect zero-knowledge interactive proof systems with some very interesting properties. First, they are three round public-key protocols (meaning that the verifier's single message—or challenge—is just a random string); second, if the statement is not in the language, then for every first prover message there exists just a single verifier challenge that can be answered; third, there exists a simulator that is given the statement and verifier challenge and generates the exact distribution over the prover's messages with this challenge. Although seemingly specific, there exist Sigma protocols for a wide variety of tasks like proving that a tuple is of the Diffie-Hellman type, that an ElGamal commitment is to a certain value, that a Paillier encryption is to zero, and many more. It is also possible to efficiently combine Sigma protocols to prove compound statements [5]; e.g., $(x_1 \in L \wedge y_1 \in L) \vee (x_2 \in L \wedge y_2 \in L)$. Finally, it is possible to efficiently compile a Sigma protocol to a zero-knowledge proof (resp., zero-knowledge proof of knowledge) with the addition of one additional round (resp., two additional rounds) [22, Ch. 7].

The Fiat-Shamir transform and NIZK. The Fiat-Shamir transform [14] is a way of transforming any *public-coin zero-knowledge proof* into a *non-interactive zero-knowledge proof* [2, 12].[1] The transform is very simple, and works by having the prover compute the verifier's (random) messages by applying an "appropriate" hash function to the previous prover messages. The security of this transform was proven in the random oracle model [29]. This means that if the hash function is modeled as an external random function, then the result of applying the Fiat-Shamir transform to a public-coin zero-knowledge proof is a non-interactive zero-knowledge proof. However, it was also shown that it is not possible to prove this statement for any concrete instantiation of the hash function. Rather, there exist public-coin zero-knowledge proofs for which every concrete instantiation of the hash function in the Fiat-Shamir transform yields an insecure scheme [20].

When applying the Fiat-Shamir transform to a Sigma protocol, the result is an extraordinarily efficient non-interactive zero-knowledge proof. We remark that this is not immediate since Sigma protocols are only *honest-verifier* zero knowledge. Thus, the Fiat-Shamir transform both removes interaction and guarantees zero knowledge for malicious verifiers.

The Fiat-Shamir transform is very beneficial in obtaining efficient protocols since it saves expensive rounds of communication without increasing the computational complexity of the original protocol. In addition, it is very useful in settings where the non-interactive nature of the proof is essential (e.g., in anonymous credentials). However, as we have seen, this reliance on the Fiat-Shamir

[1] The Fiat-Shamir transform was designed to construct signature schemes from public-coin zero-knowledge proofs, and later works also studied its security as a signature scheme. However, the results are actually the same for non-interactive zero knowledge.

is only sound in the random-oracle model. This leads us to the following question:

> *Can we construct a Fiat-Shamir type transformation, that is highly efficient and is secure in the standard model (without a random oracle)?*

In this paper, we take a first step towards answering this question.

The random-oracle saga. Reliance on the random oracle model is controversial, with strong advocates on one side and strong opponents on the other. However, it seems that almost all agree that proving security without reliance on the random oracle model is preferable. As such, there has been a long line of work attempting to prove security of existing schemes without reliance on a random oracle, and to construct new schemes that are comparable to existing ones (e.g., with respect to efficiency) but don't require a random oracle. In the case of the Fiat-Shamir transform, there is no chance of proving it secure in general without a random oracle, due to the impossibility result of [20]. Thus, the aim is to construct a transform that is comparable to Fiat-Shamir in terms of efficiency, but can be proven secure in the standard model.

An interesting development regarding the random oracle is that not all random oracles are equal. In particular, Nielsen introduced the notion of a *non-programmable* random oracle [26], based on the observation that many proofs of security—including that of the Fiat-Shamir transform—rely inherently on the ability of the simulator (or security reduction) to "program" the random oracle and fix specific input/output pairs. In contrast, a non-programmable random oracle is simply a random function that all parties have access to. In some sense, reliance on a non-programmable random oracle seems conceptually preferable since it more closely models the intuition that "appropriate hash functions" behave in a random way. Formally, of course, this makes no sense. However, proofs of security that do not require programmability are preferable in the sense that they rely on less properties of the random oracle and can be viewed as a first step towards removing it entirely.

Our results. In this paper, we present a Fiat-Shamir type transform from Sigma protocols to non-interactive zero knowledge proofs (that are *not* proofs of knowledge). The transform is extremely efficient; for example, under the Decisional Diffie-Hellman assumption, the cost of transforming a Sigma protocol to a non-interactive zero-knowledge proof is just 4 exponentiations, and the transmission of a single number in $\mathbb{Z}_q$ (where q is the order of the group). Our transform achieves two advantages over the Fiat-Shamir transform:

1. The zero-knowledge property holds in the *standard model* and does not require any random oracle at all. This is in contrast to the standard Fiat-Shamir transform when applied to Sigma protocols, for which the only known proof uses a (fully programmable) random oracle. Our transform utilizes the common reference string model, which is inherent since one-round zero-knowledge protocols do not exist for languages not in $\mathcal{BPP}$ [19].

2. The soundness property holds when the hash function is modeled as a *non-programmable* random oracle.

The fact that zero knowledge holds without any random oracle implies that the difficulties regarding zero knowledge composition that arise in the random oracle model [32] are not an issue here. It also implies that the random oracle is not needed for any simulation, and one only needs it to prove soundness.

Our transform. The technique used in our transform is very simple. We use a two-round equivocal commitment scheme for which there exists a trapdoor with which commitments can be decommitted to any value. One example of such a scheme is that of Pedersen's commitment [28]. Specifically, let g and h be two random generators of a group in which the discrete log problem is assumed to be hard. Then, $c = \mathsf{Com}_{g,h}(x) = g^r \cdot h^x$, where $r \leftarrow \mathbb{Z}_q$ is random. This scheme is perfectly hiding, and it can be shown to be computationally binding under the discrete log assumption. However, if the discrete log of h with respect to g is known, then it is possible to decommit to any value (if $h = g^\alpha$ and α is known to the committer, then it can define $c = g^y$ and then for any x it simply sets $r = y - \alpha \cdot x$).

