Communication-Efficient Non-Interactive Proofs of Knowledge with Online Extractors

Marc Fischlin*

Institute for Theoretical Computer Science,
ETH Zürich, Switzerland

marc.fischlin@inf.ethz.ch
http://www.fischlin.de/

Abstract. We show how to turn three-move proofs of knowledge into non-interactive ones in the random oracle model. Unlike the classical Fiat-Shamir transformation our solution supports an online extractor which outputs the witness from such a non-interactive proof instantaneously, without having to rewind or fork. Additionally, the communication complexity of our solution is significantly lower than for previous proofs with online extractors. We furthermore give a superlogarithmic lower bound on the number of hash function evaluations for such online extractable proofs, matching the number in our construction, and we also show how to enhance security of the group signature scheme suggested recently by Boneh, Boyen and Shacham with our construction.

1 Introduction

The Fiat-Shamir transformation [17] is a well-known technique to remove interaction from proofs of knowledge and to derive signature schemes from such proofs. The starting point is a three-move proof between a prover, holding a witness w to a public value x, and a verifier. The prover sends a commitment com, then receives a random challenge ch from the verifier and finally replies with resp. For the non-interactive version the prover computes the challenge himself by applying a hash function H to the commitment. The security of this transformation has later been analyzed under the idealized assumption that the hash function behaves as a random oracle [8, 27], and has led to security proofs for related signature schemes.

Limitations. In the interactive case, all common knowledge extractors work by repeatedly rewinding the prover to the step after having sent com and completing the executions with independent random challenges. This eventually yields two valid executions $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$, $(\mathsf{com}, \mathsf{ch}', \mathsf{resp}')$ for different challenges $\mathsf{ch} \neq \mathsf{ch}'$ from which the extractor can compute the witness w. The same technique is reflected in the security proofs of the non-interactive version: The extractor

* This work was supported by the Emmy Noether Programme Fi 940/1-1 of the German Research Foundation (DFG).

continuously rewinds to the point where the prover has asked the random oracle H about com and completes the executions with independent hash values to find two valid executions (called "forking" in [27]).

The notable fact above is that, although the proof is non-interactive, the extractor still works by rewinding. As pointed out by [29] for example, this causes problems for some cryptographic schemes. Consider for example the ElGamal encryption $(R, C) = (g^r, pk^r \cdot m)$ for messages m. One attempt to make this scheme secure against chosen-ciphertext attacks is to append a non-interactive proof of knowledge for $r = \log R$ to the ciphertext. The idea is that, giving such a proof, any party generating a ciphertext would already "know" r and therefore $m = C \cdot pk^{-r}$. In other words, decryption queries in a chosen-ciphertext attack should be simulatable with the help of the knowledge extractor. However, this intuition cannot be turned into a proof, at least not with the rewinding extractor. Consider for example an adversary which sequentially puts n hash queries for the proofs of knowledge and then asks a decryption oracle for ciphertexts involving these hash queries *in reverse order*. Then, to answer each decryption query the extractor would have to rewind to the corresponding hash query. By this, it destroys all previously simulated decryption queries and must redo them from scratch, and the overall running time would become exponential in n.

We remark that the rewinding strategy also causes a loose security reduction. The results in [27] show that, if the adversary makes Q queries to the random oracle and forges, say, Schnorr signatures in time T with probability ϵ, then we can compute discrete logarithms in expected time QT/ϵ with constant probability. Hence, the number of hash queries enters multiplicatively in the time/success ratio. In contrast, for RSA-PSS and similar schemes [11, 12, 21] tight reductions are known. For other schemes like discrete-logarithm signatures different approaches relying on potentially stronger assumptions have been taken to get tight security results [18].

Constructing Online Extractors. The solution to the problems above is to use extractors which output the witness immediately, i.e., without having to rewind. Following the terminology of [29], where this problem was discussed but circumvented differently, we call them *online extractors.*[1] Informally, such an extractor is given the value x, a valid proof π produced by a prover and all hash queries and answers the prover made for generating this proof (i.e., even queries which are later ignored in the proof). The extractor then computes the witness w without further communication with the prover. Note that here we use the fact that we work in the random oracle model, where the extractor sees the hash queries.

One known possibility to build such online extractors is to use cut-and-choose techniques combined with hash trees [26, 22]. That is, one limits the challenge space to logarithmically many bits and repeats the following atomic protocol sufficiently often in parallel. The prover computes the initial commitment com of the interactive protocols and computes the answers for all possible challenges. Since there are only polynomially many challenges and answers, the prover can

[1] Sometimes such extractors are also called straight-line extractors in the literature.

build a binary hash tree with all answers at the leaves. Then he computes the actual challenge as the hash value over com and the root of the tree, and opens only the corresponding answer and all siblings on the path up to the root as a proof of correctness. For reasonable parameters these revealed hash values easily add about $10,000$ to $25,000$ bits to the non-interactive proof for all executions together, and thus cause a significant communication overhead.

Here we propose a different approach to build online extractors, producing much shorter proofs than the tree-based solution while having comparable extraction error and running time characteristics. In this introduction we provide a simplified description of our solution, omitting some necessary modifications. We also start with a polynomially bounded challenge space and a non-constant number of parallel executions. For each execution i the prover first computes com_i but now tries all polynomially many challenges $\mathsf{ch}_i = 0, 1, 2, \ldots$ and answers $\mathsf{resp}_i = \mathsf{resp}_i(\mathsf{ch}_i)$ till it finds one for which a predetermined (at most logarithmic) number of least significant bits of $H(x, \vec{\mathsf{com}}, i, \mathsf{ch}_i, \mathsf{resp}_i)$ are all zero. The prover outputs the vector $(\vec{\mathsf{com}}, \vec{\mathsf{ch}}, \vec{\mathsf{resp}})$, no further hash values need to be included, and the verifier now also checks in all executions that the lower bits of the hash values are zero.

The honest prover is able to find a convincing proof after a polynomial number of trials for each execution (except with negligible probability which can be adapted through parameters). It is also clear that any prover who probes at most one valid challenge-response pair for each execution most likely does not find a hash value with zero-bits.[2] If, on the other hand, the prover tries at least two samples, then the knowledge extractor can find them in the list of hash queries and compute the witness. It follows that the (online) extraction probability is negligibly close to the verifier's acceptance probability.

Our construction, outlined above, still requires a non-constant number of parallel repetitions in order to decrease the soundness error from polynomial to negligible. However, for proofs which are already based on small challenges, such as RSA with small exponents or "more quantum-resistant" alternatives like the recently proposed lattice-based proofs with bit challenges [24], several repetitions have to be carried out anyway, and our construction only yields an insignificant overhead in such cases. For other scenarios, like proofs of knowledge for discrete logarithms, the repetitions may still be acceptable, e.g., if the proof is only executed occasionally as for key registration. Alternatively, for the discrete logarithm for example, the prover can precompute the commitments $\mathsf{com}_i = g^{r_i}$ offline and the verifier is able to use batch verification techniques [6] to reduce the computational cost.

