Perfect Non-Interactive Zero Knowledge for NP

Jens Groth* Rafail Ostrovsky† Amit Sahai‡

U.C.L.A.
Department of Computer Science
{jg,rafail,sahai}@cs.ucla.edu

August 31, 2005

Abstract

Non-interactive zero-knowledge (NIZK) 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 protocols for any NP language in the presence of a dynamic/adaptive adversary.

While it was already known how to construct efficient prover computational NIZK proofs for any NP language, the known techniques yield large common reference strings and large NIZK proofs. As an additional implication of our techniques, we considerably reduce both the size of the common reference string and the size of the proofs.

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

*Supported by NSF grant No. 0456717, and NSF Cybertrust grant.

†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 grant No. 0456717 from the NSF ITR and Cybertrust programs, an equipment grant from Intel, and an Alfred P. Sloan Foundation Research Fellowship.

knowledge (ZK), it has long been known how to construct statistical zero-knowledge argument systems for all NP languages [BCC88], for NIZK this question has remained open for nearly two decades.

IN CONTEXT WITH PREVIOUS WORK – STATISTICAL ZERO KNOWLEDGE: Blum, Feldman, and Micali [BFM88] 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 [BDMP91]. Both [BFM88] and [BDMP91] based their NIZK systems on certain number-theoretic assumptions (specifically, the hardness of deciding quadratic residues modulo a composite number). Feige, Lapidot, and Shamir [FLS90] 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 [BC86, BCC88]; 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 [BFM88] showed how an 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 non-trivial language for both proofs and arguments were shown to imply the existence of a one-way function by Ostrovsky [Ost91]. Statistical NIZK proof systems were further explored by De Santis, Di Crescenzo, Persiano, and Yung [DDPY98] and Goldreich, Sahai, and Vadhan [GSV99], 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 [For87, AH87][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 [BGN05].

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

- Perfect NIZK arguments for any NP language.
- UC-secure perfect NIZK arguments for any NP language, secure against adaptive/dynamic adversaries.

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 [BGN05]. To

[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 [Ost91] 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 [PS05] for an alternative proof.

[3]see also [GOP98] appendix regarding subtleties of this proof, and [SV03] for an alternative proof.

the best of our knowledge, our techniques are completely different from all previous constructions of NIZK proofs. 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. We point out that 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 [KP98] 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.

Reference	CRS size	Proof Size	Assumption
Kilian-Petrank	$O(\|C\|k^2)$	$O(\|C\|k^2)$	Trapdoor Permutations
Kilian-Petrank	$O(k^3)$	$O(\|C\|k^3)$	Trapdoor Permutations
This paper	$O(k)$	$O(\|C\|k)$	Specific Number-Theoretic [BGN05]

Table 1: Comparison of CRS size and NIZK Proof Size for Efficient-Prover NIZK Proof systems for NP-complete language

The NIZK proofs we construct are built using encryptions of the bits in the circuit. However, by a slight modification to only the reference string, we effectively transform the cryptosystem into a perfectly hiding commitment scheme. With this transformation, we obtain a perfect NIZK argument for NP statements. The result comes in two flavors:

- Perfect NIZK arguments for circuit satisfiability with "ordinary" soundness.
- Perfect NIZK arguments for circuit satisfiability with adaptive soundness, but for circuits of limited size.

By "ordinary" soundness we mean: for any NP statement, it is infeasible to make a valid NIZK argument for that statement given a random common reference string. However, in real life we can of course imagine an adversary that first sees the common reference string, and then chooses the false statement on which he will attempt to cheat. This is normally handled by an adaptive definition of soundness (e.g. [FLS90]). We make two observations regarding adaptive soundness:

First, we note that we can obtain full adaptive soundness if we restrict the size of statements to be proven. Let $\nu_{SD}(k)$ be the advantage of an adversary trying to decide the subgroup decision problem of [BGN05]. We can construct NIZK arguments with adaptive soundness by limiting the adversary to picking circuits of size $\ell(k)$ such that $\ell(k)^{\ell(k)}\nu_{SD}(k)$ is negligible[4]

Second, we observe that our construction of perfect NIZK arguments (with only "ordinary" soundness) already achieves a weaker, but sufficient, form of adaptive soundness. It turns out, informally speaking, that if an adversary succeeds in producing an NIZK argument for a false statement, it cannot "know" that it has done so. In other words, if the adversary can efficiently recognize when it has succeeded in specifying a false statement, then it *cannot* produce a valid proof of that statement.

We are able to formalize the second observation and illustrate its utility by constructing perfect NIZK arguments that satisfy Canetti's UC definition of security. Canetti introduced the universal composability (UC) framework [Can01] 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 and security when arbitrary protocols are executed concurrently. The notion of non-malleability was introduced by Dolev, Dwork and

[4]For instance, if $\ell(k) = k^\epsilon$, then we assume that $\nu_{SD}(k) = 2^{-\epsilon k^\epsilon \log k}\nu(k)$, where ν is negligible.