We define a common reference string (CRS) that contains the first message of the commitment scheme. Thus, when the simulator chooses the CRS, it will know the trapdoor, thereby enabling it to equivocate. Let Σ be a Sigma protocol for some language, and denote the messages of the proof that $x \in L$ by (a, e, z). Then, in the Fiat-Shamir transform, the prover uses the verifier challenge $e = H(x, a)$. In our transform, the prover first computes a commitment to a using randomness r, denoted $c = \mathsf{Com}(a; r)$, and sets $e = H(x, c)$. Then, the proof contains (a, r, z), and the verifier computes $c = \mathsf{Com}(a; r)$, $e = H(x, c)$ and verifies that (a, e, z) is an accepting proof that $x \in L$. Intuitively, since c is a commitment to a, soundness is preserved like in the original Fiat-Shamir transform. However, since the simulator can choose the common reference string, and so can know the trapdoor, it can equivocate to any value it likes. Thus, the simulator can generate a commitment c that can be opened later to anything. Next, it computes $e = H(x, c)$. Finally, it runs the Sigma protocol simulator with the verifier challenge e already known, in order to obtain an accepting proof (a, e, z). Finally, it finds r such that $c = \mathsf{Com}(a; r)$ to "explain" c as a commitment to a. This reverse order of operations is possible since the simulator can equivocate; soundness is preserved since the real prover cannot.

As appealing as the above is, the proof of soundness is problematic since the commitment is only computational binding and the reduction would need to construct an adversary breaking the binding from any adversary breaking the soundness. However, since a cheating prover outputs a single proof, such a reduction seems problematic, even in the random oracle model.[2] We therefore use a *dual-mode commitment* (or hybrid trapdoor commitment [4]) which means that there are two ways to choose the common reference: in one way the commitment is perfectly binding, and in the other it is equivocal. This enables us to prove soundness when the commitment is perfectly binding, and zero knowledge when it is equivocal. We construct dual-mode commitments from any "hard" language with an associated Sigma protocol (see Section 3.2). Thus, the security of our transform for such languages requires no additional assumptions. We also demonstrate a concrete instantiation of our construction that is secure under the DDH assumption, and requires only 4 exponentiations to generate a commitment. Our DDH instantiation of this primitive appeared in [4] (for different applications); we present the construction here in any case for completeness.

Open questions. The major question left open by our work is whether or not it is possible to prove the security of our transform or a similar one using a (concretely efficient) hash function whose security is based on a *standard* cryptographic assumption. Note that even achieving a falsifiable assumption is difficult, and this has been studied by [1] and [11]. However, we have the additional power of a CRS, and this may make it easier.

[2] It may be possible to prove by relying on the extractability of the random oracle, meaning that it is possible to "catch" the cheating prover's queries to the random oracle. We do not know how to do this in this context. In addition, our solution is preferable since we do not even require extractability of random oracle queries.

Related work. Damgård [7] used a very similar transform to obtain 3-round concurrent zero knowledge in the CRS model. Specifically, [7] uses a trapdoor commitment applied to the first prover message, as we do. This enables simulation without rewinding and thus achieves concurrent zero knowledge. However, as we have described, it seems that in our setting a regular tradpoor commitment does not suffice since there is *no interaction* and thus no possibility of rewinding the adversary (note that in the context of concurrent zero knowledge it is problematic to rewind a cheating verifier when proving zero knowledge, but there is no problem rewinding a cheating prover in order to prove soundness, and this is indeed what [7] do).

The problem of constructing zero knowledge in the non-programmable random oracle model was considered by [9, 10]. However, their construction are not completely non-interactive and require two messages. This is due to the fact that their aim is to solve the problem of deniability and transferability of NIZK proofs, and so some interaction is necessary. We also remark that the transform from Σ-protocols to Ω-protocols used in their construction requires repeating the proof multiple times, and so is far less efficient.

2 Definitions

2.1 Preliminaries

Let R be a relation; we denote the associated language by L_R. That is, $L_R = \{x \mid \exists w : (x,w) \in R\}$. We denote the security parameter by n. We model a random oracle simply as a random length-preserving function $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$. In our work, we use the random oracle only to prove soundness, and there is therefore no issue of "programmability". When S is a set, $x \leftarrow S$ denotes choosing x from S with a uniform distribution.

2.2 Sigma Protocols and NIZK

For the sake of completeness, we define Sigma protocol and adaptive non-interactive zero knowledge (NIZK). Our formulation of non-interactive zero knowledge is both adaptive (meaning that statements can be chosen as a function of the common reference string) and considers the case where many proofs are given.

Sigma protocols. We briefly define Sigma protocols. For more details, see [8] and [22, Ch. 7]. Let R be a binary polynomial-bounded relation. A Σ protocol $\pi = (P_1, P_2, V_\Sigma)$ is a 3-round public-coin protocol: the prover's first message is denoted $a = P_1(x)$; the verifier's message is a random string $e \in_R \{0,1\}^n$, and the prover's second message is denoted $z = P_2(x,a,e)$. We write $V_\Sigma(x,a,e,z) = 1$ if and only if the verifier accepts, and in this case we say tha transcript (a,e,z) is accepting for x. We now formally define the notion of a Sigma-protocol:

Definition 2.1 *A protocol $\pi = (P_1, P_2, V_\Sigma)$ is a* Sigma-protocol *for relation R if it is a three-round public-coin protocol, and the following requirements hold:*

- **Completeness:** *If P and V follow the protocol on input x and private input w to P where $(x,w) \in R$, then V always accepts.*
- **Special soundness:** *There exists a polynomial-time algorithm A that given any x and any pair of accepting transcripts $(a,e,z), (a,e',z')$ for x, where $e \neq e'$, outputs w such that $(x,w) \in R$.*

- **Special honest verifier zero knowledge:** *There exists a probabilistic polynomial-time simulator $\mathcal{S}_\Sigma$ such that*

  $$\left\{\mathcal{S}_\Sigma(x,e)\right\}_{x\in L; e\in\{0,1\}^n} \equiv \left\{\langle P(x,w), V(x,e)\rangle\right\}_{x\in L; e\in\{0,1\}^n}$$

  where $\mathcal{S}_\Sigma(x,e)$ denotes the output of simulator M upon input x and e, and $\langle P(x,w), V(x,e)\rangle$ denotes the output transcript of an execution between P and V, where P has input (x,w), V has input x, and V's random tape (determining its query) equals e.

Adaptive non-interactive zero-knowledge. In the model of non-interactive zero-knowledge proofs [2], the prover and verifier both have access to a public common reference string (CRS). We present the definition of adaptive zero knowledge, meaning that both the soundness and zero-knowledge hold when statements can be chosen as a function of the CRS. We also consider the unbounded version, meaning that zero knowledge holds for any polynomial number of statements proven. We present the definition directly, and refer to [15, Section 4.10] for motivation and discussion. We define soundness in the non-programmable random oracle model, since this is what we use in our construction.

