Perfect Non-interactive Zero Knowledge for NP

Jens Groth*, Rafail Ostrovsky**, and Amit Sahai***

UCLA, Computer Science Department,
4732 Boelter Hall,
Los Angeles, CA 90095, USA
{jg, rafail, sahai}@cs.ucla.edu

Abstract. Non-interactive zero-knowledge (NIZK) proof systems are fundamental cryptographic primitives used in many constructions, including CCA2-secure cryptosystems, digital signatures, and various cryptographic protocols. What makes them especially attractive, is that they work equally well in a concurrent setting, which is notoriously hard for interactive zero-knowledge protocols. However, while for interactive zero-knowledge we know how to construct statistical zero-knowledge argument systems for all NP languages, for non-interactive zero-knowledge, this problem remained open since the inception of NIZK in the late 1980's. Here we resolve two problems regarding NIZK:

- We construct the first perfect NIZK argument system for any NP language.
- We construct the first UC-secure NIZK argument for any NP language in the presence of a dynamic/adaptive adversary.

While it is already known how to construct efficient prover computational NIZK *proofs* for any NP language, the known techniques yield large common reference strings and large proofs. Another contribution of this paper is NIZK proofs with much shorter common reference string and proofs than previous constructions.

Keywords: Non-interactive zero-knowledge, universal composability, non-malleability.

1 Introduction

In this paper, we resolve a central open problem concerning Non-Interactive Zero-Knowledge (NIZK) protocols: how to construct *statistical* NIZK arguments for any NP language. While for *interactive* zero knowledge (ZK), it has long been known how to construct statistical zero-knowledge argument systems for all NP languages [5], for NIZK this question has remained open for nearly two decades.

* Supported by NSF Cybertrust ITR grant No. 0456717.

** Supported in part by a gift from Teradata, Intel equipment grant, NSF Cybertrust grant No. 0430254, OKAWA research award, B. John Garrick Foundation and Xerox Innovation group Award.

*** Supported by NSF Cybertrust ITR grant No. 0456717, an equipment grant from Intel, and an Alfred P. Sloan Foundation Research Fellowship.

S. Vaudenay (Ed.): EUROCRYPT 2006, LNCS 4004, pp. 339–358, 2006.

In context with previous work – statistical zero knowledge: Blum, Feldman, and Micali [3] introduced the notion of NIZK in the common random string model and showed how to construct *computational* NIZK proof systems for proving a single statement about any NP language. The first computational NIZK proof system for multiple theorems was constructed by Blum, De Santis, Micali, and Persiano [2]. Both [3] and [2] based their NIZK systems on certain number-theoretic assumptions (specifically, the hardness of deciding quadratic residues modulo a composite number). Feige, Lapidot, and Shamir [18] showed how to construct computational NIZK proofs based on any trapdoor permutation.

The above work, and the plethora of research on NIZK that followed, mainly considered NIZK where the zero-knowledge property was only true *computationally*; that is, a computationally bounded party cannot extract any information beyond the correctness of the theorem being proven. In the case of *interactive* zero knowledge, it has long been known that all NP statements can in fact be proven using *statistical* (in fact, perfect) zero knowledge arguments [6, 5]; that is, even a computationally unbounded party would not learn anything beyond the correctness of the theorem being proven, though we must assume that the prover, *only during the execution of the protocol*, is computationally bounded to ensure soundness[1].

Achieving statistical NIZK has been an elusive goal. The original work of [3] showed how a computationally unbounded prover can prove to a polynomially bounded verifier that a number is a quadratic-residue, where the zero-knowledge property is perfect. Statistical ZK (including statistical NIZK[2]) for any nontrivial language for both proofs and arguments were shown to imply the existence of a one-way function by Ostrovsky [29]. Statistical NIZK proof systems were further explored by De Santis, Di Crescenzo, Persiano, and Yung [14] and Goldreich, Sahai, and Vadhan [23], who gave complete problems for the complexity class associated with statistical NIZK proofs. However, these works came far short of working for all NP languages, and in fact NP-complete languages cannot have (even interactive) statistical zero-knowledge proof systems unless the polynomial hierarchy collapses [19, 1][3]. Unless our computational complexity beliefs are wrong, this leaves open only the possibility of argument systems.

Do there exist *statistical* NIZK arguments for all NP languages? Despite nearly two decades of research on NIZK, the answer to this question was not known. In this paper, we answer this question in the affirmative, based on a number-theoretic complexity assumption introduced in [4].

[1] Such systems where the soundness holds computationally have come to be known as *argument systems*, as opposed to *proof systems* where the soundness condition must hold unconditionally.

[2] We note that the result of [29] is for *honest-verifier* SZK, and does not require the simulator to produce Verifier's random tape, and therefore it includes NIZK, even for the common reference string which is not uniform. See also [31] for an alternative proof.

[3] See also [22] appendix regarding subtleties of this proof, and [33] for an alternative proof.

Our results. Our main results, which we describe in more detail below, are:

- Significantly more efficient NIZK *proofs* for circuit satisfiability.
- Perfect NIZK *arguments* for any NP language.
- UC-secure perfect NIZK arguments for any NP language, secure against adaptive/dynamic adversaries.

In our second result, we prove that our perfect NIZK argument has non-adaptive soundness, i.e., it is infeasible to forge a proof for a false statement that is chosen independently of the common reference string. In our third result, however, we construct a UC-secure perfect NIZK argument (i.e., in which the adversary sees the common reference string first, and can adaptively choose theorems and try to forge arguments afterwards)[4].

NIZK proofs. As a building block we start by constructing a simple and efficient computational NIZK proof of knowledge for circuit satisfiability, based on the subgroup decision problem introduced in [4]. To the best of our knowledge, our techniques are completely different from all previous constructions of NIZK proofs, which use the hidden bits model. In this NIZK proof system, the size of the common reference string is $\mathcal{O}(k)$, where k is the security parameter; thus it is independent of the size of the NP statements. The NIZK proofs have size $\mathcal{O}(k|C|)$, where $|C|$ is the size of the circuit. This is a significant result in its own right; the most efficient NIZK proof systems for an NP-complete problem with efficient provers previously known [27] required a reference string of size at least $\mathcal{O}(k^3)$ and the NIZK proofs of size at least $\mathcal{O}(|C|k^2)$. For comparison with the most efficient previous work, please see Table 1.

Table 1. Comparison of CRS size and NIZK proof size for efficient-prover NIZK proof systems for circuit satisfiability

Reference	CRS size	Proof Size	Assumption
Damgård [9]	$O(\|C\|k^2+k^3)$	$O(\|C\|k^2+k^3)$	Quadratic Residuosity
Kilian-Petrank [27]	$O(\|C\|k^2)$	$O(\|C\|k^2)$	Trapdoor Permutations
Kilian-Petrank [27]	$O(k^3)$	$O(\|C\|k^3)$	Trapdoor Permutations
De Santis et al. [12, 13]	$O(k+\|C\|^{\varepsilon})$	poly($\|C\|k$)	NIZK & One-Way Functions
This paper	$O(k)$	$O(\|C\|k)$	Subgroup Decision [4]

Perfect NIZK arguments. The NIZK proofs we construct are built using encryptions of the bits in the circuit. However, by a slight modification to only

[4] In general, there is an interesting and subtle technical point regarding the UC framework that should be pointed out here: while the UC framework does rule out proofs of false theorems in the ideal model, it does not explicitly rule out the possibility of proofs of false theorems in the real world. Instead, since the ideal and real world executions are indistinguishable, the UC framework rules out the possibility that the adversary (or the environement) can gain any *advantage* from proofs of false theorems that it manages to generate in the real world.

the reference string, we effectively transform the cryptosystem into a perfectly hiding commitment scheme. With this transformation, we obtain a perfect NIZK argument for circuit satisfiability.