A Lower Bound. Both the hash-tree construction and our solution here require a non-constant number of repetitions of the atomic protocol. An interesting question is if one can reduce this number. As a step towards disproving this

[2] We presume that it is infeasible to find distinct responses to a single challenge. Indeed, this requirement is not necessary for the Fiat-Shamir transformation, yet all proofs we know of have this additional property.

we show that the number of hash function evaluations for the prover must be superlogarithmic in order to have an online extractor (unless finding witnesses is trivial, of course).[3] While this superlogarithmic bound would be straightforward if we restrict the hash function's output size to a few bits, our result holds independently of the length of hash values.

The proof of our lower bound requires that the knowledge extractor does not have the ability to choose the hash values. If we would allow the extractor to program the random oracle then we could apply the hash function to generate a common random string and run a non-interactive zero-knowledge proof of knowledge in the standard model (based on additional assumptions, though) [16]. For unrestricted (but polynomial) output length a single hash function evaluation for both the prover and verifier would then suffice. For non-programming extractors the number of hash function evaluations in our construction and the hash-tree solution are optimal with respect to general protocols.

A Word About Random Oracles. Our solution is given in the random oracle model, and a sequence of works [10, 20, 23, 4] has shown that constructions in this model may not yield a secure scheme in the real world when the oracle is instantiated by some function. It is therefore worthwhile to take a look at the way we utilize the random oracle. In our transformation we essentially use the random oracle as a predicate with the following properties: The only way to evaluate this predicate is by computing it explicitly (thus "knowing" the input), that predicate outcomes are well distributed (i.e., random), the predicate values for related inputs are uncorrelated.

In comparison to the Fiat-Shamir transformation our construction somewhat "decouples" the hash function from the protocol flow. Indeed, the dependency of the answer and the hash function in the Fiat-Shamir transformation is exploited by Goldwasser and Tauman [20] to prove insecurity of the random oracle approach for the transformation. Because of the aforementioned separation of the protocol flow and the hash function in our solution, the counterexample in [20] does not seem to carry over (yet, similar results may hold here as well). The point is that our solution is provided as an alternative to the Fiat-Shamir transformation, given one accepts the random oracle model as a viable way to design efficient non-interactive proofs. Finding truly efficient non-interactive proofs of knowledge without random oracles is still open.

Applications. Clearly, proofs of knowledge with online extractors are especially suitable for settings with concurrent executions such as key registration steps. As another, more specific example, we show that our method can be used to enhance security of the Boneh et al. group signature scheme [5]. Roughly, a group signature scheme allows a set of users to sign messages such that a signature does

[3] To be more precise, we give a slightly stronger result relating the number of hash queries of the verifier and the prover. This stronger result shows for example that hard relations cannot have efficient provers if the verifier only makes a constant number of hash function queries to verify proofs.

not reveal the actual signer, yet a group manager holding a secret information can revoke this anonymity and identify the true signer. A stringent formalization of these two properties, called full anonymity and full traceability, has been put forth by Bellare et al. [7].

Although achieving strong traceability guarantees the protocol by Boneh et al. only realizes a slightly weaker anonymity notion. In the original definition [7] anonymity of a signer of a message should hold even if an adversary can request the group manager to open identities for other signatures (thus resembling chosen-ciphertext attacks on encryption schemes). In the weaker model such open queries are not allowed, and this property is consequently called CPA-full-anonymity in [5].

Without going into technical details we remark that the weaker anonymity property in [5] originates from the underlying (variation of the) ElGamal encryption scheme and its CPA-security. A promising step towards fully anonymous group signature is therefore to turn the ElGamal encryption into a CCA-secure scheme. As explained before, standard Fiat-Shamir proofs of knowledge for the randomness used to generate ciphertexts do not work because of the rewinding problems. And although there is a very efficient method to secure basic ElGamal against chosen-ciphertexts in the random oracle model [1], this technique inherently destroys the homomorphic properties of the ciphertexts. But this homomorphic property is crucial to the design of the group signature scheme as it allows to efficiently prove relations about the encrypted message.

Proofs of knowledge with online extractors provide a general solution. However, since one of the initial motivations of [5] was to design a scheme with *short* signatures of a couple of hundred bits only, the aforementioned hash-tree based constructions with their significant communication overhead, for example, are prohibitively expensive. We show that with our protocol we obtain a fully-anonymous scheme and for reasonable parameters the length of signatures increases from $1,500$ to about $5,000$ bits. In comparison, the RSA-based group signature scheme in [2], presumably one of the most outstanding group signature schemes, still requires more than $10,000$ bits. Based on implementation results about elliptic curves [15], and the fact that the signer in the scheme by Ataniese et al. [2] cannot apply Chinese-Remainder techniques to compute the exponentiations with $1,000$ and more bits, we estimate that our variation of the Boneh group signature is still more efficient, despite the repetitions for the proof of knowledge. This is especially true for the verifier who can apply batch verification techniques on top.

Organization. In Section 2 we give the basic definitions of three-move Fiat-Shamir proofs of knowledge and non-interactive ones with online extractors in the random oracle model. The main part of the paper, Section 3, presents our construction and the lower bound. Some of the proofs and the construction of secure signature schemes from our solution have been omitted due to lack of space. Section 4 finally presents our enhancement of the Boneh et al. group signature scheme.

2 Definitions

A security parameter k in our setting is an arbitrary string describing general parameters. In the most simple case $k = 1^\kappa$ describes the length κ of the cryptographic primitives in unary. More generally, k can for example consist of the description of a group $\mathcal{G}$ of prime order q and generator g of that group, i.e., $k = (\mathcal{G}, q, g)$. The security parameter also describes a sequence of efficiently verifiable relations $W = (W_k)_k$.

A (possibly negative) function $f(k)$ is called *negligible* if $f(k) \leq 1/p(k)$ for any polynomial $p(k)$ and all sufficiently large k's. A function which is not negligible is called *noticeable*. For two functions f, g we denote by $f \gtrsim g$ the fact that $g - f$ is negligible. Accordingly, $f \approx g$ stands for $f \gtrsim g$ and $g \gtrsim f$. A function f with $f \gtrsim 1$ is called *overwhelming*.

We usually work in the random oracle model where parties have access to a random function H with some domain and range depending on k. We note that we do not let the relation W depend on the random oracle H in order to avoid "self-referencing" problems. We occasionally let an algorithm "output a random oracle", $H \leftarrow A$, meaning that A generates a description of a (pseudo)random function H.

