Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures[*]

Jens Groth[†]

UCLA, Computer Science Department
3531A Boelter Hall
Los Angeles, CA 90095, USA
jg@cs.ucla.edu

December 10, 2006

Abstract

Non-interactive zero-knowledge proofs play an essential role in many cryptographic protocols. We suggest several NIZK proof systems based on prime order groups with a bilinear map. We obtain linear size proofs for relations among group elements without going through an expensive reduction to an NP-complete language such as Circuit Satisfiability. Security of all our constructions is based on the decisional linear assumption.

The NIZK proof system is quite general and has many applications such as digital signatures, verifiable encryption and group signatures. We focus on the latter and get the first group signature scheme satisfying the strong security definition of Bellare, Shi and Zhang [BSZ05] in the standard model without random oracles where each group signature consists only of a constant number of group elements.

We also suggest a simulation-sound NIZK proof of knowledge, which is much more efficient than previous constructions in the literature.

Caveat: The constants are large, and therefore our schemes are not practical. Nonetheless, we find it very interesting for the first time to have NIZK proofs and group signatures that except for a constant factor are optimal without using the random oracle model to argue security.

Keywords: Non-interactive zero-knowledge, simulation-sound extractability, group signatures, decisional linear assumption.

[*]An extended abstract appears at Asiacrypt 2006. This is the full paper.

[†]Supported by NSF grant No. 0456717, and NSF Cybertrust grant.

1 Introduction

A non-interactive proof system allows a prover to convince a verifier about the truth of a statement. Zero-knowledge captures the notion that the verifier learns no more from the proof than the truth of the statement. We refer to Section 2 for formal definitions of non-interactive zero-knowledge (NIZK) proofs. NIZK proofs play a central role in the field of cryptography. Our goal in this paper is to construct short efficient prover NIZK proofs for languages that come up in practice when constructing cryptographic protocols. As an example of the usefulness of these new techniques, we construct group signatures consisting of a constant number of group elements.

1.1 Setup

We use two cyclic groups $\mathbb{G}, \mathbb{G}_1$ of order p, where p is a prime. 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, and the bilinear map be efficiently computable. Such groups have been widely used in cryptography in recent years.

Let $\mathcal{G}$ be an algorithm that takes a security parameter as input and outputs $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ such that p is prime, $\mathbb{G}, \mathbb{G}_1$ are descriptions of groups of order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is an admissible bilinear map as described above and g is a random generator of $\mathbb{G}$.

We use the decisional linear assumption introduced by Boneh, Boyen and Shacham [BBS04].

Definition 1 (Decisional Linear Assumption (DLIN)) *We say the decisional linear assumption holds for the bilinear group generator $\mathcal{G}$ if for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\begin{aligned} &\Pr\left[(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k); x, y, r, s \leftarrow \mathbb{Z}_p : \mathcal{A}(p, \mathbb{G}, \mathbb{G}_1, e, g, g^x, g^y, g^{xr}, g^{ys}, g^{r+s}) = 1\right] \\ \approx\ &\Pr\left[(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k); x, y, r, s, d \leftarrow \mathbb{Z}_p : \mathcal{A}(p, \mathbb{G}, \mathbb{G}_1, e, g, g^x, g^y, g^{xr}, g^{ys}, g^d) = 1\right]. \end{aligned}$$

Throughout the paper, we work over a bilinear group $(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k)$ generated such that the DLIN assumption holds for $\mathcal{G}$. We call this a DLIN group. Honest parties always check group membership of $\mathbb{G}, \mathbb{G}_1$ when relevant and halt if an element does not belong to a group that it was supposed to according to the protocol.

Given a DLIN group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ we can set up a semantically secure cryptosystem as in [BBS04]. We choose at random $x, y \leftarrow \mathbb{Z}_p^*$. The public key is (f, h), where $f = g^x, h = g^y$, and the secret key is (x, y). To encrypt a message $m \in \mathbb{G}$ we choose $r, s \leftarrow \mathbb{Z}_p$ and let the ciphertext be $(u, v, w) = (f^r, h^s, g^{r+s}m)$. To decrypt a ciphertext $(u, v, w) \in \mathbb{G}^3$ we compute $m = D(u, v, w) = u^{-1/x}v^{-1/y}w$.