UC-SECURE PERFECT NIZK ARGUMENT. We generalize our techniques to construct perfect NIZK arguments that satisfy Canetti's UC definition of security. Canetti introduced the universal composability (UC) framework [7] as a general method to argue security of protocols in an arbitrary environment. It is a strong security definition; in particular it implies non-malleability [17], and security when arbitrary protocols are executed concurrently.

We define NIZK arguments in the UC framework and construct a NIZK argument that satisfies the UC security definition. From the theory behind the UC framework, this means that we can plug in our NIZK argument in arbitrary settings and maintain soundness and privacy. At the same time, we can prove that our UC NIZK argument enjoys a perfect zero-knowledge property.

We stress that our result of prefect NIZK in the UC framework holds even in the setting of dynamic/adaptive adversaries without erasures: where the adversary can corrupt parties adaptively, and upon corruption of a party, it learns the entire history of the internal state of this party. Prior to our result, no NIZK protocol was known to be UC-secure against dynamic/adaptive adversaries. In [8], it was observed that De Santis et al. [11] achieve UC-security, but only for the setting with *static* adversaries (in the non-erasure model).

2 Non-interactive Zero-Knowledge

Let R be an efficiently computable binary relation. For pairs $(x, w) \in R$ we call x the statement and w the witness. Let L be the language consisting of statements in R.

An argument system for a relation R consists of a key generation algorithm K, a prover P and a verifier V. The key generation algorithm produces a common reference string σ. The prover takes as input (σ, x, w) and checks whether $(x, w) \in R$. In that case, it produces a proof or argument π, otherwise it outputs `failure`. The verifier takes as input (σ, x, π) and outputs 1 if the proof is acceptable and 0 if rejecting the proof. We call (K, P, V) an argument system for R if it has the completeness and soundness properties described below.

(PERFECT) COMPLETENESS. For all $(x, w) \in R$, we have

$$\Pr\left[\sigma \leftarrow K(1^k); \pi \leftarrow P(\sigma, x, w) : V(\sigma, x, \pi) = 1 \text{ if } (x, w) \in R\right] = 1.$$

(NON-ADAPTIVE COMPUTATIONAL) SOUNDNESS. For all non-uniform polynomial time adversaries $\mathcal{A}$ and $x \notin L$, we have

$$\Pr\left[\sigma \leftarrow K(1^k); \pi \leftarrow \mathcal{A}(x, \sigma) : V(\sigma, x, \pi) = 0\right] \approx 1.$$

We call (K, P, V) a proof system for R if soundness also holds for computationally unbounded adversaries.

(Perfect) Knowledge Extraction. We call (K, P, V) a proof of knowledge for R if there exists a knowledge extractor $E = (E_1, E_2)$ with the properties described below.

For all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Pr\left[\sigma \leftarrow K(1^k) : \mathcal{A}(\sigma) = 1\right] = \Pr\left[(\sigma, \tau) \leftarrow E_1(1^k) : \mathcal{A}(\sigma) = 1\right]$$

For all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Pr\Big[(\sigma, \tau) \leftarrow E_1(1^k); (x, \pi) \leftarrow \mathcal{A}(\sigma); w \leftarrow E_2(\sigma, \tau, x, \pi) : \\ V(\sigma, x, \pi) = 0 \text{ or } (x, w) \in R\Big] = 1.$$

Since perfect knowledge extraction implies the existence of a witness for the statement being proven, it implies perfect adaptive soundness.

(Adaptive Multi-theorem) Zero-Knowledge. We call (K, P, V) a NIZK argument or NIZK proof for R if there exists a simulator $S = (S_1, S_2)$ with the following zero-knowledge property. For all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Pr\left[\sigma \leftarrow K(1^k) : \mathcal{A}^{P(\sigma,\cdot,\cdot)}(\sigma) = 1\right] \approx \Pr\left[(\sigma, \tau) \leftarrow S_1(1^k) : \mathcal{A}^{S'(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right],$$

where $S'(\sigma, \tau, x, w) = S_2(\sigma, \tau, x)$ for $(x, w) \in R$ and outputs `failure` if $(x, w) \notin R$.

Honest prover state reconstruction. In modeling adaptive UC security without erasures, an honest prover may be corrupted at some time. To handle such cases, we want to extend the zero-knowledge property such that not only can we simulate an honest party making a proof, we also want to be able to simulate how it constructed the proof. In other words, once the party is corrupted, the adversary will learn the witness and the randomness used; we want to create convincing randomness so that it looks like the simulated proof was constructed by an honest prover using this randomness.

We say a NIZK argument or proof for R has honest prover state reconstruction if there exists a simulator $S = (S_1, S_2, S_3)$ so for all $\mathcal{A}$ we have

$$\Pr\left[\sigma \leftarrow K(1^k) : \mathcal{A}^{PR(\sigma,\cdot,\cdot)}(\sigma) = 1\right] \approx \Pr\left[(\sigma, \tau) \leftarrow S_1(1^k) : \mathcal{A}^{SR(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right],$$

where $PR(\sigma, x, w)$ runs $r \leftarrow \{0,1\}^{\ell_P(k)}; \pi \leftarrow P(\sigma, x, w; r)$ and returns π, r, and where SR runs $\rho \leftarrow \{0,1\}^{\ell_S(k)}; \pi \leftarrow S_2(\sigma, \tau, x; \rho); r \leftarrow S_3(\sigma, \tau, x, w, \rho)$ and returns π, r, both of the oracles outputting `failure` if $(x, w) \notin R$.

Perfect completeness, soundness, knowledge extraction and zero-knowledge. We speak of perfect completeness, perfect soundness, perfect knowledge extraction, perfect zero-knowledge and perfect honest prover state reconstruction if for sufficiently large security parameters we have equalities in the respective definitions.

3 The Boneh-Goh-Nissim Cryptosystem

Boneh, Goh and Nissim [4] suggest a cryptosystem with interesting homomorphic properties. The BGN-cryptosystem is the main building block in the paper.

Bilinear groups. We use two cyclic groups $\mathbb{G}, \mathbb{G}_1$ of order n, where $n = pq$ and p, q are primes. We make use of a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. I.e., for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$ we have $e(u^a, v^b) = e(u, v)^{ab}$. We require that $e(g, g)$ is a generator of $\mathbb{G}_1$ if g is a generator of $\mathbb{G}$. We also require that group operations, group membership, sampling of a random generator for $\mathbb{G}$ and the bilinear map be efficiently computable.

[4] suggest the following example. Pick large primes p, q and let $n = pq$. Find the smallest ℓ so $P = \ell n - 1$ is prime and equal to 2 modulo 3. Consider the points on the elliptic curve $y^2 = x^3 + 1$ over $\mathbb{F}_P$. This curve has $P + 1 = \ell n$ points, so it has a subgroup $\mathbb{G}$ of order n. We let $\mathbb{G}_1$ be the order n subgroup of $\mathbb{F}^*_{P^2}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be the modified Weil-pairing.

The subgroup decision problem. Let $\mathcal{G}$ be an algorithm that takes a security parameter as input and outputs $(p, q, \mathbb{G}, \mathbb{G}_1, e)$ such that p, q are primes, $n = pq$ and $\mathbb{G}, \mathbb{G}_1$ are descriptions of groups of order n and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map.