Naor [DDN00] in the interactive setting for Zero-Knowedlge and Commitment protocols. In the non-interactive setting, the first non-malleable commitment protocol was given by Di Crescenzo, Ishai and Ostrovsky [DIO98]. Sahai introduced the first non-malleable NIZK proof system, for a single theorem [Sah99]. De Santis, Di Crescenzo, Ostrovsky, Persiano and Sahai showed how to construct non-malleable NIZK proofs for polynomially-many theorems. As mentioned above, the UC framework guarantees a strong form of non-malleability, and in [CLOS02], it was observed that [DDO+01] achieves UC-security, but only for the setting with *static* advrsaries.

We define NIZK arguments in the UC framework and construct a NIZK argument (without any restrictions on the size of the NP statements that we prove) 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 security (including *soundness*!). At the same time, we can prove that our UC NIZK argument enjoys a perfect zero-knowledge property.

We stress that our result 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.

1.1 Notation

We model adversarial behavior as non-uniform interactive probabilistic polynomial time algorithms. Unless otherwise specified all other algorithms are uniform probabilistic polynomial time algorithms. A function $\nu : \mathbb{N} \to [0;1]$ is negligible if for all $\forall c > 0 \exists K \forall k > K : \nu(k) < \frac{1}{k^c}$. For two functions $f_1, f_2 : \mathbb{N} \to [0;1]$ we write $f_1(k) \approx f_2(k)$ if $|f_1(k) - f_2(k)|$ is negligible. We write $output \leftarrow A(input)$ for the process of selecting randomness r and setting $output = A(input; r)$.

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.

A proof 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 or a proof system for R if it has the completeness and soundness properties described below.

COMPLETENESS. For all adversaries $\mathcal{A}$ we have

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

SOUNDNESS. For all adversaries $\mathcal{A}$ we have

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

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

KNOWLEDGE EXTRACTION. We call (K, P, V) an argument of knowledge or a proof of knowledge for R if there exists a knowledge extractor $E = (E_1, E_2)$ with the properties described below.

For all adversaries $\mathcal{A}$ we have

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

For all adversaries $\mathcal{A}$ we have

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

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 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 security without erasures, the 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.

Remark. In the paper, we will construct protocols with perfect completeness, perfect soundness, perfect zero-knowledge, etc. In doing so we assume the ability to pick elements from certain sets, e.g., $r \leftarrow \mathbb{Z}_n^*$. If we consider the more strict setting, where the parties only have access to a source of unbiased coin-flips, we can still pick such elements from these sets in expected polynomial time. Alternatively, we can simply truncate the algorithms, in which case we do not get perfect completeness, perfect soundness, etc., but do get statistical completeness, statistical soundness, etc.

3 The Boneh-Goh-Nissim Cryptosystem

Boneh, Goh and Nissim [BGN05] 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.

[BGN05] 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 $\mathbb{G}$ and let $\mathcal{A}$ be an adversary. Define

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

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 adversary $\mathcal{A}$ we have SD-Adv$_{\mathcal{A}}(1^k) < \nu_{SD}(k)$ for sufficiently large k.*

We remark that we have changed the wording of the subgroup decision problem slightly in comparison with [BGN05], 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 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 if a ciphertext c contains 0 or 1, then either $c \in \mathbb{G}_q$ or $cg^{-1} \in \mathbb{G}_q$, so $e(c, cg^{-1})$ has order q. Write $c = g^y$, then $e(c, cg^{-1}) = e(g, g)^{y(y-1)}$. If $e(c, cg^{-1})$ has order q, then $y(y - 1) = 0 \bmod p$, so $y = 0 \bmod p$ or $y = 1 \bmod p$. Our strategy is 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 if $m = 1$ we have $e(c, cg^{-1}) = e(gh^w, h^w) = e(h, (gh^w)^w)$. So in both cases we get $e(c, cg^{-1}) = e(h, (g^{2m-1}h^w)^w)$. 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 $c \in \mathbb{G}, m \in \{0, 1\}$ 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

Figure 1: NIZK proof of plaintext being zero or one.

Theorem 2 *The protocol in Figure 1 is a NIZK proof that $c \in \mathbb{G}$ has plaintext $m \in \{0, 1\}$ with honest prover state reconstruction.*

Proof. PERFECT COMPLETENESS. Let x be the secret discrete logarithm so $h = g^x$. We know that $c = g^m h^w$, where $m \in \{0, 1\}$. This gives us $e(c, cg^{-1}) = e(g^{m+xw}, g^{m-1+xw}) =$

$e(g,g)^{m(m-1)+xw(2m-1+xw)} = e(g,g)^{rx(2m-1+xw)wr^{-1}} = e(h^r,(g^{2m-1}h^w)^{wr^{-1}}) = e(\pi_1,\pi_2)$. Furthermore, $e(\pi_1,g) = e(h^r,g) = e(h,g^r) = e(h,\pi_3)$.

PERFECT SOUNDNESS. Let again x be the secret discrete logarithm so $h = g^x$. Consider c,π so $e(c,cg^{-1}) = e(\pi_1,\pi_2)$ and $e(\pi_1,g) = e(h,\pi_3)$. There exist $0 \le m < p$ and $w \in \mathbb{Z}$ so $c = g^m h^w$.

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 there exists some r so $\pi_1 = h^r$.