The cryptosystem (K_{cpa}, E, D) has several nice properties. The DLIN assumption for $\mathcal{G}$ implies semantic security under chosen plaintext attack (CPA). All triples $(u, v, w) \in \mathbb{G}^3$ are valid ciphertexts. Also, the cryptosystem is homomorphic in the sense that

$$E(m_1; r_1, s_1)E(m_2, r_2, s_2) = E(m_1 m_2; r_1 + r_2, s_1 + s_2).$$

Given a group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ we define a pairing product equation of length ℓ over variables $a_1, \ldots, a_n$ to be an equation of the following form.

$$\prod_{j=1}^{\ell} e(q_{j,0}, q_{j,1}) = 1, \text{ where } q_{j,b} = b_{j,b} \prod_{i=1}^{n} a_i^{e_{j,b,i}} \text{ with } b_{j,b} \in \mathbb{G}\ ,\ e_{j,b,i} \in \mathbb{Z}_p.$$

Given a set S of pairing product equations $\text{eq}_1, \ldots, \text{eq}_m$ we can ask the natural question: *Is there a tuple $(a_1, \ldots, a_n) \in \mathbb{G}^n$ such that all equations in S are simultaneously satisfied?*

To illustrate the generality of the language of satisfiable pairing product equations we observe a reduction from the NP-complete language Circuit Satisfiability. Let $a_1, \dots, a_n$ correspond to the wires of the circuit, which without loss of generality contains only NAND-gates. Let S contain equations $e(a_i, a_i g^{-1}) = 1$ forcing each $a_i = g^{b_i}$ to encode a bit $b_i \in \{0, 1\}$. For each NAND-gate with input wires i_0, i_1 and output i_2 add to S the equation $e(a_{i_0}, a_{i_1}) = e(g, g a_{i_2}^{-1})$, which is satisfied if and only if $b_{i_2} = \neg(b_{i_0} \wedge b_{i_1})$.

Our main motivation for being interested in satisfiability of pairing product equations is not NP-completeness though. Satisfiability of pairing product equations comes up in practice when constructing cryptographic protocols and by making a direct NIZK proof instead of first reducing the problem to some other language such as Circuit Satisfiability we keep proofs short.

For concreteness, let us use verifiable encryption as an example of a pairing product satisfiability question that may come up in practice. Suppose (u, v, w) is a ciphertext under the public key (f, h) of the DLIN-based cryptosystem described earlier. We are interested in whether this ciphertext encrypts a particular message m. This is the case, if and only if there exists a such that $e(g, u) = e(a, f)$ and $e(h, wm^{-1}a^{-1})) = e(v, g)$. If we know r, s we can compute the satisfiability witness $a = g^r$.

1.2 NIZK Proofs for Satisfiability of Pairing Product Equations

NIZK PROOFS. The central technical contribution of this paper is an NIZK proof of size $\mathcal{O}(n + \ell)$ group elements for satisfiability of a set of pairing product equations of combined length $\ell = \sum_{j=1}^{m} \ell_j$. The proof system has perfect completeness and perfect soundness.

RELATED WORK ON NIZK PROOFS. NIZK proofs were introduced by Blum, Feldman and Micali [BFM88] and they suggested an NIZK proof for a single statement based on the hardness of deciding quadratic residousity. Blum et al. [BDMP91] extended this to multi-theorem NIZK proofs. Feige, Lapidot and Shamir [FLS99] and Kilian and Petrank [KP98] give constructions based on trapdoor permutations.

Recently Groth, Ostrovsky and Sahai [GOS06b] have constructed NIZK proofs from composite order bilinear groups introduced by Boneh, Goh and Nissim [BGN05]. Even more recently Groth, Ostrovsky and Sahai [GOS06a] have introduced the setting in this paper, a bilinear group of prime order and the DLIN assumption. They construct non-interactive witness-indistinguishable proofs without any setup assumptions. In the common reference string (CRS) model both results give NIZK proofs for Circuit Satisfiability of size $\mathcal{O}(|C|)$ group elements.

All the above-mentioned papers have in common that they focus on an NP-complete language, usually Circuit Satisfiability, and suggest a bit-by-bit or gate-by-gate NIZK proof for this language. Our paper differs by introducing new techniques that allows making *direct* NIZK proofs for satisfiability of pairing product equations. This allows us to construct constant/linear size cryptographic protocols for digital signatures, RCCA-secure encryption[CKN03], verifiable encryption and group signatures.