Let $\mathbb{G}_q$ be the subgroup of $\mathbb{G}$ of order q. The subgroup decision problem is to distinguish elements of $\mathbb{G}$ from elements of $\mathbb{G}_q$. Let $\mathbb{G}_{gen}$ be the generators of $\mathbb{G}$.

Definition 1. *The subgroup decision assumption holds for generator $\mathcal{G}$ if there exists a negligible function $\nu_{SD} : \mathbb{N} \to [0; 1]$ so for any non-uniform polynomial time adversary $\mathcal{A}$ we have*

$$\begin{aligned} &\Pr\Big[(p, q, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(1^k); n = pq; g, h \leftarrow \mathbb{G}_{gen} : \mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, g, h) = 1\Big] \\ -&\Pr\Big[(p, q, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(1^k); n = pq; g \leftarrow \mathbb{G}_{gen}, h \leftarrow \mathbb{G}_q \setminus \{1\} : \\ &\qquad\qquad \mathcal{A}(n, \mathbb{G}, \mathbb{G}_1, e, g, h) = 1\Big] \quad < \quad \nu_{SD}(k). \end{aligned}$$

We remark that we have changed the wording of the subgroup decision problem slightly in comparison with [4], but the definitions are equivalent.

The BGN-cryptosystem. We generate a public key by running $(p, q, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(1^k)$, setting $n = pq$, selecting g as a random generator of $\mathbb{G}$ and h as a random generator of $\mathbb{G}_q$. The public key is $(n, \mathbb{G}, \mathbb{G}_1, e, g, h)$ while the decryption key is p, q.

To encrypt a message m of length $\mathcal{O}(\log k)$ using randomness $r \leftarrow \mathbb{Z}^*_n$ we compute the ciphertext $c = g^m h^r$. To decrypt we compute $c^q = g^{mq} h^{mq} = (g^q)^m$ and exhaustively search for m.

By the subgroup decision assumption, we could indistinguishably select h to be a random generator of $\mathbb{G}$ as well. In this case, we do not have a cryptosystem but rather a perfectly hiding trapdoor commitment scheme.

4 Non-interactive Zero-Knowledge Proof

4.1 NIZK Proof That c Encrypts 0 or 1

We will construct a NIZK proof of knowledge for circuit satisfiability in Section 4.2. As a building block in this NIZK proof, we will encrypt the truth-values of the wires in the circuit. We need to convince the verifier that these ciphertexts have been correctly formed. We therefore start by constructing a NIZK proof that a BGN-ciphertext has either 0 or 1 as plaintext.

We observe that a ciphertext c contains 0 or 1, if and only if $c \in \mathbb{G}_q$ or $cg^{-1} \in \mathbb{G}_q$. Our strategy is therefore to show that $e(c, cg^{-1})$ has order q. If we know m, w so $c = g^m h^w$ then $m = 0$ implies $e(c, cg^{-1}) = e(h^w, g^{-1}h^w) = e(h, (g^{-1}h^w)^w)$ and $m = 1$ means $e(c, cg^{-1}) = e(gh^w, h^w) = e(h, (gh^w)^w)$. In both cases we get $e(c, cg^{-1}) = e(h, (g^{2m-1}h^w)^w)$. Since h has order q, revealing the two components will immediately convince the verifier that $e(c, cg^{-1})$ has order q, however may not be zero-knowledge.

Instead, we make a NIZK proof for $e(c, cg^{-1})$ having order q as follows. We choose a random exponent r and compute $e(c, cg^{-1}) = e(h^r, (g^{2m-1}h^w)^{wr^{-1}})$. We reveal these two components, and must convince the verifier that the first element $\pi_1 = h^r$ has order q. For this purpose, we show him the element g^r. Since $e(\pi_1, g) = e(h^r, g) = e(h, g^r)$ the verifier can now tell that π_1 has order q.

To argue zero-knowledge we change the public key. Instead of having h of order q, we use h of order n and select g so we know the discrete logarithm. Now all ciphertexts are perfectly hiding commitments so we can create all of them as encryptions of 0. We can simulate the revelation of g^r because we know the discrete logarithm.

Common reference string:
1. $(p, q, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(1^k)$
2. $n = pq$
3. g random generator of $\mathbb{G}$
4. h random generator of $\mathbb{G}_q$
5. Return $\sigma = (n, \mathbb{G}, \mathbb{G}_1, e, g, h)$.

Statement: The statement is an element $c \in \mathbb{G}$. The claim is that there exists a pair $(m, w) \in \mathbb{Z}^2$ so $m \in \{0, 1\}$ and $c = g^m h^w$.

Proof: Input $(\sigma, c, (m, w))$.
1. Check $m \in \{0, 1\}$ and $c = g^m h^w$. Return `failure` if check fails.
2. $r \leftarrow \mathbb{Z}_n^*$
3. $\pi_1 = h^r, \pi_2 = (g^{2m-1}h^w)^{wr^{-1}}, \pi_3 = g^r$
4. Return $\pi = (\pi_1, \pi_2, \pi_3)$

Verification: Input $(\sigma, c, \pi = (\pi_1, \pi_2, \pi_3))$.
1. Check $c \in \mathbb{G}$ and $\pi \in \mathbb{G}^3$
2. Check $e(c, cg^{-1}) = e(\pi_1, \pi_2)$ and $e(\pi_1, g) = e(h, \pi_3)$
3. Return 1 if both checks pass, else return 0

Fig. 1. NIZK proof of plaintext being zero or one

Theorem 1. *The protocol in Figure 1 is a NIZK proof that $c \in \mathbb{G}$ has plaintext $m \in \{0,1\}$. The NIZK proof has perfect completeness, perfect soundness and computational zero-knowledge and honest prover state reconstruction.*

Proof. PERFECT COMPLETENESS. We know that $c = g^m h^w$, where $m \in \{0,1\}$. This gives us $e(c, cg^{-1}) = e(h, (g^{2m-1}h^w)^w) = e(h^r, (g^{2m-1}h^w)^{wr^{-1}}) = e(\pi_1, \pi_2)$. Furthermore, $e(\pi_1, g) = e(h^r, g) = e(h, g^r) = e(h, \pi_3)$.

PERFECT SOUNDNESS. We have $e(\pi_1^q, g) = e(\pi_1, g)^q = e(h, \pi_3)^q = e(h^q, \pi_3) = e(1, \pi_3) = 1$. Therefore, π_1 must have order 1 or q. This means $e(c, cg^{-1})^q = e(\pi_1, \pi_2)^q = e(\pi_1^q, \pi_2) = 1$, implying that c or cg^{-1} has order 1 or q.

COMPUTATIONAL ZERO-KNOWLEDGE AND HONEST PROVER STATE RECONSTRUCTION. First, we describe the simulator $S = (S_1, S_2, S_3)$. S_1 runs the algorithm for generating the common reference string with the following modification. It selects h to be a random generator for $\mathbb{G}$ and sets $g = h^\gamma$, where $\gamma \leftarrow \mathbb{Z}_n^*$. During the generation of the common reference string the simulator also learns p, q. S_1 outputs $(\sigma, \tau) = ((n, \mathbb{G}, \mathbb{G}_1, e, g, h), (p, q, \gamma))$.

S_2 on input (σ, τ, c) simulates a proof as follows. Either c, cg^{-1}, or both are generators for $\mathbb{G}$. The simulator picks $r \leftarrow \mathbb{Z}_n^*$. If c is a generator it sets $\pi_1 = c^r, \pi_2 = (cg^{-1})^{r^{-1}}$ and $\pi_3 = \pi_1^\gamma$. If c is not a generator for the group, then the simulator sets $\pi_1 = (cg^{-1})^r, \pi_2 = c^{r^{-1}}, \pi_3 = \pi_1^\gamma$.