As before we have $e(c,cg^{-1}) = e(g,g)^{m(m-1)+xw((2m-1)+xw)}$. At the same time we have $e(c,cg^{-1}) = e(\pi_1,\pi_2) = e(h^r,\pi_2)$ and therefore $e(c,cg^{-1})^q = e(h^{rq},\pi_2) = e(1,\pi_2) = 1$. So $m(m-1)+xw((2m-1)+xw) = 0 \bmod n$, and $p|x$ tells us $m(m-1) = 0 \bmod p$. Since $0 \le m < p$ this implies $m \in \{0,1\}$. So there does indeed exist $m \in \{0,1\}$ and w so $c = g^m h^w$.

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 $(\sigma,\tau) = ((n,\mathbb{G},\mathbb{G}_1,g,h),(p,q,\gamma))$.

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 adversaries $\mathcal{A}$ we have

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

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

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

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

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

Given ciphertexts c_0, c_1, c_2 containing plaintexts b_0, b_1, b_2 we can use the homomorphic properties 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 $b_2 = b_0$ NAND b_1, as required. We make such a NIZK proof for each NAND-gate in the circuit.

Theorem 4 *The protocol in Figure 2 is a NIZK proof of knowledge of circuit satisfiability with honest prover state reconstruction.*

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 3, 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 $g = h^\gamma$.

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 $c_{output} = g$, i.e., an easily verifiable encryption of 1.
5. For all NAND-gates, we do the following. We have input ciphertexts c_0, c_1 and output ciphertexts c_2. We wish to prove the existence of $w_0, w_1, w_2 \in \{0, 1\}$ and r_0, r_1, r_2 so $w_2 = w_0$ NAND w_1 and $c_j = g^{w_j} h^{r_j}$. To do so we make a NIZK proof that there exist m, r with $m \in \{0, 1\}$ so $c_0 c_1 c_2^2 g^{-2} = g^m h^r$.
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.

Figure 2: NIZK proof for circuit satisfiability.

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 2, we can argue that for all adversaries $\mathcal{A}$ we have

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

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 2 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 4 reveals that the argument is perfect zero-knowledge.

Soundness is trickier though. Since $g^m h^r$ is not statistically binding, we cannot prove soundness as we did in Theorem 4. 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 has negligible probability of making an acceptable NIZK argument.

However, if the common reference string is chosen first, then the adversary may choose a circuit C that depends on the common reference string. For instance, we cannot exclude the possibility that it could create an acceptable NIZK argument for h having order smaller than n. This is a false statement, since h has order n. However, if we try to argue soundness by switching the reference string to contain h with order q, then the statement is suddenly true and it might be possible to create such a NIZK argument.

In order to overcome this problem we tighten the subgroup decision assumption. We show that if all adversaries have less than $\ell(k)^{-\ell(k)}\nu(k)$ chance of distinguishing h generating either $\mathbb{G}$ or $\mathbb{G}_q$, then all adversaries have less than $\nu(k)$ chance of making an acceptable argument for an unsatisfiable circuit of size $\ell(k)$. This limits the size of the circuits for which we can prove soundness.

Let S_σ be the simulator S_1 from the proof of Theorem 4 restricted to its first output. We have the following theorem

Theorem 5 (S_σ, P, V) *is a NIZK argument for circuit satisfiability for circuits of size at most* $\ell(k)$ *if* $\nu_{SD}(k) < \ell(k)^{-\ell(k)}\nu(k)$ *for some negligible function* ν.

Proof. As in the proof of Theorem 4, we can show that the protocol has perfect completeness. Perfect zero-knowledge and honest prover state reconstruction follows from the proof of Theorem 4. This leaves us with the question of soundness.

NON-ADAPTIVE COMPUTATIONAL SOUNDNESS. We first demonstrate that the NIZK argument has non-adaptive soundness, i.e., all adversaries have negligible probability of proving a false statement if they choose this statement independently of the common reference string.

Consider any circuit C with no satisfying witness 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.

COMPUTATIONAL SOUNDNESS. Consider now an adversary $\mathcal{A}$ with probability So-Adv$_\mathcal{A}(1^k)$ for breaking the soundness property. Let C be the unsatisfiable circuit of size at most $\ell(k)$ that is most likely to be used by $\mathcal{A}$ in a valid NIZK argument. As argued in the previous paragraph, the probability of $\mathcal{A}$ selecting this circuit when it sees the reference string and making an acceptable NIZK argument is at most SD-Adv(1^k). There are at most $\ell(k)^{\ell(k)}$ circuits of size $\ell(k)$. Summing over all possible circuits we have So-Adv$_\mathcal{A}(1^k) \leq \ell(k)^{\ell(k)}\nu_{SD}(k) < \nu(k)$.

□

6 Universally Composable Non-interactive Zero-Knowledge

6.1 Modeling Non-interactive Zero-Knowledge Arguments

The universal composability (UC) framework (see [Can01] 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 an 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 [Can01] goes as follows: On input (**prove**,P, V, sid, x, w) from P the functionality $\mathcal{F}_{ZK}$ checks that $(x, w) \in R$ and in that case sends (**proof**,P, V, sid, 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 NIZK 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.