Definition 2.2 (adaptive non-interactive unbounded zero-knowledge): *A triple of probabilistic polynomial-time machines* (GenCRS, P, V) *is called an* adaptive non-interactive unbounded zero-knowledge argument system *for a language $L \in \mathcal{NP}$ with an NP-relation R_L, if the following holds:*

- Perfect completeness: *For every $(x,w) \in R_L$, $\Pr[V(x,\rho_n,P(x,w,\rho_n)) = 1] = 1$ where ρ_n is randomly sampled according to $\mathsf{GenCRS}(1^n)$.*

- Adaptive computational soundness with a non-programmable random oracle: *For every probabilistic polynomial-time function $f : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^n \setminus L$ and every probabilistic polynomial-time (cheating) prover $\mathcal{B}$,*

  $$\Pr\left[V^{\mathcal{O}}(f(\rho_n), \rho_n, \mathcal{B}^{\mathcal{O}}(\rho_n)) = 1\right] < \mu(n)$$

  where ρ_n is randomly sampled according to $\mathsf{GenCRS}(1^n)$ and $\mathcal{O} : \{0,1\}^* \to \{0,1\}^*$ is a random length-preserving function.

- Adaptive unbounded zero knowledge: *There exists a probabilistic polynomial-time simulator $\mathcal{S}_{\mathsf{zk}}$ such that for every probabilistic polynomial-time function*

  $$f : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^n \times \{0,1\}^{\mathrm{poly}(n)} \cap R_L,$$

  every polynomial $p(\cdot)$ and every probabilistic polynomial-time distinguisher D, there exists a negligible function μ such that for every n,

  $$\left|\Pr\left[D\left(\mathcal{R}^f(\rho_n, P^f(1^{n+p(n)}))\right) = 1\right] - \Pr\left[D\left(\mathcal{S}_{\mathsf{zk}}{}^f(1^{n+p(n)}))\right) = 1\right]\right| \le \mu(n)$$

  where ρ_n is randomly sampled according to $\mathsf{GenCRS}(1^n)$, f_1 and f_2 denote the first and second outputs of f respectively, and $\mathcal{R}^f(\rho_n, P^f(1^{n+p(n)}))$ and $\mathcal{S}_{\mathsf{zk}}{}^f(1^{n+p(n)})$ denote the output from the following experiments:

Real proofs $\mathcal{R}^f(\rho_n, P^f(1^{n+p(n)}))$:

1. $\rho \leftarrow \mathsf{GenCRS}(1^n)$*: a common reference string is sampled*
2. *For $i = 1, \ldots, p(n)$ (initially $\vec{x}$ and $\vec{\pi}$ are empty):*
   1. (a) *$x_i \leftarrow f_1(\rho_n, \vec{x}, \vec{\pi})$: the next statement x_i to be proven is chosen.*
   2. (b) *$\pi_i \leftarrow P(f_1(\rho_n, \vec{x}, \vec{\pi}), f_2(\rho_n, \vec{x}, \vec{\pi}), \rho_n)$: the ith proof is generated.*
   3. (c) *Set $\vec{x} = x_1, \ldots, x_i$ and $\vec{\pi} = \pi_1, \ldots, \pi_i$*
3. *Output $(\rho_n, \vec{x}, \vec{\pi})$.*

Simulation ${\mathcal{S}_{\mathsf{zk}}}^f(1^{n+p(n)})$:

1. $\rho \leftarrow \mathcal{S}_{\mathsf{zk}}(1^n)$*: Simulator $\mathcal{S}_{\mathsf{zk}}$ (upon input 1^n) outputs a reference string ρ*
2. *For $i = 1, \ldots, p(n)$ (initially $\vec{x}$ and $\vec{\pi}$ are empty):*
   1. (a) *$x_i \leftarrow f_1(\rho_n, \vec{x}, \vec{\pi})$: the next statement x_i to be proven is chosen.*
   2. (b) *$\pi_i \leftarrow \mathcal{S}_{\mathsf{zk}}(x_i)$: Simulator $\mathcal{S}_{\mathsf{zk}}$ generates a simulated proof π_i that $x_i \in L$.*
   3. (c) *Set $\vec{x} = x_1, \ldots, x_i$ and $\vec{\pi} = \pi_1, \ldots, \pi_i$*
3. *Output $(\rho, \vec{x}, \vec{\pi})$.*

Adaptive NIZK proof systems can be constructed from any (doubly) enhanced trapdoor permutation [12]; see [15, Appendix C.4.1] and [16] regarding the assumption.

3 Dual-Mode Commitments

We use a commitment scheme in the CRS model with the property that it is *perfectly binding* given the correctly constructed CRS, but is *equivocal* to a simulator who generates the CRS in an alternative but indistinguishable way. Stated differently, the simulator can generate the CRS so that it looks like a real one, but a commitment can be decommitted to any value. We show how to construct this from any "hard" NP-relation with a Sigma protocol (to be defined below). This construction has the advantage that we obtain non-interactive zero knowledge for such relations under no additional assumptions. This construction is based on the commitment scheme from Sigma protocols that appeared in [6]. However, [6] constructed a standard commitment scheme, and we show how the same ideas can be used to achieve a dual commitment scheme. Following this, we show a concrete instantiation under the DDH assumption which is extremely efficient.

Such a commitment was called a hybrid trapdoor commitment in [4], who studied this primitive in depth and presented a number of constructions. In particular, the DDH-based construction in [4] is identical to ours. We repeat it here for the sake of completeness.

3.1 Definition

Before we show how to construct such commitments, we provide a formal definition.