S_3 is given the witness (m, w) so $c = g^m h^w$ and $m \in \{0,1\}$ and wishes to reconstruct how the prover could have come up with the proof π. Since it knows γ it can write $c = h^{\gamma m + w}$. Consider first the case where c is a generator for $\mathbb{G}$, then we have $\gcd(n, \gamma m + w) = 1$. So we can write the proof as $\pi_1 = h^{r(\gamma m + w)}, \pi_2 = (g^{2m-1}h^w)^{w(r(\gamma m+w))^{-1}}, \pi_3 = g^{r(\gamma m + w)}$. We return $r(\gamma m + w) \bmod n$ as the prover's simulated randomness that would cause it to produce π. In case c is not a generator, we know that cg^{-1} is a generator and we write the proof as $\pi_1 = h^{r(\gamma(m-1)+w)}, \pi_2 = (g^{2m-1}h^w)^{w(r(\gamma(m-1)+w))^{-1}}, \pi_3 = g^{r(\gamma(m-1)+w)}$ and return $r(\gamma(m-1) + w) \bmod n$ as the prover's simulated randomness.

To argue computational zero-knowledge we consider a hybrid experiment, where we use S_1 to generate the common reference string σ, but implement the simulation oracle using the real prover P. We first show that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Big| \Pr\left[\sigma \leftarrow K(1^k) : \mathcal{A}^{PR(\sigma,\cdot,\cdot)}(\sigma) = 1\right] - \Pr\left[(\sigma,\tau) \leftarrow S_1(1^k) : \mathcal{A}^{PR(\sigma,\cdot,\cdot)}(\sigma) = 1\right] \Big| < \nu_{SD}(k),$$

where $PR(\sigma, (\sigma, c), (m, w))$ runs $r \leftarrow \mathbb{Z}_n^*; \pi \leftarrow P(\sigma, (\sigma, c), (m, w); r)$ and returns π, r, and outputs `failure` if $m \notin \{0,1\}$ or $c \neq g^m h^w$.

The only difference between the two experiments is the choice of h. In one case, h is a random generator of $\mathbb{G}$ in the other case it is a generator of $\mathbb{G}_q$. We do not use the knowledge of p, q or the discrete logarithm of g with respect to h in either experiment. Consider now a subgroup decision problem challenge $(n, \mathbb{G}, \mathbb{G}_1, e, g, h)$. The challenges correspond exactly to common reference strings produced by respectively K and S_1. The advantage of $\mathcal{A}$ is therefore bounded by $\nu_{SD}(k)$.

Next, we go from the hybrid experiment to the simulation. For all $\mathcal{A}$ we have

$$\begin{aligned}&\Pr\left[(\sigma,\tau)\leftarrow S_1(1^k):\mathcal{A}^{PR(\sigma,\cdot,\cdot)}(\sigma)=1\right]\\ =\ &\Pr\left[(\sigma,\tau)\leftarrow S_1(1^k):\mathcal{A}^{SR(\sigma,\tau,\cdot,\cdot)}(\sigma)=1\right],\end{aligned}$$

where SR runs $\rho\leftarrow\mathbb{Z}_n^*;\pi\leftarrow S_2(\sigma,\tau,(\sigma,c);\rho);r\leftarrow S_3(\sigma,\tau,(\sigma,c),(m,w),\rho)$ and returns π,r, or `failure` if $m\notin\{0,1\}$ or $c\neq g^m h^w$.

A simulated proof $\pi=(\pi_1,\pi_2,\pi_3)$ uniquely defines the randomness $r\in\mathbb{Z}_n^*$ so $\pi_1=h^r$, and it is indeed this randomness S_3 outputs. We therefore just need to argue that simulated proofs have the same distribution as real proofs in the hybrid experiment. In case c is a generator for $\mathbb{G}$, S_2 selects $r\leftarrow\mathbb{Z}_n^*$ at random and set $\pi_1=c^r$, which gives us a random generator of $\mathbb{G}$. In a real prover's proof π_1 is also a random generator of $\mathbb{G}$ when h has order n. Since π_1 uniquely defines π_2 and π_3, we see that the two distributions are identical. If c is not a generator for $\mathbb{G}$, then cg^{-1} and since a simulated $\pi_1=(cg^{-1})^r$ for $r\leftarrow\mathbb{Z}_n^*$ is a random generator of $\mathbb{G}$, we can use a similar argument to show that also in this case we get a perfect simulation. □

4.2 NIZK Proof of Knowledge for Circuit Satisfiability

Suppose we have a circuit C and want to prove that there exists w so $C(w)=1$. Since any circuit can be linearly reduced to a circuit built only from NAND-gates, we will without loss of generality focus on this simpler case.

To prove satisfiability of C we encrypt the bit value of each wire, when the circuit is evaluated on the input bits in w. Using the NIZK proof in Figure 1 it is straightforward to prove that all ciphertexts contain a plaintext in $\{0,1\}$. We form the output ciphertext with randomness 0 so it is straightforward for the verifier to check that the output of the circuit is 1.

The only thing left is to prove that all the encrypted output wires do indeed evaluate the NAND-gates correctly. We make the following observation, leaving the proof to the reader.

Lemma 1. *Let* $b_0,b_1,b_2\in\{0,1\}$.

$$b_0+b_1+2b_2-2\in\{0,1\}\text{ if and only if }b_2=\neg(b_0\wedge b_1).$$

Given ciphertexts c_0,c_1,c_2 containing plaintexts b_0,b_1,b_2 we can use the homomorphic property to form the ciphertext $c_0c_1c_2^2g^{-2}$. A NIZK proof that $c_0c_1c_2^2g^{-2}$ contains a plaintext in $\{0,1\}$ implies $b_2=\neg(b_0\wedge b_1)$, as required. We make such a NIZK proof for each NAND-gate in the circuit.

Theorem 2. *The protocol in Figure 2 is a NIZK proof of knowledge of circuit satisfiability. It has perfect completeness, perfect soundness, perfect knowledge extraction and computational zero-knowledge and honest prover state reconstruction.*

Common reference string:

1. $(p, q, \mathbb{G}, \mathbb{G}_1, e) \leftarrow \mathcal{G}(1^k)$
2. $n = pq$
3. g random generator of $\mathbb{G}$
4. h random generator of $\mathbb{G}_q$
5. Return $\sigma = (n, \mathbb{G}, \mathbb{G}_1, e, g, h)$.

Statement: The statement is a circuit C built from NAND-gates. The claim is that there exist input bits w so $C(w) = 1$.

Proof: The prover has a witness w consisting of input bits so $C(w) = 1$.

1. Extend w to contain the bits of all wires in the circuit.
2. Encrypt each bit w_i as $c_i = g^{w_i} h^{r_i}$, with $r_i \leftarrow \mathbb{Z}_n^*$.
3. For all c_i make a NIZK proof of existence of w_i, r_i so $w_i = \{0, 1\}$ and $c_i = g^{w_i} h^{r_i}$.
4. For the output of the circuit we let the ciphertext be $c_{\text{output}} = g$, i.e., an easily verifiable encryption of 1.
5. For all NAND-gates, we do the following. We have input ciphertexts c_{i_0}, c_{i_1} and output ciphertexts c_{i_2}. We wish to prove the existence of $w_{i_0}, w_{i_1}, w_{i_2} \in \{0, 1\}$ and $r_{i_0}, r_{i_1}, r_{i_2}$ so $w_2 = \neg(w_0 \wedge w_1)$ and $c_{i_j} = g^{w_{i_j}} h^{r_{i_j}}$. To do so we make a NIZK proof that there exist m, r with $m \in \{0, 1\}$ so $c_{i_0} c_{i_1} c_{i_2}^2 g^{-2} = g^m h^r$.
6. Return π consisting of all the ciphertexts and NIZK proofs.