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. For this reason, and because the protocol and the security proof becomes a little simpler, we suggest a different functionality $\mathcal{F}_{NIZK}$ to capture the essence of NIZK arguments.

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

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

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

Figure 3: NIZK functionality $\mathcal{F}_{NIZK}$.

6.2 Tools

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

PERFECTLY HIDING COMMITMENT SCHEME WITH EXTRACTION. A perfectly hiding commitment scheme with extraction (first used in [CKOS01] in the setting of perfectly hiding non-malleable commitment) has the following property. We can run a key generation algorithm $hk \leftarrow Kstat(1^k)$ to get a hiding key hk, or we can alternatively run a key generation algorithm $(hk, xk) \leftarrow Kextract(1^k)$ in which case we get both a hiding key hk and an extraction key xk. $(Kstat, com)$ constitute a perfectly hiding commitment scheme. On the other hand, $(Kextract, com, dec)$ constitute a public key cryptosystem with errorless decryption, i.e.,

$$\Pr\left[(hk, xk) \leftarrow Kextract(1^k) : \forall(m, r) : dec_{xk}(\mathrm{com}_{hk}(m; r)) = m\right] \approx 1.$$

We demand that no 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 adversary can distinguish commitment keys from cryptosystem keys.

PSEUDORANDOM CRYPTOSYSTEM. A cryptosystem $(Kpseudo, E, D)$ has pseudorandom ciphertexts of length $\ell_E(k)$ if for all adversaries $\mathcal{A}$ we have

$$\begin{aligned} & \Pr\left[(pk, dk) \leftarrow Kpseudo(1^k) : \mathcal{A}^{E_{pk}(\cdot)}(pk) = 1\right] \\ \approx\ & \Pr\left[(pk, dk) \leftarrow Kpseudo(1^k) : \mathcal{A}^{R_{pk}(\cdot)}(pk) : \mathcal{A}(c) = 1\right], \end{aligned}$$

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.

The BGN-cryptosystem serves as an example of a pseudorandom cryptosystem. It is also known that trapdoor permutations imply pseudorandom cryptosystems, we can use the Goldreich-Levin hard-core bit [GL89] of a trapdoor permutation to make a one-time pad.

TAG-BASED SIMULATION-SOUND TRAPDOOR COMMITMENT A tag-based commitment scheme has four algorithms. The key generation algorithm $Kcom$ 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 we reveal m, tag 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 adversaries $\mathcal{A}$ we have

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

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.

The tag-based simulation soundness property is based on the notion of simulation soundness introduced by Sahai [Sah99] for NIZK proofs. It means that a commitment using tag remains binding even if we have made equivocations for commitments using different tags. For all adversaries $\mathcal{A}$ we have

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

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 $\mathcal{O}$.

Tag-based simulation-sound trapdoor commitment were first implicitly defined in [DIO98], and explicitly in [CKOS01, MY04]. The notion of simulation soundness for NIZK [Sah99] will be critical to us here, as well (see below). Aside from [DIO98, Sah99, CKOS01, MY04], other constructions of tag-based simulation sound commitments or schemes that can easily be transformed into tag-based simulation-sound commitments have appeared in [DDO+01, CLOS02, GMY03, DG03, Gro04, Gro05].

STRONG ONE-TIME SIGNATURES. We remind the reader that strong one-time signatures allow an 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 also infeasible to come up with a different signature on the same message. 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 $\mathcal{F}_{NIZK}$ 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 $C(w) = 1$. We must show that our NIZK proof has a simulation-soundness property (see above) to ensure that only true statements can be proven. On the other hand, if the NIZK proof has the honest prover state reconstruction property, then we can simulate NIZK proofs and the prover's random coins when forming this NIZK proof. 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 $\mathcal{F}_{NIZK}$.

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 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 $(vk, sk) \leftarrow Ksign(1^k)$. 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. We note that this 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.

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

Proof. Let $\mathcal{A}$ be any adversary. We will describe an ideal adversary $\mathcal{S}$ so no environment can distinguish whether it is running in the $\mathcal{F}_{CRS}$-hybrid model with parties $P_1, \ldots, P_n$ and adversary $\mathcal{A}$ or in the ideal process with $\mathcal{F}_{NIZK}$, $\mathcal{S}$ and dummy parties $\tilde{P}_1, \ldots, \tilde{P}_n$.

$\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$. It will run a simulated interaction of $\mathcal{A}$, the parties and the environment. In particular, whenever the simulated $\mathcal{A}$ communicates with the environment, $\mathcal{S}$ just passes this information along. And whenever $\mathcal{A}$ corrupts a party P_i, $\mathcal{S}$ corrupts the corresponding dummy party $\tilde{P}_i$.

SIMULATING $\mathcal{F}_{CRS}$. $\mathcal{S}$ chooses the common reference string in the following way. It selects, $(hk, xk) \leftarrow Kextract(1^k); (ck, tk) \leftarrow Kcom(1^k); (pk, dk) \leftarrow Kpseudo(1^k)$ and $(\sigma, \tau) \leftarrow S_1(1^k)$. This means $\mathcal{S}$ is capable of extracting plaintext committed under hk, able to create and equivocate simulation sound trapdoor commitments, decrypt pseudorandom ciphertexts and simulate NIZK proofs and make honest prover state reconstruction of NIZK proofs.