The only other way we know of to get linear size NIZK proofs/arguments for any practical language is the Fiat-Shamir heuristic: Make a 3-move public coin (honest verifier) zero-knowledge protocol non-interactive by computing the verifier's challenge as a hash of the statement and the initial protocol message. To argue security, one models the hash-function as a random oracle [BR93]. It is well known that using the random oracle model sometimes results in insecure real life protocols [CGH98, CGH04, Nie02, GK03, BBP04]. In comparison, our NIZK proofs have *provable security* under the DLIN assumption.

SIMULATION-SOUND EXTRACTABLE NIZK PROOFS. Combining the definitions of simulation-soundness introduced by Sahai [Sah01] and proofs of knowledge from De Santis and Persiano [DP92], we get simulation-sound extractability. Here the simulator first creates a simulated CRS together with a simulation trapdoor and an extraction trapdoor. We require that even after the adversary has seen simulated proofs on arbitrary statements, if it constructs a new valid proof on any statement, then we can extract a witness. Simulation-sound extractability is a very strong notion, in particular it implies non-malleability as defined by De Santis et al. [DDO$^+$02].

We construct a simulation-sound extractable NIZK proof for satisfiability of pairing product equations. Our NIZK proof has a CRS with a description of the group and a constant number of group elements, and the proofs consist of $\mathcal{O}(n + \ell)$ group elements.

RELATED WORK ON SIMULATION-SOUND NIZK PROOFS. As stated before, our interest in this paper is satisfiability of pairing products equations. However, in order to compare our scheme with previous work let us look at the case of Circuit Satisfiability. [Sah01] constructed a one-time simulation-sound NIZK proof system using techniques from Dwork, Dolev and Naor [DDN00]. Later a construction for unbounded simulation-sound extractable NIZK arguments was given by [DDO$^+$02], where the adversary can see many simulated arguments of arbitrary statements. The schemes from both these papers are based on trapdoor permutations but are not practical. For the sake of fairness in evaluating the quality of our contribution, we have also considered whether the techniques from [GOS06b] could be used to get good efficiency for simulation-sound extractability. The answer to this question seems to be negative, the best construction we can think of using GOS-techniques gives an additive polynomial size overhead.

Scheme	NIZK proof bit size	Assumption
[DDO$^+$02]	$\mathcal{O}(\lvert C\rvert \mathrm{poly}(k))$	Trapdoor permutations
Potential use of [GOS06b] techniques	$\mathcal{O}(\lvert C\rvert k + \mathrm{poly}(k))$	Subgroup decision
This paper	$\mathcal{O}(\lvert C\rvert k)$	DLIN

Figure 1: Comparison of simulation-sound extractable proofs for Circuit Satisfiability

COMMON REFERENCE STRING VERSUS UNIFORM RANDOM STRING. We will construct NIZK proofs and simulation-sound extractable NIZK proofs in the common reference string model, where the prover and the verifier both have access to a CRS chosen according to some distribution. If this distribution is uniform at random we call it the uniform random string model. In some settings it is easier to work with a URS, for instance a URS can easily be jointly generated using multi-party computation techniques.

Our NIZK proofs use a common reference string that contains a description of a bilinear group and a number of group elements. Depending on the group elements, the CRS will give either perfect soundness of perfect zero-knowledge. With overwhelming probability random group elements will lead to a perfect soundness CRS. Assuming that we can use a uniform random string to get a description of a DLIN group and a number of random group elements, we will therefore get NIZK proofs and simulation-sound NIZK proofs in the URS-model. Since there is a negligible chance of picking a perfect zero-knowledge CRS, this gives statistical soundness instead of perfect soundness, which is the best we can hope for in the URS-model. We remark that natural candidates for bilinear DLIN groups based on elliptic curves are efficiently samplable from a URS [GOS06a]. For the sake of simplicity we will just work with the CRS-model in the paper, but invite the reader to note that all constructions work in the URS-model as well.