Verification: The verifier given a circuit C and a proof π.

1. Check that all wires have a corresponding ciphertext and that the output wire's ciphertext is g.
2. Check that all ciphertexts have a NIZK proof of the plaintext being 0 or 1.
3. Check that all NAND-gates have a valid NIZK proof of compliance.
4. Return 1 if all checks pass, else return 0.

Fig. 2. NIZK proof for circuit satisfiability

Proof. PERFECT COMPLETENESS. Knowing a satisfying assignment w for C, we can compute truth-values for all wires that are consistent with the NAND-gates and make the circuit have 1 as output. Perfect completeness follows from the perfect completeness of the NIZK proofs of plaintexts being either 0 or 1.

PERFECT SOUNDNESS. Since we prove for each wire that the encrypted plaintext is either 0 or 1, we have made a perfectly binding commitment to a bit for each wire. By Lemma 1, the NIZK proofs for the gates imply that all encrypted wire-bits respect the NAND-gates. Finally, we know that the output ciphertext is g, so the output bit is 1.

PERFECT KNOWLEDGE EXTRACTION. The extractor sets up the common reference string by running the key generator for the NIZK proof. In the process it learns p, q. This allows it to decrypt the ciphertexts containing the input-bits. Since the NIZK proof has perfect soundness, these input bits must correspond to a witness w so $C(w) = 1$.

COMPUTATIONAL ZERO-KNOWLEDGE AND HONEST PROVER STATE RECONSTRUCTION. Let S_1 be the simulator of the NIZK proof for a ciphertext having 0 or 1 as plaintext. We use the same algorithm to create the common reference string for simulation of circuit satisfiability NIZK proofs. In other words, both g, h are random generators of $\mathbb{G}$ and the simulator knows $\gamma \in \mathbb{Z}_n^*$ so $g = h^\gamma$.

S_2 starts by choosing the ciphertexts for the wires: The output wire gets the ciphertext g. For all other wires, it selects a ciphertext $c_i = h^{r_i}$ with $r_i \leftarrow \mathbb{Z}_n^*$. Later, when S_3 learns a witness w, it can compute the corresponding messages $m_i \in \{0,1\}$ for all these ciphertexts, and open them as $c_i = g^{m_i} h^{r_i - m_i \gamma^{-1}}$.

For all these ciphertexts S_2 simulates a NIZK proof that they contain 0 or 1 as the plaintext. Also for all NAND-gates with input wires i_0, i_1 and output wire i_2 it simulates a NIZK proof that $c_{i_0} c_{i_1} c_{i_2}^2 g^{-2}$ contains a plaintext that is 0 or 1. Later, upon learning the witness w, S_3 knows the plaintexts $w_{i_j} \in \{0,1\}$ and randomizers $r_{i_j} - w_{i_j}\gamma^{-1}$ that constitute a satisfactory encryption of the wires of a satisfied circuit. For each NIZK proof of a plaintext being 0 or 1, S_3 can run the honest prover state reconstructor to get convincing randomness that would make the prover produce this proof.

To prove that this is a good simulation, we first consider a hybrid experiment where we use the simulator to create the common reference string, but use the real prover to create the NIZK proofs. As in the proof of Theorem 1, we can argue that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Big| \Pr\left[\sigma \leftarrow K(1^k) : \mathcal{A}^{PR(\sigma,\cdot,\cdot)}(\sigma) = 1\right] \\ - \Pr\left[(\sigma,\tau) \leftarrow S_1(1^k) : \mathcal{A}^{PR(\sigma,\cdot,\cdot)}(\sigma) = 1\right] \Big| < \nu_{SD}(k),$$

where $PR(\sigma, C, w)$ runs $\pi \leftarrow P(\sigma, C, w; r)$ and returns π, r.

Next, we modify the way we create proofs. Instead of running the real prover, we create the encryptions of the wires c_i as the real prover, but simulate the NIZK proofs of 0 or 1 being the plaintext and simulate the NIZK proofs for the NAND-gates as well. From the proof of Theorem 1 we get that this modification does not increase $\mathcal{A}$'s probability of outputting 1. We have

$$\Pr\left[(\sigma,\tau) \leftarrow S_1(1^k) : \mathcal{A}^{PR(\sigma,\cdot,\cdot)}(\sigma) = 1\right] \\ = \Pr\left[(\sigma,\tau) \leftarrow S_1(1^k) : \mathcal{A}^{PSR(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right],$$

where $PSR(\sigma, \tau, C, w)$ creates ciphertexts c_i correctly but simulates NIZK proofs for 0- or 1-plaintexts and the randomness involved, and outputs `failure` if $C(w) \neq 1$.

Finally, we go to the full simulation. For all $\mathcal{A}$ we have

$$\Pr\left[(\sigma,\tau) \leftarrow S_1(1^k) : \mathcal{A}^{PSR(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right] \\ = \Pr\left[(\sigma,\tau) \leftarrow S_1(1^k) : \mathcal{A}^{SR(\sigma,\tau,\cdot,\cdot)}(\sigma) = 1\right],$$

where SR runs $\pi \leftarrow S_2(\sigma, \tau, C; \rho); r \leftarrow S_3(\sigma, \tau, C, w, \rho)$ and returns π, r, and outputs `failure` if $C(w) \neq 1$. The only difference here is in the way we create the ciphertexts, but since they are perfectly hiding, we cannot distinguish the two experiments. □

5 Non-interactive Statistical Zero-Knowledge Argument

In this section, we construct a NIZK argument of circuit satisfiability with perfect zero-knowledge. The main idea is a simple modification of the NIZK proof for circuit satisfiability in Figure 2. Instead of choosing h of order q, we let h be a random generator of $\mathbb{G}$. This way $g^m h^r$ is no longer an encryption of m, but a perfectly hiding commitment to m. It corresponds to using S_1 restricted to the first half of its outputs as key generator. Completeness is obvious and the proof of Theorem 2 reveals that the argument is perfect zero-knowledge.

Soundness is also simple enough. Suppose we have circuit $C \notin L$ generated independently of the common reference string. We can argue that no adversary can distinguish an h of order n from an h of order q, and therefore by Theorem 2 has negligible probability of making an acceptable NIZK argument.

Let S_σ be the simulator S_1 from the proof of Theorem 2 restricted to its first output. We have the following theorem

Theorem 3. *(S_σ, P, V) is a NIZK argument for circuit satisfiability.*

Proof. As in the proof of Theorem 2, we can show that the protocol has perfect completeness. Perfect zero-knowledge and honest prover state reconstruction follows from the proof of Theorem 2. This leaves us with the question of soundness.

SOUNDNESS. We first demonstrate that the NIZK argument is sound, i.e., for any fixed false statement, all adversaries have negligible probability of generating a valid proof of this statement.

Consider any unsatisfiable circuit C and a polynomial time adversary $\mathcal{A}$ that with probability So-Adv$_\mathcal{A}(1^k)$ breaks the soundness property. In other words, $\mathcal{A}$ is given a common reference string and proceeds to output a valid argument π. We will construct an adversary $\mathcal{B}$ that decides the subgroup decision problem with probability SD-Adv$_\mathcal{B}(1^k)$ =So-Adv$_\mathcal{A}(1^k)$.