Definition 3.1 *A* dual-mode commitment scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{GenCRS}, \mathsf{Com}, \mathcal{S}_{\mathsf{com}})$ *such that*

- $\mathsf{GenCRS}(1^n)$ *outputs a common reference string, denoted ρ,*

- $(\mathsf{GenCRS}, \mathsf{Com})$*: When* $\rho \leftarrow \mathsf{GenCRS}(1^n)$ *and* $m \in \{0,1\}^n$*, the algorithm* $\mathsf{Com}_\rho(m;r)$ *with a random* r *is a non-interactive perfectly-binding commitment scheme,*

- $(\mathsf{Com}, \mathcal{S}_{\mathsf{com}})$*: For every probabilistic polynomial-time adversary* $\mathcal{A}$ *and every polynomial* $p(\cdot)$*, the output of the following two experiments is computationally indistinguishable:*

$\text{REAL}_{\mathsf{Com},\mathcal{A}}(1^n)$	$\text{SIMULATION}_{\mathcal{S}_{\mathsf{com}}}(1^n)$
1. $\rho \leftarrow \mathsf{GenCRS}(1^n)$	*1.* $\rho \leftarrow \mathcal{S}_{\mathsf{com}}(1^n)$
2. For $i = 1, \ldots, p(n)$*:*	*2. For* $i = 1, \ldots, p(n)$*:*
(a) $m_i \leftarrow \mathcal{A}(\rho, \vec{c}, \vec{r})$	*(a)* $c_i \leftarrow \mathcal{S}_{\mathsf{com}}$
(b) $r_i \leftarrow \{0,1\}^{\text{poly}(n)}$	*(b)* $m_i \leftarrow \mathcal{A}(\rho, \vec{c}, \vec{r})$
(c) $c_i = \mathsf{Com}_\rho(m_i; r_i)$	*(c)* $r_i \leftarrow \mathcal{S}_{\mathsf{com}}(m_i)$
(d) Set $\vec{c} = c_1, \ldots, c_i$ *and* $\vec{r} = r_1, \ldots, r_i$	*(d) Set* $\vec{c} = c_1, \ldots, c_i$ *and* $\vec{r} = r_1, \ldots, r_i$
3. Output $\mathcal{A}(\rho, m_1, r_1, \ldots, m_{p(n)}, r_{p(n)})$	*Output* $\mathcal{A}(\rho, m_1, r_1, \ldots, m_{p(n)}, r_{p(n)})$

3.2 Membership-Hard Languages with Efficient Sampling

Intuitively, a *membership-hard language* L is one for which it is possible to sample instances of the problem in a way that it is hard to detect if a given instance is in the language or not. In more detail, there exists a sampling algorithm S_L that receives for input a bit b and outputs an instance in the language together with a witness w if $b = 0$, and an instance not in the language if $b = 1$. The property required is that no polynomial-time distinguisher can know which bit S_L received. We let S_L^x denote the instance part of the output (without the witness, in the case that $b = 0$). We now define this formally.

Definition 3.2 *Let* L *be a language. We say that* L *is* membership-hard with efficient sampling *if there exists a probabilistic polynomial-time sampler* S_L *such for every probabilistic polynomial-time distinguisher* D *there exists a negligible function* $\mu(\cdot)$ *such that*

$$\left| \Pr[D(S_L^x(0,1^n),1^n)=1] - \Pr[D(S_L(1,1^n),1^n)=1] \right| \le \mu(n)$$

Such languages can be constructed from essentially any cryptographic assumption. Specifically, if one-way functions exist then there exists a pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$. Now, define L to be the language of all images of G; i.e., $L = \{G(s) \mid s \in \{0,1\}^*\}$, and define $S_L(0,1^n) = (G(U_n), U_n)$, and $S_L(1,1^n) = U_{2n}$, where U_k is a uniformly distributed string of length k. It is clear that this language is membership-hard with efficient sampling.

Nevertheless, we will be more interested in such languages that have efficient Sigma protocols associated with them. One simple such examples is the language of Diffie-Hellman tuples (where $S_L(0,1^n)$ outputs a random Diffie-Hellman tuple (g,h,g^a,h^a) together with a, and $S_L(1,1^n)$ outputs a random non Diffie-Hellman tuple (g,h,g^a,h^b), where a and b are random).

We remark that Feige and Shamir [13] consider the notion of an invulnerable generator for a language. Their notion considers a *relation* for which it is possible to generate an instance such that it is hard to find the associated witness. In contrast, our notion relates to languages and not relations, and on deciding membership rather than finding witnesses.

3.3 Dual-Mode Commitments from Membership-Hard Languages with Sigma Protocols

We now construct a dual-mode commitment scheme from any language L that is membership hard, and has an associated Sigma protocol. Recall that the verifier message of a Sigma protocol is always a uniformly distributed $e \in_R \{0,1\}^n$. We denote the first and second prover messages of the Sigma protocol on common input x (and witness w for the prover) by $a = P_1(x,w)$ and $z = P_2(x,w,a,e)$, respectively. We denote by $\mathcal{S}_\Sigma$ the simulator for the Sigma protocol. Thus, $\mathcal{S}_\Sigma(x,e)$ outputs (a,z).

PROTOCOL 3.3 (Dual-Mode Commitment (General Construction))

- **Regular CRS generation (perfect binding):** Run the sampler S_L for the language L with input $(1,1^n)$, and receive back an x (recall that $x \notin L$). The CRS is $\rho = x$.
- **Commitment Com:** To commit to a value $m \in \{0,1\}^n$, set $e = m$, run $\mathcal{S}_\Sigma(x,e)$ and obtain (a,z). The commitment is $c = a$.
- **Decommitment:** To decommit, provide e,z and the receiver checks that $V_\Sigma(a,e,z) = 1$.
- **Simulator $\mathcal{S}_{\mathsf{com}}$:**
  1. Upon input 1^n, simulator $\mathcal{S}_{\mathsf{com}}$ runs the sampler S_L for the language L with input $(0,1^n)$, and receives back (x,w) (recall that $x \in L$ and w is a witness to this fact). Then, $\mathcal{S}_{\mathsf{com}}$ computes $a = P_1(x,w)$, sets $c = a$ and $\rho = x$, and outputs (c,ρ).
  2. Upon input $m \in \{0,1\}^n$, simulator $\mathcal{S}_{\mathsf{com}}$ sets $e = m$ and outputs $z = P_2(x,w,a,e)$.

The fact that the commitment scheme is *perfectly binding* in the regular CRS case holds since $x \notin L$ and thus for every a, there exists a *single* e,z for which (a,e,z) is an accepting proof. In contrast, in the alternative CRS generation case, $x \in L$ and the simulator knows the witness w. Thus, it can generate a "commitment" $a = P_1(x,w)$, and then for any $m \in \{0,1\}^t$ chosen later, it can decommit to m by setting $e = m$, computing $z = P_2(x,e)$ and supplying (e,z). Since (a,e,z) is a valid proof, and the Sigma protocol simulator is perfect, the only difference between this and a real commitment is the fact that $x \in L$. However, by the property of the sampler S_L, this is indistinguishable from the case that $x \notin L$.