1.3 An Application: Constant Size Group Signatures

Group signatures, introduced by Chaum and van Heyst [CvH91], allow a member to sign messages anonymously on behalf of a group. A group manager controls the group and decides who can join. In case of abuse, the group manager is able to open a signature to reveal who the signer is. It is hard to design group signatures and most schemes [CS97, CM98, ACJT00, CL02, AdM03, CG04, KTY04, CL04, BBS04, FI05, KY05] use the random oracle model in the security proof.

Bellare, Micciancio and Warinschi [BMW03] suggest rigorous security definitions for group signatures in the *static* case where the set of members is fixed from the start and never changes. Bellare, Shi and Zhang

[BSZ05] extend the security model to the partially *dynamic* case where the group manager can enroll new members in the group. Both [BMW03] and [BSZ05] suggest constructions of group signatures based on trapdoor permutations. These constructions are very inefficient and only indicate feasability.

Boyen and Waters [BW06] use a combination of the Waters signature scheme [Wat05] and the [GOS06b] NIZK proofs. They assume a static setting and as part of a group signature they encrypt the identity of the signer bit by bit. This means that a group signature consists of $\mathcal{O}(\log n)$ group elements, where n is the number of members in the group. The group signature scheme satisfies a relaxed version of the [BMW03] security definition, where the anonymity is guaranteed only when no signatures have been opened and traced to the signer. In comparison, the full-anonymity definition in [BMW03] demands that anonymity is preserved even when the adversary can get an opening of any other signature than the challenge.

Ateniese et al. [ACHdM05] use a bilinear group of prime order. The advantage of this scheme is that it is very efficient, a group signature consists of 8 group elements. However, they use several strong security assumptions and their security model is even weaker than that of [BW06] since it does not protect against key-exposures; knowledge of a signing key immediately allows one to tell which signatures this member has made. In comparison, the BMW,BSZ-models do guard against key exposure.

The tools in this paper give a construction of group signatures where both keys and signatures consist of a constant number of group elements. The construction involves carefully constructing and tailoring a signature scheme and the simulation-sound extractable NIZK proof system such that they fit each other. The constant is large; we do not claim this to be a practical scheme. Rather this should be seen as an interesting feasibility result; under a simple and natural security assumption there exists an up to a constant optimal dynamic group signature scheme satisfying the strong security definitions from [BMW03, BSZ05].

Scheme	Signature in bits	Security model	Assumption
[BMW03]	$\mathrm{poly}(k)$	[BMW03] (fixed group)	Trapdoor permutations
[BSZ05]	$\mathrm{poly}(k)$	[BSZ05] (dynamic group)	Trapdoor permutations
[BW06]	$3k + 2k \log n$	[BMW03], CPA-anonymity	Subgroup decision and CDH
[ACHdM05]	$8k$	UC-model, non-adaptive adversary	Strong SXDH, q-EDH, strong LRSW
This paper	$\mathcal{O}(k)$	[BSZ05]	DLIN

Figure 2: Comparison of group signature schemes

2 Definitions: Non-interactive Zero-Knowledge Proofs

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 CRS σ. The prover takes as input (σ, x, w) and produces a proof π. 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) a proof system for R if it has the completeness and soundness properties described below.

In this paper, we will extend the usual definitions of NIZK proofs by allowing the relation R to depend on the CRS σ. In our constructions, the CRS will contain a description of a bilinear group as well as some group elements, and we will make NIZK proofs for relations over this group and these group elements. This all builds up to an NIZK proof for satisfiability of pairing product equations. The relation for satisfiability of pairing product equations does not depend on the group elements in the CRS, but it still depends on the group in question. One way of looking at this result is that given a DLIN group, we can formulate the problem of satisfiability of pairing product equations, and we can on top of this group construct a CRS so we can prove satisfiability of pairing product equations.

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

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

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

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

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 adversaries $\mathcal{A}$ we have

$$\Pr\Big[\sigma \leftarrow K(1^k) : \mathcal{A}(\sigma) = 1\Big] = \Pr\Big[(\sigma,\xi) \leftarrow E_1(1^k) : \mathcal{A}(\sigma) = 1\Big].$$

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

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

(UNBOUNDED) COMPUTATIONAL ZERO-KNOWLEDGE. We call (K,P,V) an NIZK proof for R if there exists a polynomial time simulator $S = (S_1, S_2)$ with the following zero-knowledge property. For all non-uniform polynomial time adversaries $\mathcal{A}$ we have

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

where $S(\sigma,\tau,x,w) = S_2(\sigma,\tau,x)$ for $(x,w) \in R$ and both oracles output `failure` if $(x,w) \notin R$. Here $f(k) \approx g(k)$ means that there exists a negligible function ν so $|f(k) - g(k)| < \nu(k)$.