$\mathcal{B}$ gets a challenge $(n, \mathbb{G}, \mathbb{G}_1, e, g, h)$ and has to decide whether h has order n or not. This corresponds to a common reference string generated by either K or S_σ. So we can give it to $\mathcal{A}$ and output 1 if and only if $\mathcal{A}$ forms a valid argument for C being true.

In case h has order n, the common reference string produced by $\mathcal{B}$ is distributed exactly as in a real argument. The adversary therefore has probability So-Adv$_\mathcal{A}(1^k)$ of generating an acceptable argument.

On the other hand, in case h has order q the common reference string produced by $\mathcal{B}$ is distributed as the reference string in the previously described NIZK proof. Since the NIZK proof has perfect soundness, the probability of $\mathcal{A}$ producing a valid argument is 0. □

6 Universally Composable Non-interactive Zero-Knowledge

6.1 Modeling Non-interactive Zero-Knowledge Arguments

The universal composability (UC) framework (see [7] for a detailed description) is a strong security model capturing security of a protocol under concurrent execution of arbitrary protocols. We model all other things not directly related to the protocol through a polynomial time environment. The environment can at its own choosing give inputs to the parties running the protocol, and according to the protocol specification, the parties can give outputs to the environment. In addition, there is a non-uniform polynomial time adversary $\mathcal{A}$ that attacks the protocol. $\mathcal{A}$ can communicate freely with the environment. It can also corrupt parties, in which case it learns the entire history of that party and gains complete control over the actions of this party.

To model security we use a simulation paradigm. We specify the functionality $\mathcal{F}$ that the protocol should realize. The functionality $\mathcal{F}$ can be seen as a trusted party that handles the entire protocol execution and tells the parties what they would output if they executed the protocol correctly. In the ideal process, the parties simply pass on inputs from environment to $\mathcal{F}$ and whenever receiving a message from $\mathcal{F}$ they output it to the environment. In the ideal process, we have an ideal process adversary $\mathcal{S}$. $\mathcal{S}$ does not learn the content of messages sent from $\mathcal{F}$ to the parties, but is in control of when, if ever, a message from $\mathcal{F}$ is delivered to the designated party. $\mathcal{S}$ can corrupt parties, at the time of corruption it will learn all inputs the party has received and all outputs it has sent to the environment. As the real world adversary, $\mathcal{S}$ can freely communicate with the environment.

We now compare these two models and say that it is secure if no environment can distinguish between the two worlds. This means, the protocol is secure, if for any $\mathcal{A}$ running in the real world, there exists an $\mathcal{S}$ running in the ideal process with $\mathcal{F}$ so no environment can distinguish between the two worlds.

The standard zero-knowledge functionality $\mathcal{F}_{ZK}$ as defined in [7] goes as follows: On input $(\mathbf{prove}, P, V, sid, ssid, x, w)$ from P the functionality $\mathcal{F}_{ZK}$ checks that $(x, w) \in R$ and in that case sends $(\mathbf{proof}, P, V, sid, ssid, x)$ to V. It is thus part of the model that the prover will send the proof to a particular receiver and that this receiver will learn who the prover is. This is a very reasonable model when we talk about interactive zero-knowledge proofs of knowledge. We remark that with only small modifications in the UC NIZK argument that we are about to suggest we could securely realize this functionality.

Parameterized with relation R and running with parties $P_1, \ldots, P_n$ and adversary $\mathcal{S}$.

Proof: On input $(\mathbf{prove}, sid, ssid, x, w)$ from party P ignore if $(x, w) \notin R$. Send $(\mathbf{prove}, x)$ to $\mathcal{S}$ and wait for answer $(\mathbf{proof}, \pi)$. Upon receiving the answer store (x, π) and send $(\mathbf{proof}, sid, ssid, \pi)$ to P.

Verification: On input $(\mathbf{verify}, sid, ssid, x, \pi)$ from V check whether (x, π) is stored. If not send $(\mathbf{verify}, x, \pi)$ to $\mathcal{S}$ and wait for an answer $(\mathbf{witness}, w)$. Upon receiving the answer, check whether $(x, w) \in R$ and in that case, store (x, π). If (x, π) has been stored return $(\mathbf{verification}, sid, ssid, 1)$ to V, else return $(\mathbf{verification}, sid, ssid, 0)$.

Fig. 3. NIZK argument functionality $\mathcal{F}_{\mathrm{NIZK}}$

However, when we talk about NIZK arguments we do not always know who is going to receive the NIZK argument. We simply create a string π, which is the NIZK argument. We may create this string in advance and later decide to whom to send it. Furthermore, anybody who intercepts the string π can verify the truth of the statement and can use the string to convince others about the truth of the statement. The NIZK argument is not deniable; quite on the contrary, it is transferable [30]. For this reason, and because the protocol and the security proof becomes a little simpler, we suggest a different functionality $\mathcal{F}_{\text{NIZK}}$ to capture the essence of NIZK arguments.

6.2 Tools

We will need a few cryptographic tools to securely realize $\mathcal{F}_{\text{NIZK}}$.

Perfectly hiding commitment scheme with extraction. A perfectly hiding commitment scheme with extraction (first used in [16] in the setting of perfectly hiding non-malleable commitment) has the following property. We can run a key generation algorithm $hk \leftarrow K_{\text{hiding}}(1^k)$ to get a hiding key hk, or we can alternatively run a key generation algorithm $(hk, xk) \leftarrow K_{\text{extract}}(1^k)$ in which case we get both a hiding key hk and an extraction key xk. $(K_{\text{hiding}}, \text{com})$ constitute a perfectly hiding commitment scheme. On the other hand, $(K_{\text{extract}}, \text{com}, \text{dec})$ constitute a public key cryptosystem with errorless decryption, i.e.,

$$\Pr\Big[(hk, xk) \leftarrow K_{\text{extract}}(1^k) : \forall (m, r) : \text{dec}_{xk}(\text{com}_{hk}(m; r)) = m\Big] \approx 1.$$

We demand that no non-uniform polynomial time adversary $\mathcal{A}$ can distinguish between the two key generation algorithms. This implies that the cryptosystem is semantically secure against chosen plaintext attack since the perfectly hiding commitment does not reveal what the message is.

We have already seen one example of a perfectly hiding commitment scheme with extraction. We can set up the BGN-cryptosystem with a public key, where h has full order n. In this case, the cryptosystem is a perfectly hiding commitment scheme. We can also set it up with h having order q, in this case, the cryptosystem has errorless decryption. The subgroup decisional assumption implies that no non-uniform polynomial time adversary can distinguish commitment keys from cryptosystem keys.

Pseudorandom cryptosystem. A cryptosystem $(K_{\text{pseudo}}, E, D)$ has pseudorandom ciphertexts of length $\ell_E(k)$ if for all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\begin{aligned} &\Pr\Big[(pk, dk) \leftarrow K_{\text{pseudo}}(1^k) : \mathcal{A}^{E_{pk}(\cdot)}(pk) = 1\Big] \\ \approx &\Pr\Big[(pk, dk) \leftarrow K_{\text{pseudo}}(1^k) : \mathcal{A}^{R_{pk}(\cdot)}(pk) = 1\Big], \end{aligned}$$

where $R_{pk}(m)$ runs $c \leftarrow \{0,1\}^{\ell_E(k)}$ and returns c. We require that the cryptosystem have errorless decryption as defined earlier.

Trapdoor permutations imply pseudorandom cryptosystems, we can use the Goldreich-Levin hard-core bit [21] of a trapdoor permutation to make a one-time pad. In the concrete case of the BGN cryptosystem, we observe that it implies hardness of factorization and it is possible to transform Rabin-encryption into a pseudorandom cryptosystem. When working over elliptic curves, there are also more direct constructions of pseudorandom cryptosystems based on the subgroup decision assumption.