CRS generation:

1. $hk \leftarrow Kstat(1^k)$
2. $(ck, tk) \leftarrow Kcom(1^k)$
3. $(pk, dk) \leftarrow Kpseudo(1^k)$
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. $(vk, sk) \leftarrow Ksign(1^k)$
3. For $i = 1$ to ℓ select r_i at random and let $c_i = \mathrm{commit}_{ck}(w_i, (vk, C); r_i)$
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 $c = \mathrm{com}_{hk}(c_1, c_{1,0}, c_{1,1}, \ldots, c_\ell, c_{\ell,0}, c_{\ell,1}; r)$
6. Create a NIZK proof 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. $s \leftarrow \mathrm{sign}_{sk}(C, vk, c, \pi)$
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 proof π
4. Return 1 if all checks work out, else return 0

Figure 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.

Figure 5: Protocol for UC NIZK common reference string generation.

Let $\Sigma = (hk, ck, pk, \sigma)$. $\mathcal{S}$ simulates $\mathcal{F}_{CRS}$ sending (**crs**,sid, Σ) to all parties. Whenever $\mathcal{A}$ decides to deliver such a message to a party P_i, $\mathcal{S}$ will simulate P_i receiving this string.

SIMULATING UNCORRUPTED PROVERS. Suppose $\mathcal{S}$ receives (**proof**,$sid, ssid, C$) from $\mathcal{F}_{NIZK}$. This means that some dummy party $\tilde{P}$ received input (**prove**,$sid, ssid, C, w$), where $C(w) = 1$. We must simulate the output a real party P would make, however, we may not know w.

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**,$sid, ssid, b$).

Figure 6: Protocol for UC NIZK argument.

We create $(vk, sk) \leftarrow Ksign(1^k)$. Let $tag = (vk, C)$ and form equivocal commitments $(c_i, ek) \leftarrow \mathrm{Tcom}_{pk,tk}(tag)$. We simulate openings of the c_i's to both 0 and 1. For all $i = 1$ to ℓ and $b = 0$ to 1 compute $\rho_{i,b} \leftarrow \mathrm{Topen}_{ck,ek}(c_i, b, tag)$. Select $r_{i,b}$ at random and set $c_{i,b} = E_{pk}(\rho_{i,b}; r_{i,b})$. Compute $c = E_{hk}(c_1, c_{1,0}, c_{1,1}, \ldots, c_\ell, c_{\ell,0}, c_{\ell,1}; r)$ for a random r. Choose randomness ρ and simulate the NIZK proof as $\pi \leftarrow S_2(\sigma, \tau, (C, vk, c); \rho)$. Finally, create a one-time signature s on C, vk, c, π.

Let $\Pi = (vk, c, \pi, s)$ and return (**proof**,Π) to $\mathcal{F}_{NIZK}$. $\mathcal{F}_{NIZK}$ subsequently sends (**proof**,$sid, ssid, \Pi$) to $\tilde{P}$ and we deliver this message so it gets output to the environment.

SIMULATING UNCORRUPTED VERIFIERS. Suppose $\mathcal{S}$ receives (**verify**,C, Π) from $\mathcal{F}_{NIZK}$. This means an honest dummy party $\tilde{V}$ has received (**verify**,$sid, ssid, C, \Pi$) from the environment.

$\mathcal{S}$ checks the UC NIZK argument, $b \leftarrow V(\Sigma, C, \Pi)$. If invalid, it sends (**witness**,`no witness`) to $\mathcal{F}_{NIZK}$ and delivers the consequent message (**verification**,$sid, ssid, 0$) to $\tilde{V}$ that outputs this rejection to the environment.

On the other hand, if the UC NIZK argument is valid we must try to extract a witness w. If C has ever been proved by an honest prover that was later corrupted, we will know the witness and do not need to run the following extraction procedure. If the witness is not known already $\mathcal{S}$ uses the extraction key xk to extract a plaintext $c_1, c_{1,0}, c_{1,1}, \ldots, c_\ell, c_{\ell,0}, c_{\ell,1}$ from c. Since it knows the decryption key dk, it can then decrypt all $c_{i,b}$. This gives us plaintexts $\rho_{i,b}$. We check whether $c_i = \mathrm{Tcom}_{ck}(b, (vk, C); \rho_{i,b})$ and in that case b is a possible candidate for the i-th bit of w.

If successful in all of this, $\mathcal{S}$ lets w be these bits. However, if any of the bits are ambiguous, i.e., w_i could be both 0 and 1, or if any of them are inextractable, then it sets $w =$ `no witness`. It sends (**witness**,w) to $\mathcal{F}_{\mathcal{NIZK}}$. It delivers the resulting output message to $\tilde{V}$ that outputs it to the environment.

We will later argue that the probability of the UC NIZK argument being valid, yet not being able to supply a good witness to $\mathcal{F}_{NIZK}$ is negligible. That means with overwhelming probability we input a valid witness w to $\mathcal{F}_{NIZK}$ when Π is an acceptable UC NIZK argument for satisfiability of C.