(UNBOUNDED) SIMULATION SOUNDNESS. Simulating a proof for a false statement might jeopardize the soundness of the proof system. We say an NIZK proof is simulation sound if an adversary cannot prove any false statement even after seeing simulated proofs of arbitrary statements.

More precisely, an NIZK proof is simulation sound if for all non-uniform polynomial time adversaries we have

$$\Pr\Big[(\sigma,\tau) \leftarrow S_1(1^k); (x,\pi) \leftarrow \mathcal{A}^{S_2(\sigma,\tau,\cdot)}(\sigma) : (x,\pi) \notin Q \text{ and } x \notin L \text{ and } V(\sigma,x,\pi) = 1\Big] \approx 0,$$

where Q is the list of simulation queries and responses (x_i, π_i).

(UNBOUNDED) SIMULATION SOUND EXTRACTABILITY. Combining simulation soundness and knowledge extraction, we may require that even after seeing many simulated proofs, whenever the adversary makes a new proof we are able to extract a witness. We call this property simulation sound extractability. Simulation sound extractability implies simulation soundness, because if we can extract a witness from the adversary's proof, then obviously the statement must belong to the language in question.

Consider an NIZK proof of knowledge (K,P,V,S_1,S_2,E_1,E_2). Let SE_1 be an algorithm that outputs (σ,τ,ξ) such that it is identical to S_1 when restricted to the first two parts (σ,τ). We say the NIZK proof is simulation sound if for all non-uniform polynomial time adversaries we have

$$\Pr\Big[(\sigma,\tau,\xi) \leftarrow SE_1(1^k); (x,\pi) \leftarrow \mathcal{A}^{S_2(\sigma,\tau,\cdot)}(\sigma,\xi); w \leftarrow E_2(\sigma,\xi,x,\pi) :$$
$$(x,\pi) \notin Q \text{ and } (x,w) \notin R \text{ and } V(\sigma,x,\pi) = 1\Big] \approx 0,$$

where Q is the list of simulation queries and responses (x_i, π_i).[1]

Simulation sound extractability implies non-malleability as defined and proven in [DDO$^+$02]. As we shall see in Appendix 6.1 it also implies universally composable NIZK secure against adaptive adversaries in a model where we allow parties to erase data from their tapes.

COMPOSABLE ZERO-KNOWLEDGE. We will strengthen the definition of zero-knowledge in the following way. First, we require that an adversary cannot distinguish a real CRS from a simulated CRS. Second, we require that the adversary, *even when it gets access to the secret key τ*, cannot distinguish real proofs on a simulated CRS from simulated proofs. A hybrid argument shows that it is sufficient to require that $\mathcal{A}$ cannot distinguish one real proof from one simulated proof.

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

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

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

$$\begin{aligned} & \Pr\left[(\sigma, \tau) \leftarrow S_1(1^k); (x, w) \leftarrow \mathcal{A}(\sigma, \tau); \pi \leftarrow P(\sigma, x, w) : \mathcal{A}(\pi) = 1 \text{ and } (x, w) \in R\right] \\ \approx\ & \Pr\left[(\sigma, \tau) \leftarrow S_1(1^k); (x, w) \leftarrow \mathcal{A}(\sigma, \tau); \pi \leftarrow S_2(\sigma, \tau, x) : \mathcal{A}(\pi) = 1 \text{ and } (x, w) \in R\right]. \end{aligned}$$

In [DDO$^+$02] reference indistinguishability is also separated from simulation indistinguishability. They require the simulated CRS to be statistically indistinguishable from a real CRS, whereas we are satisfied with computational indistinguishability. This is necessary to obtain NIZK proofs, their schemes are arguments that are only secure against a polynomial time prover.

On the other hand, our definition of simulation indistinguishability is stronger than the definition in [DDO$^+$02]. We allow the adversary to see the simulation trapdoor τ, whereas they do not give the adversary such power.