We require some additional properties of the underlying Fiat-Shamir proof to make our transformation work. First, we need that the prover's initial commitment com has nontrivial entropy. This can be achieved easily by appending a superlogarithmic number of public random bits to com if necessary. Second, we need that the verifier sends a uniform bit string as the challenge ch; all common proofs have this property. Third, we require that the prover's response is quasi unique, i.e., it should be infeasible to find another valid response resp' to a proof $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$, even if one knows the witness. This holds for example if resp is uniquely determined by $x, \mathsf{com}, \mathsf{ch}$, e.g., as for the protocols by Guillou-Quisquater [19] and Schnorr [28], but also for Okamoto's witness-indistinguishable variations these protocols [25] (if the parameter k contains the system parameters like the RSA modulus N with unknown factorization).

Definition 1. *A Fiat-Shamir proof of knowledge (with $\ell(k)$-bit challenges) for relation W is a pair (P, V) of probabilistic polynomial-time algorithms $P = (P_0, P_1)$, $V = (V_0, V_1)$ with the following properties.*

[Completeness.] For any parameter k, any $(x, w) \in W_k$, any $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow (P(x, w), V_0(x))$ it holds $V_1(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1$.

[Commitment Entropy.] For parameter k, for any $(x, w) \in W_k$, the min-entropy of $\mathsf{com} \leftarrow P_0(x, w)$ is superlogarithmic in k.

[Public Coin.] For any k, any $(x, w) \in W_k$ any $\mathsf{com} \leftarrow P_0(x, w)$ the challenge $\mathsf{ch} \leftarrow V_0(x, \mathsf{com})$ is uniform on $\{0, 1\}^{\ell(k)}$.

[Unique Responses.] For any probabilistic polynomial-time algorithm A, for parameter k and $(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp}, \mathsf{resp}') \leftarrow A(k)$ we have, as a function of k,

$$\mathrm{Prob}[\, V_1(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = V_1(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = 1 \wedge \mathsf{resp} \neq \mathsf{resp}'] \approx 0.$$

[Special Soundness.] There exists a probabilistic polynomial-time algorithm K, the knowledge extractor, such that for any k, any $(x,w) \in W_k$, any pairs $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$, $(\mathsf{com}, \mathsf{ch}', \mathsf{resp}')$ *with* $V_1(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp}) = V_1(x, \mathsf{com}, \mathsf{ch}', \mathsf{resp}') = 1$ *and* $\mathsf{ch} \neq \mathsf{ch}'$, *for* $w' \leftarrow K(x, \mathsf{com}, \mathsf{ch}, \mathsf{resp}, \mathsf{ch}', \mathsf{resp}')$ *it holds* $(x, w') \in W_k$.

[Honest-Verifier Zero-Knowledge.] There exists a probabilistic polynomial-time algorithm Z, the zero-knowledge simulator, such that for any pair of probabilistic polynomial-time algorithms $D = (D_0, D_1)$ the following distributions are computationally indistinguishable[4]*:*

- *Let* $(x, w, \delta) \leftarrow D_0(k)$, *and* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow (P(x,w), V_0(x))$ *if* $(x,w) \in W_k$, *and* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \perp$ *otherwise. Output* $D_1(\mathsf{com}, \mathsf{ch}, \mathsf{resp}, \delta)$.
- *Let* $(x, w, \delta) \leftarrow D_0(k)$, *and* $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow Z(x, \text{YES})$ *if* $(x,w) \in W_k$, *and* $(\mathsf{com}\, ch, \mathsf{resp}) \leftarrow Z(x, \text{NO})$ *otherwise. Output* $D_1(\mathsf{com}, \mathsf{ch}, \mathsf{resp}, \delta)$.

Below we sometimes use a stronger kind of zero-knowledge property which basically says that the simulator is able to generate proofs for a specific challenge, as long as this challenge is given in advance. To formalize this let V_0^{CH} be a verifier which on input x, ch merely outputs ch. Then a Fiat-Shamir proof of knowledge (with $\ell(k)$-bit challenges) is *special* zero-knowledge if the following holds:

[Special Zero-Knowledge.] There exists a probabilistic polynomial-time algorithm Z, the special zero-knowledge simulator, such that for any pair of probabilistic polynomial-time algorithms $D = (D_0, D_1)$ the following distributions are computationally indistinguishable:

- Let $(x, w, \mathsf{ch}, \delta) \leftarrow D_0(k)$, and $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow (P(x,w), V_0^{\text{CH}}(x, \mathsf{ch}))$ if $(x,w) \in W_k$, and $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow \perp$ else. Output $D_1(\mathsf{com}, \mathsf{ch}, \mathsf{resp}, \delta)$.
- Let $(x, w, \mathsf{ch}, \delta) \leftarrow D_0(k)$, and $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow Z(x, \mathsf{ch}, \text{YES})$ if $(x,w) \in W_k$, and $(\mathsf{com}, \mathsf{ch}, \mathsf{resp}) \leftarrow Z(x, \mathsf{ch}, \text{NO})$ else. Output $D_1(\mathsf{com}, \mathsf{ch}, \mathsf{resp}, \delta)$.

We note that all common protocols obey this special zero-knowledge property. In fact, it is easy to see that *any* Fiat-Shamir proof of knowledge is special zero-knowledge if the challenge size $\ell(k) = O(\log k)$ is logarithmic (which holds for our transformation in the next section). The idea is to simply run many copies of the (regular) zero-knowledge simulator to find a transcript including a matching challenge.

We next define non-interactive proofs of knowledge with online extractors. We note that, in the random oracle model, it is easy to see that the verifier can be assumed wlog. to be deterministic.

Definition 2. *A pair (P, V) of probabilistic polynomial-time algorithms is called a non-interactive zero-knowledge proof of knowledge for relation W with an online extractor (in the random oracle model) if the following holds.*

[Completeness.] For any oracle H, any $(x,w) \in W_k$ and any $\pi \leftarrow P^H(x,w)$ we have $\text{Prob}\left[V^H(x,\pi) = 1\right] \gtrsim 1$.

[4] Meaning that the probability that D_1 outputs 1 is the same in both experiments, up to a negligible difference.