Theorem 3.4 *Let L be a membership-hard language, and let (P_1, P_2, V_Σ) be a Sigma protocol for L. Then, Protocol 3.3 is a dual-mode commitment scheme.*

Proof: The fact that $\mathsf{Com}_\rho(m;r)$ is perfectly binding when $\rho \leftarrow \mathsf{GenCRS}(1^n)$ follows from the fact that when $x \notin L$ it holds that for every a there exists a *single* e such that $V_\Sigma(x,a,e,z) = 1$.

We now show that the outputs of $\textsc{real}_{\mathsf{Com},\mathcal{A}}(1^n)$ and $\textsc{simulation}_{\mathcal{S}_{\mathsf{com}}}(1^n)$ (as in Definition 3.1) are computationally indistinguishable. (We prove this first since we will use it later to prove the computational hiding of $(\mathsf{GenCRS},\mathsf{Com})$.) We begin by modifying the $\textsc{real}_{\mathsf{Com},\mathcal{A}}(1^n)$ experiment to $\textsc{hybrid}_{\mathsf{Com},\mathcal{A}}(1^n)$, where the only difference is that the CRS is generated by running $S_L(0,1^n)$ in the way that $\mathcal{S}_{\mathsf{com}}$ generates it, instead of running $S_L(1,1^n)$. Apart from this, everything remains exactly the same. (Observe that since Com runs the Sigma protocol simulator, it makes no difference if $x \in L$ or $x \notin L$.) By the assumption that $S_L^x(0,1^n)$ is computationally indistinguishable from $S_L(1,1^n)$, it follows that the outputs of $\textsc{real}_{\mathsf{Com},\mathcal{A}}(1^n)$ and $\textsc{hybrid}_{\mathsf{Com},\mathcal{A}}(1^n)$ are computationally indistinguishable. Next, we show that $\textsc{hybrid}_{\mathsf{Com},\mathcal{A}}(1^n)$ and $\textsc{simulation}_{\mathcal{S}_{\mathsf{com}}}(1^n)$ are identically distributed. There are two differences between them. First, in SIMULATION the real Sigma-protocol prover is used instead of the simulator; second, in SIMULATION the value c_i is generated before m_i is given, in every iteration. Regarding the first difference, the distributions are identical by the perfect zero-knowledge property of $\mathcal{S}_\Sigma$. Regarding the second difference, once the real prover is used, it makes no difference if c_i is given before or after, since the distribution over a_i is identical. We conclude that $\text{HYBRID}_{\mathsf{Com},\mathcal{A}}(1^n)$ and $\text{SIMULATION}_{\mathcal{S}_{\mathsf{com}}}(1^n)$ are identically distributed, and thus $\text{REAL}_{\mathsf{Com},\mathcal{A}}(1^n)$ and $\text{SIMULATION}_{\mathcal{S}_{\mathsf{com}}}(1^n)$ are computationally indistinguishable.

It remains to show that $(\mathsf{GenCRS}, \mathsf{Com})$ is computationally hiding as a commitment scheme. In order to see this, observe that $\text{SIMULATION}_{\mathcal{S}_{\mathsf{com}}}(1^n)$ is *perfectly hiding.* Intuitively, since it is computationally indistinguishable from a real commitment, this proves computational hiding. More formally, for any pair m_0, m_1 of the same length, the output of REAL with m_0 is computationally indistinguishable from the output of SIMULATION with m_0, and the output of REAL with m_1 is computationally indistinguishable from the output of SIMULATION with m_1. (It is straightforward to modify the experiments to have a fixed message, or to have $\mathcal{A}$ output a pair and choose one at random.) Since the commitment in SIMULATION is perfectly hiding, it follows that the output of SIMULATION with m_0 is identical to the output of SIMULATION with m_1. This implies computational indistinguishability of the output of REAL with m_0 from the output of REAL with m_1. ■

3.4 A Concrete Instantiation from DDH

In this section, we present a dual-mode commitment scheme from the DDH assumption. This can be used for any transform, and may be more efficient if the Sigma protocol for the language being used is less efficient. The complexity is 4 exponentiations for a commitment (by the prover), and 4 exponentiations for a decommitment (by the receiver).

Let $\mathcal{G}$ be the "generator algorithm" of a group in which the DDH assumption is assumed to hold. We denote the output of $\mathcal{G}(1^n)$ by $(\mathbb{G}, q, g, h)$ where $\mathbb{G}$ is the description of a group of order $q > 2^n$ with two random generators g, h.

PROTOCOL 3.5 (Dual-Mode Commitment from DDH)

- **Regular CRS generation (perfect binding):** Run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g, h)$. Choose $\rho_1, \rho_2 \in_R \mathbb{Z}_q$ and compute $u = g^{\rho_1}$ and $v = h^{\rho_2}$. The CRS is $(\mathbb{G}, q, g, h, u, v)$.
- **Alternative CRS generation (equivocal):** As above, except choose a single $\rho \in_R \mathbb{Z}_q$ and compute $u = g^\rho$ and $v = h^\rho$.
- **Commitment:** To commit to a value $m \in \{0,1\}^n$, choose a random $z \in_R \mathbb{Z}_q$ and compute $a = g^z/u^m$ and $b = h^z/v^m$. The commitment is $c = (a, b)$.
- **Decommitment:** To decommit to $c = (a, b)$, provide m, z and the receiver checks that $g^z = a \cdot u^m$ and $h^z = b \cdot v^m$.

The fact that the commitment scheme is *perfectly binding* in the regular CRS case holds since (g, h, u, v) is *not* a Diffie-Hellman tuple. Thus, by the property of the DH Sigma Protocol, for *every* (a, b) there exists a *unique* e for which there exists a value z such that $g^z = a \cdot u^e$ and $h^z = b \cdot v^e$. In contrast, in the alternative CRS generation case, (g, h, u, v) *is* a Diffie-Hellman tuple and the simulator knows the witness ρ. Thus, it can generate $a = g^r$ and $b = h^r$ and then for any $m \in \{0,1\}^n$ chosen later, it can decommit to m by computing $z = r + m\rho$ and supplying (m, z). Since $u = g^\rho$

and $v = h^\rho$ it follows that $g^z = g^{r+m\rho} = g^r \cdot (g^\rho)^m = a \cdot u^m$ and $h^z = h^{r+m\rho} = h^r \cdot (h^\rho)^m = b \cdot v^m$, as required.