Theorem 2 *If (K, P, V, S_1, S_2) is a proof system with composable zero-knowledge, then it is unbounded zero-knowledge.*

Proof. Reference string indistinguishability implies

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

by the indistinguishability of the reference strings, since $\mathcal{A}$ can simply run the prover itself.

Let $q(k)$ be an upper bound on the number of queries $\mathcal{A}$ can ask the oracle, i.e., let for instance $q(k)$ be the run-time of $\mathcal{A}$. Define $PS[i](\sigma, \tau, \cdot, \cdot)$ to be an oracle that on query $j \in 1, \ldots, q(k)$ with a valid pair $(x, w) \in R$ responds with $S_2(\sigma, \tau, x)$ if $j \leq i$ and $P(\sigma, x, w)$ if $j > i$. Notice $P(\sigma, \cdot, \cdot) = PS[0](\sigma, \tau, \cdot, \cdot)$ and $S(\sigma, \tau, \cdot, \cdot) = PS[q(k)](\sigma, \tau, \cdot, \cdot)$. A hybrid argument shows that if (K, P, V, S_1, S_2) is not unbounded zero-knowledge, then for some $0 \leq i < q(k)$ the adversary must be able to distinguish $PS[i]$ from $PS[i+1]$.

To conclude the proof, we observe that simulation indistinguishability implies that for all i we have

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

[1]It is optional for our purposes whether to give the adversary access to ξ or not, but since we can prove the stronger statement we define simulation sound extractability as described above.

since the oracles can be efficiently implemented if one knows σ, τ and inserts the challenge π as the answer to query i. $\square$

Our motivation for introducing this stronger notion of zero-knowledge is that it allows different zero-knowledge proofs to use the same CRS. Suppose we have relations $R_1, \ldots, R_n$ and corresponding NIZK proofs with composable zero-knowledge using the same key generator and CRS simulator K, S_1. A hybrid argument shows that no polynomial time adversary can distinguish real proofs or simulated proofs for relation R_i, *even if it sees arbitrary proofs or simulations for statements in $L_{j\neq i}$ using the same CRS*. The reason this is the case is that in the definition of simulation indistinguishability we give τ to the adversary, so it can itself implement the simulator S_2 for any of the relations.

In our paper, all the NIZK proofs will indeed generate the CRS in the same way and also simulate the CRS in the same way, so we get better performance by not having to deal with different common reference strings for each proof system. At the same time, it simplifies our exposition.

3 A Homomorphic Commitment Scheme

We use the cryptosystem from Section 1.1 to create a homomorphic commitment scheme such that depending on how we generate the public key we get either a perfectly binding commitment scheme or a perfectly hiding trapdoor commitment scheme. The idea is that if K is an encryption of 1, then $K^m E(1; r, s)$ is also an encryption of 1 and we have a perfectly hiding commitment to m. On the other hand, if K is not an encryption of 1, then $K^m E(1; r, s)$ is perfectly binding.

Perfectly binding key generation: Let $ck = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w)$ where f, h is a public key for the cryptosystem and $(u, v, w) = (f^{r_u}, h^{s_v}, g^{t_w})$ with $t_w \neq r_u + s_v$ is an encryption of a non-trivial element.

Perfectly hiding trapdoor key generation: Let $ck = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w)$ where f, h is a public key for the cryptosystem and $(u, v, w) = (f^{r_u}, h^{s_v}, g^{r_u+s_v})$ is an encryption of 1.

The corresponding trapdoor key is $tk = (ck, x, y, r_u, s_v)$.

Commitment: To commit to message $m \in \mathbb{Z}_p$ pick $r, s \leftarrow \mathbb{Z}_p$ and let the commitment be $c = (c_1, c_2, c_3) = \mathrm{com}(m; r, s) = (u^m f^r, v^m h^s, w^m g^{r+s})$.

The commitment schemes $(K_{\text{binding}}, \mathrm{com})$ and $(K_{\text{hiding}}, \mathrm{com})$ have several nice properties. The CPA-security of the cryptosystem implies that one cannot distinguish perfect binding keys from perfect hiding keys. This in turn implies computational hiding respectively computational binding for the two schemes. The homomorphic property of the cryptosystem transfers to the commitment scheme.

$$\mathrm{com}(m_1 + m_2; r_1 + r_2, s_1 + s_2) = \mathrm{com}(m_1; r_1, s_1)\mathrm{com}(m_2; r_2, s_2).$$

For the perfectly binding commitment scheme, any $c \in \mathbb{G}^3$ is a commitment to some message $m \in \mathbb{Z}_p$.