[Zero-Knowledge.] There exist a pair of probabilistic polynomial-time algorithms $Z = (Z_0, Z_1)$, the zero-knowledge simulator, such that for any pair of probabilistic polynomial-time algorithms $D = (D_0, D_1)$ the following distributions are computationally indistinguishable:

- *Let H be a random oracle, $(x, w, \delta) \leftarrow D_0^H(k)$, and $\pi \leftarrow P^H(x, w)$ if $(x, w) \in W_k$, and $\pi \leftarrow \perp$ otherwise. Output $D_1^H(\pi, \delta)$.*
- *Let $(H_0, \sigma) \leftarrow Z_0(k)$, $(x, w, \delta) \leftarrow D_0^{H_0}(k)$, and $(H_1, \pi) \leftarrow Z_1(\sigma, x, \text{YES})$ if $(x, w) \in W_k$, and $(H_1, \pi) \leftarrow Z_1(\sigma, x, \text{NO})$ otherwise. Output $D_1^{H_1}(\pi, \delta)$.*

[Online Extractor.] There exist a probabilistic polynomial-time algorithm K, the online extractor, such that the following holds for any algorithm A. Let H be a random oracle, $(x, \pi) \leftarrow A^H(k)$ and $\mathcal{Q}_H(A)$ be the sequence of queries of A to H and H's answers. Let $w \leftarrow K(x, \pi, \mathcal{Q}_H(A))$. Then, as a function of k,

$$\text{Prob}\left[(x, w) \notin W_k \wedge V^H(x, \pi) = 1\right] \approx 0.$$

Note that we allow the zero-knowledge simulator to program the random oracle, but only in two stages. Namely, Z_0 generates H_0 for D_0 and then Z_1 selects H_1 for the find-stage of D_1. Since the adversary D_0 in the first stage can pass on all interactions with H_0 to D_1 through the state information δ, the simulator Z_1 must guarantee that H_1 is consistent with H_0. However, Z_1 now has the opportunity to adapt oracle H_1 with respect to the adversarial chosen theorem x. Simulator Z_1 also gets the information whether x is in the language or not (in which case the simulator can simply set $\pi \leftarrow \perp$).

3 Constructions

Our starting point are interactive Fiat-Shamir proofs with logarithmic challenge length ℓ. Note that such proofs can be easily constructed from proofs with smaller challenge length l by combining $\lceil \ell/l \rceil$ parallel executions. It is easy to verify that all required properties are preserved by these parallel executions, including unique responses and honest-verifier zero-knowledge. Analogously, we can go the other direction and limit the challenge size to at most ℓ bits while conserving the properties.

3.1 Generic Construction

Recall the idea of our construction explained in the introduction. In each of the r repetitions we let the prover search through challenges and responses to find a tuple $(\mathsf{com}, \mathsf{ch}, \mathsf{resp})$ whose b least significant bits of the hash are 0^b for a small b. From now on we assume for simplicity that H only has b output bits; this can always be achieved by cutting off the leading bits.

Instead of demanding that all r hash values equal 0^b we give the honest prover more flexibility and let the verifier accept also proofs $(\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)_{i=1,2,\ldots,r}$ such that the sum of the r hash values $H(x, \vec{\mathsf{com}}, i, \mathsf{ch}_i, \mathsf{resp}_i)$ (viewed as natural numbers) does not exceed some parameter S. With this we can bound the prover's number of trials in each execution by 2^t for another parameter t, slightly larger than b, and guarantee that the prover terminates in strict polynomial time.

For sake of concreteness the reader may think of $b = 9$ (output length of the hash function), $t = 12$ (challenge size), $r = 10$ (number of repetitions) and $S = 10 = r$ (maximum sum). For these values the probability of the honest prover failing to find a valid proof is about 2^{-60}, and the knowledge extractor will obtain the witness whenever the proof is valid, except with probability approximately $Q \cdot 2^{-70}$ where Q denotes the number of hash queries the prover makes.

Construction 1. *Let $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ be an interactive Fiat-Shamir proof of knowledge with challenges of $\ell = \ell(k) = O(\log(k))$ bits for relation W. Define the parameters b, r, S, t (as functions of k) for the number of test bits, repetitions, maximum sum and trial bits such that $br = \omega(\log k)$, $2^{t-b} = \omega(\log k)$, $b, r, t = O(\log k)$, $S = O(r)$ and $b \leq t \leq \ell$. Define the following non-interactive proof system for relation W in the random oracle model, where the random oracle maps to b bits.*

[Prover.] The prover P^H on input (x, w) first runs the prover $P_{\mathsf{FS}}(x, w)$ in r independent repetitions to obtain r commitments $\mathsf{com}_1, \ldots, \mathsf{com}_r$. Let $\vec{\mathsf{com}}=(\mathsf{com}_1, \ldots, \mathsf{com}_r)$. Then P^H does the following, either sequentially or in parallel for each repetition i. For each $\mathsf{ch}_i = 0, 1, 2, \ldots, 2^t - 1$ (viewed as t-bit strings) it lets P_{FS} compute the final responses $\mathsf{resp}_i = \mathsf{resp}_i(\mathsf{ch}_i)$ by rewinding, until it finds the first one such that $H(x, \vec{\mathsf{com}}, i, \mathsf{ch}_i, \mathsf{resp}_i) = 0^b$; if no such tuple is found then P^H picks the first one for which the hash value is minimal among all 2^t hash values. The prover finally outputs $\pi = (\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)_{i=1,2,\ldots,r}$.

[Verifier.] The verifier V^H on input x and $\pi = (\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)_{i=1,2,\ldots,r}$ accepts if and only if $V_{1,\mathsf{FS}}(x, \mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i) = 1$ for each $i = 1, 2, \ldots, r$, and if $\sum_{i=1}^{r} H(x, \vec{\mathsf{com}}, i, \mathsf{ch}_i, \mathsf{resp}_i) \leq S$.

Note that for common iterated hash functions like SHA-1 the prover and the verifier can store the intermediate hash value of the prefix $(x, \vec{\mathsf{com}})$ and need not compute it from scratch for each of the r repetitions.

Our protocol has a small completeness error. For deterministic verifiers this error can be removed in principle by standard techniques, namely, by letting the prover check on behalf of the verifier that the proof is valid before outputting it; if not the prover simply sends the witness to the verifier. In practice, in case of this very unlikely event, the prover may just compute a proof from scratch.

Theorem 2. *Let $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ be an interactive Fiat-Shamir proof of knowledge for relation W. Then the scheme (P, V) in Construction 1 is a non-interactive zero-knowledge proof of knowledge for relation W (in the random oracle model) with an online extractor.*

Proof. (Sketch) We show that completeness, zero-knowledge and online extraction according to the definition are satisfied.

Completeness. For the completeness we show that the prover fails to convince the verifier with negligible probability only. For this let s_i be the random value $H(x, \vec{\mathsf{com}}, i, \mathsf{ch}_i, \mathsf{resp}_i)$ associated to the output of the i-th execution. Then,

$$\mathrm{Prob}[\,\exists i : \mathsf{s}_i > S] \le r \cdot \left(1 - (S+1)2^{-b}\right)^{2^t} \le r \cdot e^{-(S+1)2^{t-b}}$$

because in each of the at most 2^t tries the prover gets a random hash value of at most S with probability at least $(S+1)2^{-b}$, and all hash values are independent. The probability of having a value larger than S in one execution is thus negligible as r is logarithmic and 2^{t-b} is superlogarithmic. Hence, the sum of all r values exceeds rS with negligible probability only, and we from now on we can condition on the event that the sum of all s_i is at most rS. We also presume $r \ge 2$ in the sequel, else the claim already follows.