The proof of the following theorem follows directly from Theorem 3.4 and the fact that the language of Diffie-Hellman tuples is membership hard, under the DDH assumption.

Theorem 3.6 *If the Decisional Diffie-Hellman assumption holds relative to $\mathcal{G}$, then Protocol 3.5 is a dual-mode commitment scheme.*

4 The Non-Interactive Zero-Knowledge Transformation

We denote by P_1, P_2 the prover algorithms for a Sigma protocol for the relation R. Thus, a proof of common statement x with witness w (for $(x, w) \in R$) is run by the prover sending the verifier the first message $a = P_1(x, w)$, the verifier sending a random query $e \leftarrow \{0,1\}^t$, and the prover replying with $z = P_2(x, w, e)$. We denote the verification algorithm by $V_\Sigma(x, a, e, z)$.

PROTOCOL 4.1 (NIZK from Sigma Protocol for Relation R)

- **Inputs:** common statement x; the prover also has a witness w such that $(x, w) \in R$
- **Common reference string:** the (regular) CRS ρ of a dual-mode commitment scheme, and a key s for a hash function family H.
- **Auxiliary input:** 1^n, where $n \in \mathbb{N}$ is the security parameter
- **The prover algorithm $\boldsymbol{P(x, w, \rho)}$:**
  1. Compute $a = P_1(x, w)$
  2. Choose a random value $r \in \{0,1\}^{\text{poly}(n)}$ and compute $c = \mathsf{Com}_\rho(a; r)$, where $\mathsf{Com}_\rho(a; r)$ is the dual-mode commitment to a using randomness r and CRS ρ
  3. Compute $e = H_s(x, c)$
  4. Compute $z = P_2(x, w, a, e)$
  5. Output a proof $\pi = (x, a, r, z)$
- **The verifier algorithm $\boldsymbol{V(x, \rho, a, r, z)}$:**
  1. Compute $c = \mathsf{Com}_\rho(a; r)$
  2. Compute $e = H_s(x, c)$
  3. Output $V_\Sigma(x, a, e, z)$

The intuition behind the transformation has been described in the introduction. We therefore proceed directly to prove its security.

4.1 Zero Knowledge

Lemma 4.2 *Let $\Sigma = (P_1, P_2, V_\Sigma)$ be a Sigma protocol for a relation R and let* Com *be a dual-mode commitment. Then, Protocol 4.1 with Σ is zero-knowledge for the language L_R in the common reference string model.*

Proof: We construct a simulator $\mathcal{S}_{\mathsf{zk}}$ (as in Definition 2.2) for Protocol 4.1 as follows:

- Upon input 1^n, $\mathcal{S}_{\mathsf{zk}}$ runs $\mathcal{S}_{\mathsf{com}}(1^n)$ for the dual-mode commitment scheme and obtains the value ρ. In addition, $\mathcal{S}_{\mathsf{zk}}$ samples a key s for the hash function. $\mathcal{S}_{\mathsf{zk}}$ outputs the CRS (ρ, s).
- Upon input x (for every $x_1, \ldots, x_{p(n)}$), simulator $\mathcal{S}_{\mathsf{zk}}$ runs $\mathcal{S}_{\mathsf{com}}$ to obtain some c. Then, $\mathcal{S}_{\mathsf{zk}}$ computes $e = H_s(x, c)$ and runs the simulator $\mathcal{S}_\Sigma$ for the Sigma protocol upon input (x, e). Let the output of the simulator be (a, z). Then, $\mathcal{S}_{\mathsf{zk}}$ runs $\mathcal{S}_{\mathsf{com}}(a)$ from the dual-mode commitment to obtain r such that $c = \mathsf{Com}_\rho(a; r)$. Finally, $\mathcal{S}_{\mathsf{zk}}$ outputs (x, a, r, z).

Intuitively, the difference between a simulated proof and a real one is in the dual-mode commitment. Note also that $\mathcal{S}_{\mathsf{zk}}$ uses the Sigma protocol simulator. However, by the property of Sigma protocols, these messages have an identical distribution. Thus, we prove the zero-knowledge property by reducing the security to that of the dual-commitment scheme, as in Definition 3.1.

First, we construct an alternative simulator $\mathcal{S}'$ who in every iteration (for $i = 1, \ldots, p(n)$) receives (x, w); i.e., $\mathcal{S}'$ receives both $f_1(\rho, \vec{x}, \vec{\pi})$ and $f_2(\rho, \vec{x}, \vec{\pi})$ and so also receives the witness for the fact that $x \in L$. In the first stage of the simulation, $\mathcal{S}'$ works exactly like $\mathcal{S}_{\mathsf{zk}}$ to generate the CRS (ρ, s). In addition, $\mathcal{S}'$ generates c by running $\mathcal{S}_{\mathsf{com}}$, just like $\mathcal{S}_{\mathsf{zk}}$. However, in order to generate (a, z), $\mathcal{S}'$ uses (x, w) and works as follows. It first computes $e = H_s(x, c)$ exactly like $\mathcal{S}_{\mathsf{zk}}$. However, $\mathcal{S}'$ runs $P_1(x, w)$ to obtain a (instead of running $\mathcal{S}_\Sigma$), and then runs $P_2(x, w, a, e)$ to obtain z. Finally, $\mathcal{S}'$ runs $\mathcal{S}_{\mathsf{com}}(a)$ to obtain r such that $c = \mathsf{Com}_\rho(a; r)$. The only difference between $\mathcal{S}_{\mathsf{zk}}$ and $\mathcal{S}'$ is how the values a, z are obtained. Since for every e, $\mathcal{S}_\Sigma$ outputs (a, z) that are distributed identically as in a real proof with e, it holds that the output distributions of $\mathcal{S}_{\mathsf{zk}}$ and $\mathcal{S}'$ are identical.