4 Efficient Non-interactive Zero-Knowledge Proof Systems

4.1 Common Reference String

All our NIZK proof systems will use the same key generator K and reference string simulator S_1 described below. A common reference string is a public key for the perfectly binding commitment scheme described in the previous section. The soundness of the NIZK proofs will come from the perfect binding property of the commitment scheme, which will make it impossible for any adversary to cheat. In simulations, we will use a public key for a perfectly hiding commitment scheme as the simulated common reference string. Here the perfect hiding property of the commitment scheme is what enables us to simulate proofs.

Common reference string: On input 1^k do

1. $(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k)$
2. $(pk, sk) \leftarrow K_{\text{cpa}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$
3. $ck \leftarrow K_{\text{binding}}(pk)$
4. Return $\sigma = ck = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w)$

Simulated reference string: On input 1^k do

1. $(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k)$
2. $(pk, sk) \leftarrow K_{\text{cpa}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$
3. $(ck, tk) \leftarrow K_{\text{hiding}}(pk, sk)$
4. Let $\sigma = ck = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w)$
5. Let $\tau = tk = (\sigma, x, y, r_u, s_v)$
6. Return (σ, τ)

Reference string indistinguishability follows from the semantic security of the cryptosystem, which implies that no non-uniform polynomial time adversary can distinguish a perfectly binding ck from a perfectly hiding ck.

Lemma 3 *If $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ is a DLIN group, then (K, S_1) has reference string indistinguishability.*

A consequence of Lemma 3 is that in the rest of the paper we only need to prove simulation indistinguishability to prove composable zero-knowledge.

Both the common reference string generator K and the common reference string simulator S_1 first create a DLIN group honestly. This means that instead of generating the common reference strings from scratch, it is also possible to build any of the NIZK proofs we construct in the following sections on top of an already existing DLIN group. When doing so we write $\sigma \leftarrow K(p, \mathbb{G}, \mathbb{G}_1, e, g)$ or $(\sigma, \tau) \leftarrow S_1(p, \mathbb{G}, \mathbb{G}_1, e, g)$.

4.2 NIZK Proof for Commitment to Zero

The common reference string contains a public key for a commitment scheme. As a first step we suggest an NIZK proof for a commitment containing 0. Given the reference string $\sigma = (p, \mathbb{G}, \mathbb{G}_1, e, g, f, h, u, v, w)$ we define R_{zero} to be the relation consisting of commitments to 0, using the randomness as the witness. In other words, $R_{\text{zero}} = \{(c, (r, s)) \mid c = \text{com}(0; r, s)\}$. We construct an NIZK proof for R_{zero} below.

Proof of commitment to 0: Given a commitment $c = (c_1, c_2, c_3)$ and randomness r, s so $c = \text{com}(0; r, s)$ let the proof be $\pi = g^r$.

Verification: Given commitment (c_1, c_2, c_3) and a proof π the verifier returns 1 if and only if $e(g, c_1) = e(\pi, f)$ and $e(c_2, g) = e(h, c_3/\pi)$.

Simulation of proof: Given input $(\sigma, \tau, (c_1, c_2, c_3))$ the simulator S_{zero} generates the simulated proof $\pi = c_1^{1/x}$.

Theorem 4 $(K, P_{\text{zero}}, V_{\text{zero}}, S_1, S_{\text{zero}})$ *is an NIZK proof system for* R_{zero} *with perfect completeness, perfect soundness and composable zero-knowledge with perfect simulation indistinguishability under the DLIN assumption for* $\mathcal{G}$. *The proof consists of 1 group element. Verification corresponds to evaluating two pairing product equations.*

Proof.

Perfect completeness: A commitment (c_1, c_2, c_3) to 0 uniquely define r, s so $c_1 = f^r, c_2 = h^s, c_3 = g^{r+s}$. We have $e(g, c_1) = e(g, f^r) = e(g^r, f) = e(\pi, f)$ and $e(c_2, g) = e(h^s, g) = e(h, g^s) = e(h, c_3/\pi)$.