In order for the honest prover to fail the sum T of the r values $\mathsf{s}_1, \ldots, \mathsf{s}_r \ge 0$ must be larger than S. For any such $T = S+1, S+2, \ldots, rS$ there are at most $\binom{T+r-1}{r-1}$ ways to split the sum T into r non-negative integers $s_1, \ldots, s_r$. This is upper bounded by

$$\binom{T+r-1}{r-1} \le \left(\frac{e(rS+r-1)}{r-1}\right)^{r-1} \le (e(2S+1))^{r-1} \le e^{r\ln(e(2S+1))}$$

On the other hand, the probability of obtaining such a sum for a given partition, $\mathsf{s}_1 = s_1, \ldots, \mathsf{s}_r = s_r$, is at most

$$\begin{aligned}\prod_{i=1}^{r} \mathrm{Prob}[\,\mathsf{s}_i = s_i] \le \prod_{i=1}^{r} \mathrm{Prob}[\,\mathsf{s}_i \ge s_i] &\le \prod_{i=1}^{r} \left(1 - s_i 2^{-b}\right)^{2^t} \\ &\le \prod_{i=1}^{r} e^{-s_i 2^{t-b}} = e^{-(\sum s_i)2^{t-b}} = e^{-T2^{t-b}} \le e^{-(S+1)2^{t-b}}\end{aligned}$$

By choice of the parameters the probability of getting a sum T with $S < T \le rS$ is therefore limited by $\exp(r\ln(e(2S+1)) - (S+1)2^{t-b})$. Since $\ln(2S+1) \le S+1$, $r = O(\log k)$ and $2^{t-b} = \omega(\log k)$ this is negligible.

Zero-Knowledge. The zero-knowledge simulator $Z = (Z_0, Z_1)$ in the first stage simply lets H_0 be a (pseudo)random oracle. For the second stage, Z_1 defines H_1 to be consistent with H_0 on previous queries. For any other query to H_1 simulator Z_1, on input x (and YES, the case NO is trivial), first samples 2^t random b-bit strings for each i and assigns them to the t-bit challenges $\mathsf{ch}_i \in \{0,1\}^t$. Let $\tau_i : \{0,1\}^t \to \{0,1\}^b$ describe this assignment. Let ch_i be the first one (in lexicographic order) obtaining the minimum over all these 2^t values. Z_1 next runs the (wlog.) special zero-knowledge simulator Z_{FS} of the underlying Fiat-Shamir proof r times on x and each ch_i to obtain r tuples $(\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)$. It then defines the hash function H_1 for any query $(x, \vec{\mathsf{com}}, i, \mathsf{ch}_i^*, \mathsf{resp}_i^*)$ with $V_{1,\mathsf{FS}}(x, \mathsf{com}_i, \mathsf{ch}_i^*, \mathsf{resp}_i^*) = 1$ to be the value $\tau_i(\mathsf{ch}_i^*)$. All other values of H_1 are chosen (pseudo)randomly. The simulator outputs $\pi = (\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)_{i=1,2,\ldots,r}$ as the proof. Zero-knowledge of the simulator above follows from the special zero-knowledge property of the Fiat-Shamir protocol (together with a hybrid argument).

Online Extraction. We present a knowledge extractor $K(x, \pi, \mathcal{Q}_H(A))$ that, except with negligible probability over the choice of H, is able to output a witness w to x for an accepted proof $\pi = (\mathsf{com}_i, \mathsf{ch}_i, \mathsf{resp}_i)_{i=1,2,\ldots,r}$. Algorithm K browses through the list of queries and answers $\mathcal{Q}_H(A)$ and searches for a query $(x, \vec{\mathsf{com}}, i, \mathsf{ch}_i, \mathsf{resp}_i)$ as well another query $(x, \vec{\mathsf{com}}, i, \mathsf{ch}'_i, \mathsf{resp}'_i)$ for a different challenge $\mathsf{ch}_i \neq \mathsf{ch}'_i$ but such that $V_{\mathsf{FS}}(x, \mathsf{com}_i, \mathsf{ch}'_i, \mathsf{resp}'_i) = 1$. If it finds two such queries it runs the knowledge extractor K_{FS} of the Fiat-Shamir proof on these values and copies its output; if there are no such queries then K outputs $\perp$. It is clear that K succeeds every time it finds two valid queries for the same prefix $(x, \vec{\mathsf{com}}, i)$. On the other hand, by the choice of parameters the probability of making the verifier accept while probing at most one challenge-response pair is negligible. $\square$

We remark that the upper bounds derived on the number of representations of T with r integers, for completeness and extraction, have not been optimized. Moreover, we providently note that our knowledge extractor only needs the hash queries in $\mathcal{Q}_H(A)$ with prefix $(x, \vec{\mathsf{com}})$ to extract the witness for theorem x; all other queries are irrelevant to K.

Comparison to Hash-Tree Construction. We compare our construction with online extractors based on hash tress. Recall that, for the hash tree construction, in each of the r repetitions the prover computes the commitment com and all possible responses $\mathsf{resp}(\mathsf{ch})$ for challenges $\mathsf{ch} \in \{0,1\}^b$. Hash values of all 2^b responses are placed as leaves in a hash tree, and a binary tree of depth b is computed. This requires altogether $2^b + 2^b - 1 \approx 2^{b+1}$ hash function evaluations. The challenge is computed as the hash value over all commitments and tree roots, and in each tree the corresponding leaf is opened together with the siblings on the path.

To compare the efficiency of the two approaches, we set $b = 9$, $t = 12$, $r = 10$ and $S = 10$ for our construction and $b' = 8$ and $r' = 10$ for the hash-tree construction. Then the total number of hash function evaluations is roughly $r \cdot 2^9$ in both cases, and the number of executions of the underlying protocol are identical. In favor of the hash tree construction it must be said that our solution requires twice as many response computations on the average, though.