TAG-BASED SIMULATION-SOUND TRAPDOOR COMMITMENT A tag-based commitment scheme has four algorithms. The key generation algorithm $K_{\mathrm{tag-com}}$ produces a commitment key ck as well as a trapdoor key tk. There is a commitment algorithm that takes as input the commitment key ck, a message m and any tag tag and outputs a commitment $c = \mathrm{commit}_{ck}(m, tag; r)$. To open a commitment c with tag tag we reveal m and the randomness r. Anybody can now verify whether indeed $c = \mathrm{commit}_{ck}(m, tag; r)$. As usual, the commitment scheme must be both hiding and binding.

In addition, to these two algorithms there are also a couple of trapdoor algorithms $\mathrm{Tcom}, \mathrm{Topen}$ that allow us to create an equivocal commitment and later open this commitment to any value we prefer. We create an equivocal commitment and an equivocation key as $(c, ek) \leftarrow \mathrm{Tcom}_{ck,tk}(tag)$. Later we can open it to any message m as $r \leftarrow \mathrm{Topen}_{ck,ek}(c, m, tag)$, such that $c = \mathrm{commit}_{ck}(m, tag; r)$. We require that equivocal commitments and openings are indistinguishable from real openings. For all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Pr\left[(ck, tk) \leftarrow K_{\mathrm{tag-com}}(1^k) : \mathcal{A}^{\mathcal{R}(\cdot,\cdot)}(ck) = 1\right]$$
$$\approx \Pr\left[(ck, tk) \leftarrow K_{\mathrm{tag-com}}(1^k) : \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(ck) = 1\right],$$

where $\mathcal{R}(m, tag)$ returns a randomly selected randomizer and $\mathcal{O}(m, tag)$ computes $(c, ek) \leftarrow \mathrm{Tcom}_{ck,tk}(m, tag); r \leftarrow \mathrm{Topen}_{ck,ek}(c, m, tag)$ and returns r and $\mathcal{A}$ does not submit the same tag twice to the oracle.

Tag-based simulation-sound trapdoor commitments were first implicitly constructed in [15], and explicitly in [16, 28]. The tag-based simulation soundness property is based on the notion of simulation soundness introduced by Sahai [32] for NIZK proofs. Aside from [15, 16, 28], other constructions of tag-based simulation sound commitments or schemes that can easily be transformed into tag-based simulation-sound commitments have appeared in [11, 8, 20, 10, 24, 25]. The tag-based simulation-soundness property means that a commitment using tag remains binding even if we have made equivocations for commitments using different tags. For all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Pr\Big[(ck, tk) \leftarrow K(1^k); (c, tag, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(ck) : tag \notin Q \text{ and}$$
$$c = \mathrm{commit}_{ck}(m_0, tag; r_0) = \mathrm{commit}_{ck}(m_1, tag; r_1) \text{ and } m_0 \neq m_1\Big] \approx 0,$$

where $\mathcal{O}(commit, tag)$ computes $(c, ek) \leftarrow \mathrm{Tcom}_{ck,tk}(tag)$, returns c and stores (c, tag, ek), and $\mathcal{O}(open, c, m, tag)$ returns $r \leftarrow \mathrm{Topen}_{ck,ek}(c, m, tag)$ if (c, tag, ek) has been stored, and where Q is the list of tags for which equivocal commitments have been made by $\mathcal{O}$.

Strong one-time signatures. We remind the reader that strong one-time signatures allow a non-uniform polynomial time adversary to ask an oracle for a signature on one arbitrary message. Then it must be infeasible to forge a signature on any different message and infeasible to come up with a different signature on the same message. Strong one-time signatures can be constructed from one-way functions.

6.3 UC NIZK

The standard technique to prove that a protocol securely realizes a functionality in the UC framework is to show that the ideal model adversary $\mathcal{S}$ can simulate everything that happens on top of the ideal functionality. In our case, there are two tricky parts. First, $\mathcal{S}$ may learn that a statement C has been proved and has to simulate a UC NIZK argument π without knowing the witness. Furthermore, if this honest prover is corrupted later then we learn the witness but must now simulate the randomness of the prover that would lead it to produce π. The second problem is that whenever $\mathcal{S}$ sees an acceptable UC NIZK argument π for a statement C, then an honest verifier V will accept. We must therefore, input a witness w to $\mathcal{F}_{\mathrm{NIZK}}$ so it can instruct V to accept.

The main idea in overcoming these hurdles is to commit to the witness w and make a NIZK proof that indeed we have committed to a witness w so $C(w) = 1$. If the NIZK proof has the honest prover state reconstruction property, then we can simulate NIZK proofs and the prover's random coins.

This leaves us with the commitment scheme. On one hand, when we simulate UC NIZK arguments we want to make equivocal commitments that can be opened to anything since we do not know the witness yet. On the other hand, when we see a UC NIZK argument that we did not construct ourselves we want to be able to extract the witness, since we have to give it to $\mathcal{F}_{\mathrm{NIZK}}$.

We will construct such a commitment scheme from the tools specified in the previous section. We use a tag-based simulation-sound trapdoor commitment scheme to commit to each bit of w. If w has length ℓ this gives us commitments $c_1, \ldots, c_\ell$. For honest provers we can use the trapdoor key tk to create equivocal commitments that can be opened to any bit we like. This enables us to simulate the commitments of the honest provers, and when we learn w upon corruption, we can simulate the randomness they could have used to commit to the witness w.

We still have an extraction problem, it is not clear that we can extract a witness from tag-based commitments created by a malicious adversary. To solve this problem we choose to encrypt the openings of the commitments. Now we can extract witnesses, but we have reintroduced the problem of equivocation. In a simulated commitment we may know two different openings of a commitment c_i to respectively 0 and 1, however, if we encrypt the opening then we are stuck with one possible opening. This is where the pseudorandomness property of the cryptosystem comes in handy. We can simply make two encryptions, one of an opening to 0 and one of an opening to 1. Since the ciphertexts are pseudorandom, we can open the ciphertext containing the opening we want and claim that the other ciphertext was chosen as a random string. To recap, the idea so far to commit to a bit b is to make a commitment c_i to this bit, and create a ciphertext $c_{i,b}$ containing an opening of c_i to b, while choosing $c_{i,1-b}$ as a random string.

The commitment scheme is equivocable, however, again we must be careful that we can extract a message from an adversarial commitment. The problem is that since we equivocate commitments for honest provers it may be the case that the adversary can produce equivocable commitments. This means, the adversary can produce some simulation sound commitment c_i and encryptions $c_{i,0}, c_{i,1}$ of openings to respectively 0 and 1. To resolve this issue we will select the tags for the commitments in a way so the adversary is forced to use a tag that has not been used to make an equivocable commitment. When an honest prover is making a commitment, we will select keys for a strong one-time signature scheme $(vk, sk) \leftarrow K_{\text{sign}}(1^k)$. We will use $tag = (vk, C)$ when making the commitment c_i. The verification key vk will be published together with the commitment, and we will sign the commitment (as well as something else) using this key. Since the adversary cannot forge signatures, it must use a different tag, and therefore the commitment is binding and only one of the ciphertexts can contain an opening of c_i. This allows us to establish simulation soundness.

If the adversary corrupts a party that has used vk earlier, then it may indeed sign messages using vk and can therefore use vk in the tag for commitments. However, since we also include the statement C in the tag for the commitment using vk, the adversary can only create an equivocable commitment in a UC NIZK argument for the same statement C. We will observe that in this particular case we do not need to extract the witness w, because we can get it during the corruption of the prover.