SIMULATING CORRUPTION. Suppose a simulated party P_i is corrupted by $\mathcal{A}$. Then we have to simulate the transcript of P_i. We start by corrupting $\tilde{P}_i$ thereby learning all UC NIZK arguments it has verified. It is straightforward to simulate P_i's internal tapes when running these verification processes.

We also learn all statements C that it has proved together with the corresponding witnesses w. Recall, the UC NIZK arguments Π have been provided by $\mathcal{S}$. Here is how we can simulate the randomness that would lead P_i to produce such a UC NIZK argument Π. Since $\mathcal{S}$ created $c_i, c_{i,0}, c_{i,1}$ such that $c_{i,0}$ contains a 0-opening of c_i and $c_{i,1}$ contains a 1-opening of c_i it can produce good looking randomness to claim that it committed to w_i. This also gives us convincing randomness for constructing all these commitments and for producing the ciphertext c, so we can run the honest prover state reconstruction algorithm S_3 to simulate randomness that would lead the prover to produce π.

HYBRIDS. We wish to argue that no environment can distinguish between the adversary $\mathcal{A}$ running with parties executing the UC NIZK protocol in the $\mathcal{F}_{CRS}$-hybrid model and the ideal adversary $\mathcal{S}$ running in the $\mathcal{F}_{NIZK}$-hybrid model with dummy parties. In order to do so we define several hybrid experiments and show that the environment cannot distinguish between any of them.

H0: This is the $\mathcal{F}_{CRS}$-hybrid model running with adversary $\mathcal{A}$ and parties $P_1, \ldots, P_n$.

H1: We modify H0 by running $(hk, xk) \leftarrow Kextract(1^k)$ instead of $hk \leftarrow Kstat(1^k)$ when generating the common reference string Σ.

H0 and H1 are indistinguishable, because otherwise we could build a distinguisher that could tell which key generation algorithm created hk.

H2: We modify H1 in the way an uncorrupted prover P creates commitments $c_1, \ldots, c_\ell$. Let $tag = (vk, C)$ as chosen in the proof. Instead of creating c_i by selecting r_i at random and setting $c_i = \text{commit}_{ck}(w_i, tag; r_i)$, we create an equivocal commitment $(c_i, ek) \leftarrow \text{Tcom}_{ck,tk}(tag)$ and subsequently produce randomness $\rho_{i,w_i} \leftarrow \text{Topen}_{ck,ek}(c_i, w_i, tag)$. We continue the proof using ρ_{i,w_i} instead of r_i.

H1 and H2 are indistinguishable. If they were distinguishable, then we could distinguish real commitments and openings from equivocal commitments and equivocated openings, in violation of the definition of trapdoor commitments.

H3: In H3, we make another modification to the procedure followed by an honest prover. We are already creating c_i as an equivocal commitment and equivocating it with randomness ρ_{i,w_i} that would open it to contain w_i. We run the equivocation procedure once more to also create convincing randomness that would explain c_i as a commitment to $1 - w_i$. This means, we compute $\rho_{i,1-w_i} \leftarrow \text{Topen}_{ck,ek}(c_i, 1 - w_i, tag)$. Instead of selecting $c_{i,1-w_i}$ as a random string, we choose to encrypt $\rho_{i,1-w_i}$ as $c_{i,1-w_i} = E_{pk}(\rho_{i,1-w_i}; r_{i,1-w_i})$ for a randomly chosen $r_{i,1-w_i}$. We still pretend that $c_{i,1-w_i}$ is a randomly chosen string when we carry out the NIZK proof π or if the prover is ever corrupted.

H2 and H3 are indistinguishable because of the pseudorandomness property of the cryptosystem. Suppose we could distinguish H2 and H3, then we can distinguish between an encryption oracle and an oracle that supplies randomly chosen strings.

H4: Consider the case where an honest party V receives (**verify**,$sid, ssid, C, \Pi$). Suppose Π is indeed an acceptable UC NIZK argument and the one-time signature scheme has verification key vk. If vk was selected by an honest party in making a UC NIZK argument, this party is still uncorrupted, yet C, Π differ from the UC NIZK argument this honest party produced, then we output `failure` to the environment.

To argue that H3 and H4 are indistinguishable we need to show that the probability of failure is negligible. This follows from the fact that outputting `failure` corresponds to a forgery of a strong one-time signature.

H5: Again, we look at the case of an uncorrupted verifier that has an acceptable UC NIZK argument C, Π to verify. If C, Π were produced by an uncorrupted prover we do not change the protocol, neither do we modify the protocol if C has been proved by an honest prover that has later been corrupted. In all other cases, we use the extraction key xk in an attempt to decrypt c to get a plaintext on the form $c_1, c_{1,0}, c_{1,1}, \ldots, c_\ell, c_{\ell,0}, c_{\ell,1}$. Then we use the decryption key dk to attempt to decrypt the $c_{i,b}$'s to get $\rho_{i,b}$ so $c_{i,b} = \text{commit}_{ck}(b, (vk, C); \rho_{i,b}$. We output `failure` if at any point we encounter a $c_i = \text{commit}_{pk}(0, (vk, C), \rho_{i,0}) = \text{commit}_{ck}(1, (vk, C), \rho_{i,1})$.