We have already remarked that the communication complexity of the hash-tree construction is significantly larger than for our construction, i.e., the partly disclosed hash trees add $br = 90$ hash values (typically of 160 or more bits) to the proof, while our solution does not add any communication overhead. As for the extraction error, the exact analysis for our construction with the given parameters shows that the extractor fails with probability at most $Q \cdot 2^{-72}$ where Q is the maximal number of hash queries (assuming that finding distinct responses is beyond feasibility). The extraction for the hash-tree construction basically fails only if one manages to guess all r challenges in advance and to put only one correct answer in each tree. This happens with probability approximately $Q/2^{br} = Q \cdot 2^{-80}$ and is only slightly smaller than for our construction. Yet, extraction in the hash-tree construction also requires that no collisions for the hash function are found. Finally, we note that the honest prover always manages to convince the honest verifier for the hash-tree construction whereas our protocol has a small completeness error.

Properties. Concerning the type of zero-knowledge, if there is a unique response for each $x, \mathsf{com}, \mathsf{ch}$, then our transformation converts an honest-verifier perfect zero-knowledge protocol into a statistical zero-knowledge one (against malicious verifiers). The small error is due to the negligible collision probability of commitments and applies to the standard Fiat-Shamir transformation as well.

As for proving logical combinations, given two interactive Fiat-Shamir protocols for two relations W^0, W^1 it is known [13, 9, 14] how to construct three-move proofs showing that one knows at least one of the witnesses to x^0, x^1 (i.e., prove OR), or one can also show that one knows both witnesses (i.e., prove AND). Since the derived protocols in both cases preserve the zero-knowledge and extraction property, and therefore constitute themselves Fiat-Shamir proofs of knowledge, our conversion can also be carried out for proving such logical statements.

Simulation Soundness and Secure Signatures. Our proof system even achieves the stronger notion of simulation soundness, i.e., even if the zero-knowledge simulator has simulated several proofs for adversarial chosen theorems, the online extractor can still extract the witness from the adversarial proof for a valid theorem (as long as either the theorem or the proof is new). It is then straightforward to construct a secure signature scheme from this simulation sound proof of knowledge, with a tight security reduction. The formal description is omitted.

3.2 Lower Bound for Hash Queries of Online Extractors

In this section we show our superlogarithmic lower bound on the number of hash function evaluations for non-programming online extractors. For notational convenience we let $f(k) = O_{\mathsf{K}}(\log k)$ or $f(k) = poly_{\mathsf{K}}(k)$ refer to a function $f(k)$ which grows only logarithmically or polynomially, restricted to all $k \in \mathsf{K}$, i.e., there is a constant c such that $f(k) \leq c \log k$ or $f(k) \leq k^c$ for all $k \in \mathsf{K}$. For any $k \notin \mathsf{K}$ the function f might exceed these bounds.

For our result we assume that the underlying relation W of the proof of knowledge is accompanied by an efficiently samplable, yet hard to invert procedure generating (x, w). For example, for the discrete-logarithm problem and parameter $k = (\mathcal{G}, q, g)$ this procedure picks $w \leftarrow \mathbb{Z}_q$ and computes $x \leftarrow g^w$. More formally, we say that the relation W has a *one-way instance generator* $\mathcal{I}$ if for any parameter k algorithm $\mathcal{I}$ returns in probabilistic polynomial-time $(x, w) \in W_k$, but such that for any probabilistic polynomial-time algorithm I, for $(x, w) \leftarrow \mathcal{I}(k)$ and $w' \leftarrow I(x)$ the probability $\mathrm{Prob}[\,(x, w') \in W_k]$ is negligible (as a function of k).

Proposition 1. *Let (P, V) be a non-interactive zero-knowledge proof of knowledge for relation W with an online extractor K in the random oracle model. Let $\rho = \rho(k)$ and $\nu = \nu(k)$ be the maximum number of hash oracle queries the prover P resp. the verifier V makes to generate and to check a proof π. Then*

$\max_{v=0,1,\ldots,\nu} \binom{\rho}{v} = poly_{\mathsf{K}}(k)$ *for an infinite set* K *implies that* W *does not have a one-way instance generator* $\mathcal{I}$.

Clearly, $\binom{\rho}{v}$ obtains its maximum at $\binom{\rho}{\lceil \rho/2 \rfloor}$, where $\lceil \rho/2 \rfloor$ is the rounded-off integer of $\rho/2$, and if $\rho = O_{\mathsf{K}}(\log k)$ for an infinite set K, then $\binom{\rho}{\lceil \rho/2 \rfloor} \leq (2e)^{\lceil \rho/2 \rfloor} = poly_{\mathsf{K}}(k)$ for the same set K, and the requirements of the proposition are satisfied. This implies that $\rho = \omega(\log k)$ must grow superlogarithmically for a one-way instance generator. Similarly, if the verifier only makes a constant number of hash function queries then the prover must perform a superpolynomial number of hash function evaluations, or else the instance generator cannot be one-way.

Proof. (Sketch) The high level idea of the proof is that, under the assumption that $\max_v \binom{\rho}{v} = poly_{\mathsf{K}}(k)$ is polynomial, we can imagine that the prover tries to guess in advance the verifier's queries (among the ρ queries) and only makes those queries. This strategy will succeed with sufficiently large probability by assumption. Then, replacing the hash queries $\mathcal{Q}_H(P)$ the prover makes to generate the proof π by the queries $\mathcal{Q}_H(V)$ the verifier makes to verify the proof suffices to extract the witness. Specifically, we prove that $K(x, \pi, \mathcal{Q}_H(V))$ then returns a witness w with noticeable probability. Replacing the original proof by an indistinguishable one from the zero-knowledge simulator $Z(x)$ (without access to w) and running $K(x, \pi, \mathcal{Q}_H(V))$ on this proof implies that we can compute the witness w with noticeable probability from x alone. □

Optimality of the Bound. Our lower bounds make essential use of the fact that the extractor cannot program the random oracle. In fact, if K was allowed to choose oracle values, then the oracle H (with unrestricted output length) could be defined to generate a sufficiently large common reference string and to run a non-interactive zero-knowledge proof of knowledge with online extractor in the standard model [16]. A single hash function evaluation would then suffice.

Also, the superlogarithmic bound cannot be improved for non-programming extractors. Namely, if we run the hash-tree construction or an easy modification of our solution for binary challenges and superlogarithmic r, then we get a negligible extraction error and make only $O(r)$ hash function queries.

4 Application to Group Signatures

In this section we show how to lift the CPA-anonymous group signature scheme by Boneh et al. [5] to a fully anonymous one. As explained in the introduction, the idea is to append a non-interactive proof of knowledge with online extractor for an ElGamal-like encryption. Although we give a brief introduction to group signatures we refer the reader to the work by Bellare et al. [7] for a comprehensive overview about (the security of) group signatures. Recall from the introduction that the two important security properties are full anonymity, the impossibility of identifying the author of a signature, and full traceability, the impossibility of generating a signature which cannot be traced to the right origin.