Finally, in order to make the UC NIZK argument perfect zero-knowledge we wrap all the commitments c_i and the ciphertexts $c_{i,b}$ inside a perfectly hiding commitment c. In the simulation, however, we generate the key for this commitment scheme in a way such that it is instead a cryptosystem and we can extract the plaintext. This last step is only added to make the UC NIZK argument perfect zero-knowledge, it can be omitted if perfect zero-knowledge is not needed.

The resulting protocol can be seen in Figure 4. We use the notation from Section 6.2.

We prove the following theorems in the full paper [26].

Theorem 4. *The protocol in Figure 6 securely realizes $\mathcal{F}_{\text{NIZK}}$ in the $\mathcal{F}_{CRS}$-hybrid model.*

Theorem 5. *The UC NIZK argument in Figure 4 is perfect zero-knowledge.*

Corollary 1. *Bilinear groups as described in Section 3 for which the decisional subgroup assumption holds imply the existence of a non-interactive perfect zero-knowledge protocol that securely realizes $\mathcal{F}_{\text{NIZK}}$.*

CRS generation:
1. $hk \leftarrow K_{\text{hiding}}(1^k)$
2. $(ck, tk) \leftarrow K_{\text{tag-com}}(1^k)$
3. $(pk, dk) \leftarrow K_{\text{pseudo}}(1^k)$
4. $(\sigma, \tau) \leftarrow S_1(1^k)$
5. Return $\Sigma = (hk, ck, pk, \sigma)$

Statement: A circuit C and a claim that there exists input wires w so $C(w) = 1$.

Proof: On input (Σ, C, w).
1. Check $C(w) = 1$ and return `failure` if not
2. $(vk, sk) \leftarrow K_{\text{sign}}(1^k)$
3. For $i = 1$ to ℓ select r_i at random and let $c_i = \text{commit}_{ck}(w_i, (vk, C); r_i)$
4. For $i = 1$ to ℓ select R_{w_i} at random and set $c_{i,w_i} = E_{pk}(r_i; R_{w_i})$ and choose $c_{i,1-w_i}$ as a random string.
5. Choose r at random and let $c = \text{com}_{hk}(c_1, c_{1,0}, c_{1,1}, \ldots, c_\ell, c_{\ell,0}, c_{\ell,1}; r)$
6. Create a NIZK argument π for the statement that there exists w such that $C(w) = 1$ and there exists randomness so c has been produced as described in steps 3,4 and 5.
7. $s \leftarrow \text{sign}_{sk}(C, vk, c, \pi)$
8. Return $\Pi = (vk, c, \pi, s)$

Verification: On input (Σ, C, Π)
1. Parse $\Pi = (vk, c, \pi, s)$
2. Verify that s is a signature on (C, vk, c, π) under vk.
3. Verify the NIZK argument π
4. Return 1 if all checks work out, else return 0

Fig. 4. UC NIZK argument

Common reference string: On input (**start**,sid) run $\Sigma \leftarrow K(1^k)$.
Send (**crs**,sid, Σ) to all parties and halt.

Fig. 5. Protocol for UC NIZK common reference string generation

Proof: Party P waits until receiving (**crs**,sid, Σ) from $\mathcal{F}_{CRS}$.
On input (**prove**,$sid, ssid, C, w$) run $\Pi \leftarrow P(\Sigma, C, w)$. Output (**proof**,$sid, ssid, \Pi$).

Verification: Party V waits until receiving (**crs**,sid, Σ) from $\mathcal{F}_{CRS}$.
On input (**verify**,$sid, ssid, C, \Pi$) run $b \leftarrow V(\Sigma, C, \Pi)$. Output (**verification**,$sid, ssid, b$).

Fig. 6. Protocol for UC NIZK argument

References

1. William Aiello and Johan Håstad. Perfect zero-knowledge languages can be recognized in two rounds. In *Proceedings of FOCS '87*, pages 439–448, 1987.
2. Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Jornal of Computation*, 20(6):1084–1118, 1991.
3. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *proceedings of STOC '88*, pages 103–112, 1988.

4. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *proceedings of TCC '05, LNCS series, volume 3378*, pages 325–341, 2005.
5. Gilles Brassard, David Chaum, and Claude Crèpeau. Minimum disclosure proofs of knowledge. *JCSS*, 37(2):156–189, 1988.
6. Gilles Brassard and Claude Crèpeau. Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for sat and beyond. In *Proceedings of FOCS '86*, pages 188–195, 1986.
7. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *proceedings of FOCS '01*, pages 136–145, 2001. Full paper available at `http://eprint.iacr.org/2000/067`.
8. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *proceedings of STOC '02*, pages 494–503, 2002. Full paper available at `http://eprint.iacr.org/2002/140`.
9. Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with proprocessing. In *proceedings of EUROCRYPT '92, LNCS series, volume 658*, pages 341–355, 1992.
10. Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *proceedings of STOC '03*, pages 426–437, 2003.
11. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 566–598, 2002.
12. Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Non-interactive zero-knowledge: A low-randomness characterization of np. In *proceedings of ICALP '99, LNCS series, volume 1644*, pages 271–280, 1999.
13. Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-optimal characterization of two np proof systems. In *proceedings of RANDOM '02, LNCS series, volume 2483*, pages 179–193, 2002.
14. Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. Image density is complete for non-interactive-szk. In *proceedings of ICALP '98, LNCS series, volume 1443*, pages 784–795, 1998.
15. Giovanni Di Crescenzo, Yvail Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *proceedings of STOC '98*, pages 141–150, 1998.
16. Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In *proceedings of EUROCRYPT '01*, pages 40–59, 2001.
17. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM J. of Computing*, 30(2):391–437, 2000. Earlier version at STOC '91.
18. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999. Earlier version entitled Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String appeared at FOCS '90.
19. Lance Fortnow. The complexity of perfect zero-knowledge. In *Proceedings of STOC '87*, pages 204–209, 1987.
20. Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 177–194, 2003. Full paper available at `http://eprint.iacr.org/2003/037`.
21. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *proceedings of STOC '89*, pages 25–32, 1989.

22. Oded Goldreich, Rafail Ostrovsky, and Erez Petrank. Computational complexity and knowledge complexity. *SIAM J. Comput.*, 27:1116–1141, 1998.
23. Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Can statistical zero knowledge be made non-interactive? or on the relationship of szk and niszk. In *CRYPTO '99, LNCS series, volume 1666*, pages 467–484, 1999.
24. Jens Groth. Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS, 2004. PhD thesis. xii+119 pp.
25. Jens Groth. Cryptography in subgroups of $\mathbb{Z}_n^*$. In *proceedings of TCC '05, LNCS series, volume 3378*, pages 50–65, 2005.
26. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero-knowledge for np. ECCC Report TR05-097, `http://eccc.uni-trier.de/eccc-reports/2005/TR05-097/index.html`, 2005.
27. Joe Kilian and Erez Petrank. An efficient noninteractive zero-knowledge proof system for np with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.
28. Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 382–400, 2004. Full paper available at `http://eprint.iacr.org/2003/252`.
29. Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Proceedings of Structure in Complexity Theory Conference*, pages 133–138, 1991.
30. Rafael Pass. On deniability in the common reference string and random oracle model. In *proceedings of CRYPTO '03, LNCS series, volume 2729*, pages 316–337, 2003.
31. Rafael Pass and Abhi Shelat. Characterizing non-interactive zero-knowledge in the public and secret parameter models. In *proceedings of CRYPTO '05, LNCS series*, 2005.
32. Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *proceedings of FOCS '01*, pages 543–553, 2001.
33. Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.