Simulation soundness of the commitment scheme implies that H4 and H5 are indistinguishable. Consider the tag (vk, C). Outputting `failure` corresponds to breaking the binding property of the commitment scheme, unless we have previously equivocated a commitment with tag (vk, C). In H4, we ruled out the possibility of vk coming from a UC NIZK argument of a party that is still uncorrupted. This leaves us with the possibility of $\mathcal{A}$ corrupting an honest prover P, learning the secret key sk corresponding to vk and making a UC NIZK argument using this. However, this means that C stems from the same honest prover that has now been corrupted, and in that case we do not try to extract $\rho_{i,b}$'s.

H6: We modify the common reference string by selecting $\sigma \leftarrow K(1^k)$ instead of $(\sigma, \tau) \leftarrow S_1(1^k)$.

Since we do not use τ for anything at the moment, the zero-knowledge property implies that H5 and H6 are indistinguishable.

H7: As in H5, we try to extract $\rho_{i,0}, \rho_{i,1}$'s. We output `failure` if we cannot decrypt c to get $c_1, c_{1,0}, c_{1,1}, \ldots, c_\ell, c_{\ell,0}, c_{\ell,1}$. We also output `failure` if there is an i so we cannot decrypt either $c_{i,0}$ or $c_{i,1}$ to give us $\rho_{i,b}$ so $c_i = \text{commit}_{ck}(b, (vk, C); \rho_{i,b})$. We ruled out the possibility of both $\rho_{i,0}$ and $\rho_{i,1}$ being an opening of c_i in H5, so if everything is OK so far we have a uniquely defined w so for all i we have $c_i = \text{commit}_{ck}(w_i, (vk, C); \rho_{i,w_i})$. We output `failure` if $C(w) \neq 1$.

¿From the soundness property of the NIZK proof and the errorless decryption property of the cryptosystems we know that we do succeed in decrypting c to some $c_1, c_{1,0}, c_{1,1}, \ldots, c_\ell, c_{\ell,0}, c_{\ell,1}$. The NIZK proof also tells us that for all $i = 1$ to ℓ at least one of the $c_{i,0}, c_{i,1}$ will have a proper $\rho_{i,b}$ so $c_i = \text{commit}_{pk}(b, (vk, C); \rho_{i,b})$. By the soundness property of the NIZK proof we have $C(w) = 1$. The probability of outputting `failure` is therefore negligible, and H6 is indistinguishable from H7.

H8: Instead of making real NIZK proofs for uncorrupted provers we use the honest prover state reconstruction simulators. In other words, we run $(\sigma, \tau) \leftarrow S_1(1^k)$ when we create the common reference string. We use $\pi \leftarrow S_2(\sigma, \tau, \cdot; \rho)$ with ρ random to simulate the honest provers NIZK proofs that c has been correctly generated. Finally, if any such prover is corrupted we use $r \leftarrow S_3(\sigma, \tau, x, \pi, \cdot, \rho)$ to create convincing randomness that would make the prover output π on the witness for c being correctly generated.

The honest prover state reconstruction property of the NIZK proof implies that H7 and H8 are indistinguishable.

SIM: This is the ideal process running with $\mathcal{F}_{NIZK}$ and $\mathcal{S}$.

H8 is already very similar to the ideal process. Honest provers in H8 make UC NIZK arguments in the same way as $\mathcal{S}$ without using the knowledge of the witness w for anything. It therefore makes no difference that $\mathcal{S}$ only learns w upon corruption of a party P when it has to simulate the random tape of said party.

Whenever an honest verifier has to verify a proof C, Π we are also very close to what happens in the simulation. If C, Π has been produced by an honest prover it returns 1, as will the dummy verifier in the ideal process. If C is a statement proved by an honest prover, but this prover has later been corrupted, then in H8 the verifier will return 1 if Π is an acceptable UC NIZK argument. $\mathcal{S}$ in a similar situation will have corrupted the dummy prover that made the UC NIZK argument, and therefore it will know the witness. If Π is an acceptable UC NIZK argument, it can therefore give this witness to $\mathcal{F}_{NIZK}$ that will make the dummy verifier output an acceptance to the environment. Finally, in the remaining case we have argued in H7 that we manage to extract a witness w if Π is acceptable and this extraction procedure is carried out exactly as it is done by $\mathcal{S}$. Therefore, $\mathcal{S}$ can submit this witness to $\mathcal{F}_{NIZK}$.

In conclusion, H8 is perfectly indistinguishable from the ideal process. Our path from H0 to SIM shows us that H0 and SIM are indistinguishable.

□

Theorem 7 *The UC NIZK argument in Figure 4 is perfect zero-knowledge.*

Proof. We start by describing the simulator $S^{UC} = (S_1^{UC}, S_2^{UC})$. S_1^{UC} runs $hk \leftarrow Kstat(1^k); (ck, tk) \leftarrow Kcom(1^k); (pk, sk) \leftarrow Kpseudo(1^k); (\sigma, \tau) \leftarrow K(1^k)$. Let $\Sigma = (hk, ck, pk, \sigma)$. S_1^{UC} outputs (Σ, τ).