Very roughly, a group signature scheme consists of a (fixed) set of users, each user receiving a secret through an initial *key generation* phase carried out by a trusted third party. In addition, a public group key is established in this phase. Each user can run the *sign protocol* to generate a signature on behalf of the group. This signature is *verifiable* through the group's public key, yet outsiders remain oblivious about the actual signer. Only the group manager can revoke this anonymity and *open* the signature through an additional secret key.

The original scheme by Boneh works over bilinear group pairs (G_1, G_2) where deciding the Diffie-Hellman problem is easy. That is, for groups G_1, G_2 of prime order q generated by g_1, g_2 there is an efficiently computable isomorphism $\psi : G_2 \to G_1$ with $\psi(g_2) = g_1$, and an efficiently computable non-degenerated bilinear mapping e with $e(u^a, v^b) = e(u, v)^{ab}$ for all $u \in G_1$, $v \in G_2$ and $a, b \in \mathbb{Z}_q$.

For the security of the scheme it is assumed that the q-strong Diffie-Hellman problem —given $(g_1, g_2, g_2^{\gamma}, g_2^{(\gamma^2)}, \ldots, g_2^{(\gamma^q)})$ find $(g_1^{\gamma+x}, x)$ for *any* $x \in \mathbb{Z}_q^*$— is intractable. See [3] for more details. It is also presumed that the decision linear assumption in G_1 holds, namely that it is infeasible to distinguish tuples $(u, v, h, u^a, v^b, h^{a+b})$ and (u, v, h, u^a, v^b, h^c) for $u, v, h \leftarrow G_1$ and $a, b, c \in \mathbb{Z}_q$. This assumption implies that ElGamal-like encryptions $(u^a, v^b, h^{a+b} \cdot m)$ of messages m under public key (u, v, h) are semantically secure.

In the original scheme of Boneh et al. [5] the group's public key contains a value $w = g_2^{\gamma}$ and each user receives a pair (A_i, x_i) with $A_i = g_1^{1/(\gamma+x_i)}$ as the secret key. In addition, the group manager's public key consists of a public encryption key (u, v, h) such that $u^{\xi_1} = v^{\xi_2} = h$ for secret key ξ_1, ξ_2. To sign a message m the user encrypts A_i with the manager's public key as $T_1 \leftarrow u^a$, $T_2 \leftarrow v^b$ and $T_3 \leftarrow A_i h^{a+b}$ for random $a, b \leftarrow \mathbb{Z}_q$. In addition, the signer also computes a non-interactive proof τ (in the random oracle model) that (T_1, T_2, T_3) encrypts such an A_i with $e(A_i, w g_2^{x_i}) = e(g_1, g_2)$. The details of this zero-knowledge proof are irrelevant for our discussion here, we merely remark that the message m enters in this proof and that an independent random oracle is needed for this part. To verify a signature one verifies this proof τ. To revoke anonymity the group manager verifies the signatures and then decrypts $A_i = T_3 / T_1^{\xi_1} T_2^{\xi_2}$ and recovers the user's identity through A_i.

We now augment the original scheme by our proof of knowledge for the ElGamal encryption:

Construction 3. *Define the following group signature scheme:*

[Key Generation.] Compute the public key as before by picking a bilinear group pair $\mathcal{G} = (G_1, G_2)$ and generators g_1, g_2, h. Sample $\xi_1, \xi_2, \gamma \leftarrow \mathbb{Z}_q^*$ and let $u^{\xi_1} = v^{\xi_2} = h$ and $w = g_2^{\gamma}$. The public key gpk consists of $(\mathcal{G}, g_1, g_2, h, u, v, w)$. Each of the n users obtains some $x_i \leftarrow \mathbb{Z}_q^*$ and $A_i = g_1^{1/(\gamma+x_i)}$ as the secret key. The group manager receives $(\xi_1, \xi_2, A_1, \ldots, A_n)$ as the secret key.

[Signing.] To sign a message $m \in \{0,1\}^*$ under a secret key (A_i, x_i) the user takes the group key $gpk = (\mathcal{G}, g_1, g_2, h, u, v, w)$ and does the following:

- *As in the original scheme pick $a, b \leftarrow \mathbb{Z}_q$ and encrypt A_i under the group manager's public key, $T_1 \leftarrow u^a$, $T_2 \leftarrow v^b$ and $T_3 \leftarrow A_i h^{a+b}$.*

- *Compute as before a non-interactive proof τ that $A_i = g_1^{1/(\gamma+x_i)}$ is encrypted in (T_1, T_2, T_3) for some $x_i \in \mathbb{Z}_q$, involving the message m.*
- *Additionally, compute a non-interactive zero-knowledge proof of knowledge π for α, β, i.e., run P^H on $(gpk, T_1, T_2, T_3, \tau, m, \alpha, \beta)$ for relation $W_k = \left\{((gpk, T_1, T_2, T_3, \tau, m), (\alpha, \beta)) \mid u^\alpha = T_1, v^\beta = T_2\right\}$ to obtain π.*
- *Output $(T_1, T_2, T_3, \pi, \tau)$ as the signature to m.*

[Verification.] To verify a signature $(T_1, T_2, T_3, \pi, \tau)$ for a message m run the original verifier of the signature scheme and also run the verifier V^H of the non-interactive proof of knowledge on $(gpk, (T_1, T_2, T_3, \tau, m), \pi)$. Accept if both verifications succeed.

[Open.] To reveal the identity of a signer for a signature $(T_1, T_2, T_3, \tau, \pi)$ the group manager first verifies the validity of the signature (including the proof π). If correct, then the manager decrypts as in the original scheme to recover some $A = T_3/(T_1^{\xi_1} T_2^{\xi_2})$ and compares this value to the list of A_i's to find the user index i.

For system parameters suggested in [5], namely, $|q| = 170$ bits and $|G_1| = 171$ bits, the original signature length is $1,533$ bits. If we use the same values $b = 9, r = S = 10, t = 12$ as in the previous section for our proof system, then our scheme adds about $2r \cdot 170 + rt = 3,520$ bits to signatures through the r repetitions of the atomic protocol for proving the AND of the two discrete logarithms. This proof requires $2r$ answers in $\mathbb{Z}_q$ (as usual in the discrete logarithm case, the commitments are not included in the proof π) and r challenges of t bits. Although the communication complexity of this new scheme is significantly larger, it is still superior to RSA-based group signatures where signatures easily exceed $10,000$ bits [2].

Interestingly, we still expect our version of the group signature scheme to be more efficient than the RSA-based scheme in [2], where half a dozen exponentiations with large exponents of more than thousand bits have to be carried out without Chinese Remainder. According to implementation results in [15] a single exponentiation for elliptic curves is estimated to be about ten times faster than such RSA exponentiations; the exact figures of course depend on implementation details. The proof of the following proposition is omitted for space reasons.