We now proceed to show that the output distribution of $\mathcal{S}'$ is computationally indistinguishable to a real proof. Formally, let $f = (f_1, f_2)$ be the function choosing the inputs as in Definition 2.2. We construct an adversary $\mathcal{A}$ for the dual-mode commitments as in Definition 3.1, with input 1^n:

1. $\mathcal{A}$ receives ρ, chooses a key s for the hash function family H, and sets the CRS to be (ρ, s).
2. For $i = 1, \ldots, p(n)$ ($\vec{x}$ and $\vec{\pi}$ are initially empty):
   (a) $\mathcal{A}$ receives $(\rho, \vec{c}, \vec{r})$ and knows $m_1, \ldots, m_{i-1}$ and $x_1, \ldots, x_{i-1}$ (since these were generated by $\mathcal{A}$ in previous iterations).
   (b) For every $j = 1, \ldots, i-1$, $\mathcal{A}$ sets $a_j = m_j$, $e_j = H_s(x_j, c_j)$, $z_j = P_2(x_j, w_j, a_j, e_j)$, and $\pi_j = (x_j, a_j, r_j, z_j)$. Finally $\mathcal{A}$ sets $\vec{x} = (x_1, \ldots, x_{i-1})$ and $\vec{\pi} = (\pi_1, \ldots, \pi_{i-1})$.
   (c) $\mathcal{A}$ computes $(x_i, w_i) = f((\rho, s), \vec{x}, \vec{\pi})$
   (d) $\mathcal{A}$ outputs $m_i = a_i = P_1(x_i, w_i)$, as in Step 2(a) of the REAL experiment in Definition 3.1
3. $\mathcal{A}$ receives $((\rho, s), m_1, r_1, \ldots, m_{p(n)}, r_{p(n)})$ and works as follows:
   (a) For every $i = 1, \ldots, p(n)$, $\mathcal{A}$ sets $a_i = m_i$, computes $e_i = H_s(x_i, \mathsf{Com}(m_i; r_i))$, $z_i = P_2(x_i, a_i, e_i)$, and defines $\pi_i = (x_i, a_i, r_i, z_i)$.
   (b) $\mathcal{A}$ outputs $((\rho, s), x_1, \ldots, x_{p(n)}, \pi_1, \ldots, \pi_{p(n)})$

Now, if $\mathcal{A}$ interacts in the "real commitment" experiment REAL for dual-mode commitments, then its output is exactly the same output as in the real proofs experiment $\mathcal{R}^f$ in Definition 2.2. This is because the CRS is generated according to the dual commitment scheme, and the algorithm run by $\mathcal{A}$ to compute all the (x_i, a_i, r_i, z_i) is exactly the same as the honest prover $P(x_i, w_i, \rho_i)$. The only difference is that $\mathcal{A}$ receives (r_i, c_i) externally. However, $c_i = \mathsf{Com}_\rho(a_i; r_i)$ and r_i is uniformly distributed in this experiment. Thus, it is exactly the same as P in Protocol 4.1.

In contrast, if $\mathcal{A}$ interacts in the "simulation" experiment SIMULATION for dual-mode commitments, then its output is distributed identically to $\mathcal{S}'$. This is because the CRS ρ is computed as $\rho \leftarrow \mathcal{S}_{\mathsf{com}}(1^n)$, as too are $c_i \leftarrow \mathcal{S}_{\mathsf{com}}$ and $r_i \leftarrow \mathcal{S}_{\mathsf{com}}(m_i)$ in the dual-commitment simulation experiment, exactly as computed by $\mathcal{S}'$.

Thus, by Definition 3.1 it follows that the output of $\mathcal{S}'$ is computationally indistinguishable from a real proof. This implies that the output of $\mathcal{S}_{\mathsf{zk}}$ is computationally indistinguishable from a real proof, as required by Definition 2.2. ■

4.2 Interactive Argument (Adaptive Soundness)

We now prove that Protocol 4.1 is a non-interactive argument system. In particular, it is computationally (adaptively) sound.

Lemma 4.3 *Let $\Sigma = (P_1, P_2, V_\Sigma)$ be a Sigma-protocol for a relation R, let* Com *be a perfectly-binding commitment, and let H be a non-programmable random oracle. Then, Protocol 4.1 with Σ is a non-interactive argument system for the language L_R in the common reference string model.*

Proof: Completeness is immediate. We proceed to prove adaptive soundness, as in Definition 2.2. We will use the fact that for any function g, the relation $\mathcal{R} = \{(x, g(x)\}$ is evasive on pairs $(x, \mathcal{O}(x))$, where $\mathcal{O}$ is a (non-programmable) random oracle. This means that, given oracle access to $\mathcal{O}$, it is infeasible to find a string x so that the pair $(x, \mathcal{O}(x)) \in \mathcal{R}$ [3].

Assume $x \notin L$. Then, by the soundness of the Sigma protocol, we have that for every a there exists a *single* $e \in \{0,1\}^n$ for which (a, e, z) is accepting, for some z. Define the function $g(x, c) = e$, where there exist a, r, z such that $c = \mathsf{Com}(a; r)$ and $V_\Sigma(x, a, e, z) = 1$. We stress that since $x \notin L$ and since c is perfectly binding, there exists a *single* value e that fulfills this property. Thus, it follows that g is a *function*, as required.

Since g is a function, it follows that the relation $\mathcal{R} = \{((x, c), g(x, c))\}$ is evasive, meaning that no polynomial-time machine $\mathcal{A}$ can find a pair (x, c) so that $\mathcal{O}(x, c) = g(x, c)$, with non-negligible probability. Assume now, by contradiction, that there exists a probabilistic polynomial-time function f and a probabilistic polynomial-time cheating prover $\mathcal{B}$ such that $V(f(\rho_n), \rho_n, \mathcal{B}(\rho_n)) = 1$ with non-negligible probability (where $\rho_n \leftarrow \mathsf{GenCRS}(1^n)$).

We construct a probabilistic polynomial-time adversary $\mathcal{A}$ as follows. $\mathcal{A}$ runs the regular generation of the dual-mode commitment scheme to obtain ρ_n. Then, $\mathcal{A}$ runs $\mathcal{B}(\rho_n)$ and obtains a tuple (x, a, e, z). If $V(f(\rho_n), \rho_n, (a, r, z)) = 1$, then $\mathcal{A}$ outputs $(x, \mathsf{Com}(a; r))$ and halts. According to the contradicting assumption, $V(f(\rho_n), \rho_n, (a, r, z)) = 1$ with non-negligible probability. This implies that with non-negligible probability, it holds that $V_\Sigma(x, a, \mathcal{O}(x, \mathsf{Com}(a; r)), z) = 1$. However, there is just a single value e for which $V_\Sigma(x, a, \mathcal{O}(x, \mathsf{Com}(a; r)), z) = 1$. Thus, this implies that $\mathcal{O}(x, \mathsf{Com}(a; r)) = e$, with non-negligible probability. Stated differently, this implies that $\mathcal{O}(x, \mathsf{Com}(a; r)) = g(x, \mathsf{Com}(a; r))$ with non-negligible probability, in contradiction to the fact that any function g is evasive for a (non-programmable) random oracle. ■