Consider next S_2 that is given a circuit C on which to simulate a UC NIZK argument Π for satisfiability. It generates keys for the one-time signature scheme $(vk, sk) \leftarrow Ksign(1^k)$. Then generates a statistically hiding commitment $c \leftarrow \mathrm{com}_{hk}(0)$. It simulates a proof π for the statement x that c has been correctly formed and contains a witness w so $C(w) = 1$ as $\pi \leftarrow S_2(\Sigma, \tau, x)$. Finally, it creates a one-time signature on everything, $s \leftarrow \mathrm{sign}_{sk}(C, vk, c, \pi)$. It outputs the simulated UC NIZK argument $\Pi = (vk, c, \pi, s)$.

Perfect zero-knowledge of the NIZK proof implies that for all adversaries $\mathcal{A}$ we have

$$\Pr\left[\Sigma \leftarrow K^{UC}(1^k) : \mathcal{A}^{P(\Sigma,\cdot,\cdot)}(\Sigma) = 1\right] = \Pr\left[(\Sigma, \tau) \leftarrow K^{UC}(1^k) : \mathcal{A}^{PS(\Sigma,\tau,\cdot,\cdot)}(\Sigma) = 1\right],$$

where PS is an oracle that on input (Σ, τ, C, w) outputs `failure` if $C(w) = 0$ and otherwise creates a UC NIZK argument $\Pi = (vk, c, \pi, s)$ by following the provers algorithm for creating vk, c, s but simulating the NIZK proof π.

Next, we argue that for all adversaries $\mathcal{A}$ we have

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

where $S'(\Sigma, \tau, C, w)$ checks that $C(w) = 1$ and in that case returns $\Pi \leftarrow S_2(\Sigma, \tau, C)$.

The only difference in the two oracles PS and S' is the message inside the commitment c. However, since the commitment scheme is perfectly hiding, this does not change the distributions.

□

Corollary 8 *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 $\mathcal{F}_{NIZK}$.*

Proof. The assumption implies the existence of strong one-time signatures since one-way functions suffice for constructing those. The existence of one-way functions also suffices for the construction of tag-based simulation sound trapdoor commitments. The BGN-cryptosystem can be set up both as a cryptosystem and as a perfectly hiding commitment scheme, and the subgroup decision assumption says that the two types of keys cannot be distinguished. The BGN-cryptosystem has pseudorandom ciphertexts, since we can sample random element from $\mathbb{G}$ and cannot distinguish a full order h from a small order h. Finally, as we saw in Section 4.2 we can construct a NIZK proof with honest prover state reconstruction from the subgroup decision assumption. According to Theorem 6, plugging all these parts into the UC NIZK argument construction in 4 gives us a protocol that securely realizes $\mathcal{F}_{NIZK}$.

As already mentioned the BGN-cryptosystem set up with a full order h is perfectly hiding. It follows from the proof of Theorem 4 that using the NIZK proof with a simulated common reference string σ gives us a perfect zero-knowledge argument. Theorem 7 then tells us that the UC NIZK argument is perfect zero-knowledge.

□

References

[AH87] William Aiello and Johan Håstad. Perfect zero-knowledge languages can be recognized in two rounds. In *Proceedings of FOCS '87*, pages 439–448, 1987.

[BC86] 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.

[BCC88] Gilles Brassard, David Chaum, and Claude Crèpeau. Minimum disclosure proofs of knowledge. *JCSS*, 37(2):156–189, 1988.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Jornal of Computation*, 20(6):1084–1118, 1991.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *proceedings of STOC '88*, pages 103–112, 1988.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *proceedings of TCC '05*, pages 325–341, 2005.

[Can01] 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`.

[CKOS01] 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.

[CLOS02] 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`.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM J. of Computing*, 30(2):391–437, 2000. Earlier version at STOC '91.

[DDO+01] 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, 2001.

[DDPY98] 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.

[DG03] Ivan Damgård and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *proceedings of STOC '03*, pages 426–437, 2003.

[DIO98] Giovanni Di Crescenzo, Yvail Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *proceedings of STOC '98)*, pages 141–150, 1998.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string. In *proceedings of FOCS '90*, pages 308–317, 1990.

[For87] Lance Fortnow. The complexity of perfect zero-knowledge. In *Proceedings of STOC '87*, pages 204–209, 1987.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *proceedings of STOC '89*, pages 25–32, 1989.

[GMY03] 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`.

[GOP98] Oded Goldreich, Rafail Ostrovsky, and Erez Petrank. Computational complexity and knowledge complexity. *SIAM J. Comput.*, 27:1116–1141, 1998.

[Gro04] Jens Groth. Honest verifier zero-knowledge arguments applied. Dissertation Series DS-04-3, BRICS, 2004. PhD thesis. xii+119 pp.

[Gro05] Jens Groth. Cryptography in subgroups of $\mathbb{Z}_n^*$. In *proceedings of TCC '05, LNCS series, volume 3378*, pages 50–65, 2005.

[GSV99] 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.

[KP98] 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.

[MY04] 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`.

[Ost91] 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.

[PS05] 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.

[Sah99] Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *proceedings of FOCS '99*, pages 543–553, 1999.

[SV03] Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM (JACM)*, 50(2):196–249, 2003.