Proposition 2. *Under the Decision Linear Diffie-Hellman and the q-strong Diffie-Hellman assumption the group signature scheme in Construction 3 is a fully anonymous and fully traceable group signature scheme in the random oracle model.*

Acknowledgments

We thank the anonymous reviewers of Crypto 2005 for very comprehensive comments and suggestions.

References

1. Masayuki Abe. *Combining Encryption and Proof of Knowledge in the Random Oracle Model. The Computer Journal*, 47(1):58–70, 2004.
2. Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. *A Practical and Provably Secure Coalition-Resistant Group Signature Scheme.* Advances in Cryptology — Crypto 2000, Volume 1880 of Lecture Notes in Computer Science. Springer-Verlag, 2000.
3. Dan Boneh and Xavier Boyen. *Short Signatures Without Random Oracles.* Advances in Cryptology — Eurocrypt 2004, Volume 3027 of Lecture Notes in Computer Science, pages 56–73. Springer-Verlag, 2004.
4. Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. *An Un-Instantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem.* Advances in Cryptology — Eurocrypt 2004, Volume 3027 of Lecture Notes in Computer Science. Springer-Verlag, 2004.
5. Dan Boneh, Xavier Boyen, and Hovav Shacham. *Short Group Signatures.* Advances in Cryptology — Crypto 2004, Volume 3152 of Lecture Notes in Computer Science, pages 41–55. Springer-Verlag, 2004.
6. Mihir Bellare, Juan Garay, and Tal Rabin. *Fast Batch Verification for Modular Exponentiation and Digital Signatures.* Advances in Cryptology — Eurocrypt'98, Volume 1403 of Lecture Notes in Computer Science, pages 236–250. Springer-Verlag, 1998.
7. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. *Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions.* Advances in Cryptology — Eurocrypt 2003, Volume 2656 of Lecture Notes in Computer Science, pages 614–629. Springer-Verlag, 2003.
8. M. Bellare and P. Rogaway. *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.* Proceedings of the Annual Conference on Computer and Communications Security (CCS). ACM Press, 1993.
9. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. *Proofs of Partial Knowledge and Simplified Desing of Witness Hiding Protocols.* Advances in Cryptology — Crypto'94, Volume 839 of Lecture Notes in Computer Science, pages 174–187. Springer-Verlag, 1995.
10. Ran Canetti, Oded Goldreich, and Shai Halevi. *The Random Oracle Methodology, Revisited.* Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1998, pages 209–218. ACM Press, 1998.
11. Jean-Sebastien Coron. *On the Exact Security of Full Domain Hash.* Advances in Cryptology — Crypto 2000, Volume 1880 of Lecture Notes in Computer Science, pages 229–235. Springer-Verlag, 2000.
12. Jean-Sebastien Coron. *Optimal Security Proofs for PSS and Other Signature Schemes.* Advances in Cryptology — Eurocrypt 2002, Volume 2332 of Lecture Notes in Computer Science, pages 272–287. Springer-Verlag, 2002.
13. David Chaum and Torben Pedersen. *Wallet Databases with Observers.* Advances in Cryptology — Crypto'92, Volume 740 of Lecture Notes in Computer Science, pages 89–105. Springer-Verlag, 1992.
14. Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. *On Monotone Formula Closure of SZK.* Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)'94, pages 454–465. IEEE Computer Society Press, 1994.

15. Erik De Win, Serge Mister, Bart Preneel, and Michael Wiener. *On the Performance of Signature Schemes Based on Elliptic Curves.* Algorithmic Number Theory Symposium — ANTS–III, Volume 1423 of Lecture Notes in Computer Science, pages 252–266. Springer-Verlag, 1998.
16. Alfredo De Santis and Giuseppe Persiano. *Zero-Knowledge Proofs of Knowledge Without Interaction.* Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)'92, pages 427–436. IEEE Computer Society Press, 1992.
17. A. Fiat and A. Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Schemes.* Advances in Cryptology — Crypto'86, Volume 263 of Lecture Notes in Computer Science, pages 186–194. Springer-Verlag, 1986.
18. Eu-Jin Goh and Stanislaw Jarecki. *Signature Scheme as Secure as the Diffie-Hellman Problem.* Advances in Cryptology — Eurocrypt 2003, Volume 2656 of Lecture Notes in Computer Science, pages 401–415. Springer-Verlag, 2003.
19. Louis Guillou and Jean-Jacques Quisquater. *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Trasmission and Memory.* Advances in Cryptology — Eurocrypt'88, Volume 330 of Lecture Notes in Computer Science, pages 123–128. Springer-Verlag, 1988.
20. Shafi Goldwasser and Yael Tauman. *On the (In)security of the Fiat-Shamir Paradigm.* Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 2003, pages 102–113. IEEE Computer Society Press, 2003.
21. Jonothan Katz and Nan Wang. *Efficiency Improvement for Signature Schemes with Tight Security Reductions.* Proceedings of the Annual Conference on Computer and Communications Security (CCS). ACM Press, 2003.
22. R. Merkle. *A Digital Signature Based on a Conventional Encryption Function.* Advances in Cryptology — Crypto'87, Volume 293 of Lecture Notes in Computer Science, pages 369–378. Springer-Verlag, 1988.
23. Ueli Maurer, Renato Renner, and Clemens Holenstein. *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology.* Theory of Cryptography Conference (TCC) 2004, Volume 2951 of Lecture Notes in Computer Science, pages 21–39. Springer-Verlag, 2004.
24. Daniele Micciancio and Salil Vadhan. *Statistical Zero-Knowledge Proofs with Efficient Provers: Lattice Problems and More.* Advances in Cryptology — Crypto 2003, Volume 2729 of Lecture Notes in Computer Science, pages 282–298. Springer-Verlag, 2003.
25. T. Okamoto. *Provable Secure and Practical Identification Schemes and Corresponding Signature Schemes.* Advances in Cryptology — Crypto'92, Volume 740 of Lecture Notes in Computer Science, pages 31–53. Springer-Verlag, 1992.
26. Rafael Pass. *On Deniability in the Common Reference String and Random Oracle Model.* Advances in Cryptology — Crypto 2003, Volume 2729 of Lecture Notes in Computer Science, pages 316–337. Springer-Verlag, 2003.
27. David Pointcheval and Jacques Stern. *Security Arguments for Digital Signatures and Blind Signatures.* *Journal of Cryptology*, 13(3):361–396, 2000.
28. C.P. Schnorr. *Efficient Signature Generation by Smart Cards.* *Journal of Cryptology*, 4:161–174, 1991.
29. Victor Shoup and Rosario Gennaro. *Securing Threshold Cryptosystems against Chosen Ciphertext Attack.* *Journal of Cryptology*, 15(2):75–96, 2002.