4.3 Summary

Combining Lemmas 4.2 and 4.3 with the fact that the dual-mode commitment scheme is perfectly binding when the CRS is chosen correctly, we have:

Corollary 4.4 *Let L be a language with an associated Sigma protocol. If dual-mode commitments exist, then there exists a non-interactive zero-knowledge argument system for L in the non-programmable random-oracle model. Furthermore, zero-knowledge holds in the standard model.*

In Theorem 3.4 we showed that dual-mode commitment schemes exist for every membership-hard language with a Sigma protocol. Combining this with the above corollary, we have:

Corollary 4.5 *Let L be a membership-hard language with an associated Sigma protocol. Then, there exists a non-interactive zero-knowledge interactive proof system for L, in the non-programmable random oracle model. Furthermore, zero-knowledge holds in the standard model.*

Acknowledgements

We thank Ben Riva, Nigel Smart and Daniel Wichs for helpful discussions.

References

[1] B. Barak, Y. Lindell and S. Vadhan. Lower Bounds for Non-Black-Box Zero Knowledge. In the *Journal of Computer and System Sciences*, 72(2):321–391, 2006. (An extended abstract appeared in *FOCS 2003*.)

[2] M. Blum, P. Feldman and S. Micali. Non-interactive Zero-Knowledge and its Applications. In *20th STOC*, pages 103–112, 1988.

[3] R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, Revisited. In the 30*th STOC*, pages 209-218, 1998.

[4] D. Catalano and I. Visconti. Hybrid Commitments and their Applications to Zero-Knowledge Proof Systems. *Theoretical Computer Science*, 374(1-3):229-260, 2007.

[5] R. Cramer, I. Damgård and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *CRYPTO'94*, Springer-Verlag (LNCS 839), pages 174–187, 1994.

[6] I. Damgård. On the Existence of Bit Commitments Schemes and Zero-Knowledge Proofs. In *CRYPTO'89*, Springer-Verlag (LNCS 435), pages 17–27, 1989.

[7] I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *EUROCRYPT 2000*, Springer (LNCS 1807), pages 418–430, 2000.

[8] I. Damgård. On Σ Protocols. `http://www.daimi.au.dk/∼ivan/Sigma.pdf`.

[9] Y. Dodis, V. Shoup and S. Walfish. Efficient Constructions of Composable Commitments and Zero-Knowledge Proofs. In *CRYPTO 2008*, Springer (LNCS 5157), pages 515–535, 2008.

[10] Y. Dodis. NIZK in the Non-Programmable Random Oracle Model. *Lecture Notes – Advanced Cryptography*, 2009. `http://www.cs.nyu.edu/courses/fall09/G22.3220-001/scribe/lecture8.pdf`

[11] Y. Dodis, T. Ristenpart and S.P. Vadhan. Randomness Condensers for Efficiently Samplable, Seed-Dependent Sources. In the 9*th TCC*, Springer (LNCS 7194), pages 618–635, 2012.

[12] U. Feige, D. Lapidot and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[13] U. Feige and A. Shamir. Witness Indistinguishable and Witness Hiding Protocols. In the 22*nd STOC*, pages 416–426, 1990.

[14] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO 1986*, Springer-Verlag (LNCS 263) pages 186–194, 1986.

[15] O. Goldreich. *Foundation of Cryptography, Volume II.* Cambridge University Press, 2004.

[16] O. Goldreich. Basing Non-Interactive Zero-Knowledge on (Enhanced) Trapdoor Permutation: The State of the Art. Technical Report, 2009.
`http://www.wisdom.weizmann.ac.il/∼oded/PSBookFrag/nizk-tdp.ps`

[17] O. Goldreich, S. Micali and A. Wigderson. How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. In *CRYPTO'86*, Springer-Verlag (LNCS 263), pages 171–185, 1986.

[18] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In 19*th STOC,* pages 218–229, 1987.

[19] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology,* 7(1):1–32, 1994.

[20] S. Goldwasser and Y. Kalai. On the (In)security of the Fiat-Shamir Paradigm. In the 44*th FOCS*, pages 102–113, 2003.

[21] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing,* 18(1):186–208, 1989.

[22] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols – Techniques and Constructions.* Springer, October 2010.

[23] S. Jarecki and V. Shmatikov. Efficient Two-Party Secure Computation on Committed Inputs. In *EUROCRYPT 2007*, Springer (LNCS 4515), pages 97–114, 2007.

[24] Y. Lindell and B. Pinkas. Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer. In *Journal of Cryptology*, 25(4):680722, 2012. (Extended abstract appeared in *TCC 2011*, Springer (LNCS 6597), pages 329–346, 2011.)

[25] Yehuda Lindell. Fast Cut-and-Choose Based Protocols for Malicious and Covert Adversaries. In *CRYPTO 2013*, Springer (LNCS 8043) pages 1–17, 2013.

[26] J.B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In *CRYPTO 2002*, Springer (LNCS 2442), pages 111–126, 2002.

[27] J.B. Nielsen and C. Orlandi. LEGO for Two-Party Secure Computation. In *TCC 2009*, Springer (LNCS 5444), pages 368–386, 2009.

[28] T.P. Pedersen. Non-interactive and Information-Theoretical Secure Verifiable Secret Sharing. In *CRYPTO'91*, Springer-Verlag (LNCS 576) pages 129–140, 1991.

[29] D. Pointcheval and J. Stern: Security Proofs for Signature Schemes. In *EUROCRYPT 1996*, Springe-Verlag (LNCS 1070), pages 387–398, 1996.

[30] B. Schoenmakers and P. Tuyls. Practical Two-Party Computation Based on the Conditional Gate. In *ASIACRYPT 2004*, Springer (LNCS 3329), pages 119–136, 2004.

[31] A. Shelat, C.H. Shen. Two-Output Secure Computation with Malicious Adversaries. In *EUROCRYPT 2011*, Springer (LNCS 6632), pages 386–405, 2011.

[32] Hoeteck Wee. Zero Knowledge in the Random Oracle Model, Revisited. In *ASIACRYPT 2009*, Springer (LNCS 5912), pages 417–434, 2009.