Perfect soundness: Given a commitment (c_1, c_2, c_3) and an acceptable proof π, we have $e(g, c_1) = e(\pi, f)$ showing us that there exists some r such that $c_1 = f^r, \pi = g^r$. There also exists some s such that $c_2 = h^s$. The second equation, $e(c_2, g) = e(h, c_3/\pi)$ reveals $e(h^s, g) = e(h, g^s) = e(h, c_3/\pi)$, showing that $c_3 = g^{r+s}$. This means, $c_1 = u^0 f^r, c_2 = v^0 h^s, c_3 = w^0 g^{r+s}$.

Composable zero-knowledge: We already have reference string indistinguishability, so all we need to prove is simulation indistinguishability. We have

$$\begin{aligned} & \Pr\left[(\sigma, \tau) \leftarrow S_1(1^k) : (c, (r, s)) \leftarrow \mathcal{A}(\sigma, \tau); \pi = g^r : \mathcal{A}(\pi) = 1\right] \\ = \; & \Pr\left[(\sigma, \tau) \leftarrow S_1(1^k) : (c, (r, s)) \leftarrow \mathcal{A}(\sigma, \tau); \pi = c_1^{1/x} : \mathcal{A}(\pi) = 1\right], \end{aligned}$$

since both computations yield the same uniquely defined π that will make the verifier accept the proof. $\square$

4.3 NIZK Proof for Commitment to Exponent.

Suppose, we have two elements a, b and a commitment c to the exponent m so $b = a^m$. We wish to form an NIZK proof for $R_{\text{expo}} = \{((a, b, c), (m, r, s)) \mid b = a^m, c = \text{com}(m; r, s)\}$.

The idea in the proof is straightforward. If $a \neq 1$ then one can use the bilinear map to verify that a pair of commitments π_1, π_m have the same exponent m so $\pi_m = \pi_1^m$. If π_1 is a commitment to 1, then π_m is a commitment to m. What remains is to prove that $\pi_1 \text{com}(-1; 0, 0)$ and $c_m \pi_m^{-1}$ are commitments to 0.

If $a = b = 1$, then the relation is trivially satisfied. However, later on we will be working on committed elements and it will not be straightforward to check whether an element is non-trivial. We therefore treat $a = 1$ as a special case and create an NIZK proof that works with the same verifier as for the $a \neq 1$ case.

Proof of commitment to exponent: Given common inputs a, b, c and a witness (m, r, s) the prover constructs the proof π as follows.

```
1. If $a = 1$
2.     $\pi_1 = \text{com}(1; 0, 0), \pi_m = c$
3.     $\pi_{\pi_1} \leftarrow P_{\text{zero}}(\sigma, \pi_1 \text{com}(-1; 0, 0), (0, 0))$
4.     $\pi_{\pi_m} \leftarrow P_{\text{zero}}(\sigma, c\pi_m^{-1}, (0, 0))$
5. Else if $a \neq 1$
6.     $r_1, s_1 \leftarrow \mathbb{Z}_p$
```

```
7.      $\pi_1 = \mathrm{com}(1; r_1, s_1)$
8.      $\pi_m = \pi_1^m$
9.      $\pi_{\pi_1} \leftarrow P_{\mathrm{zero}}(\sigma, \pi_1 \mathrm{com}(1; 0, 0)^{-1}, (r_1, s_1)))$
10.     $\pi_{\pi_m} \leftarrow P_{\mathrm{zero}}(\sigma, c\pi_m^{-1}, (r - mr_1, s - ms_1))$
11. The proof is $\pi = (\pi_1, \pi_m, \pi_{\pi_1}, \pi_{\pi_m})$.
```

Verification: Given a, b, c and the proof π do

1. Verify the NIZK proofs π_{π_1}, π_{π_m} for $\pi_1 \mathrm{com}(1; 0, 0)^{-1}, c\pi_m^{-1}$ being commitments to 0
2. Check $e(a, \pi_{m,1}) = e(b, \pi_{1,1}), e(a, \pi_{m,2}) = e(b, \pi_{1,2})$ and $e(a, \pi_{m,3}) = e(b, \pi_{1,3})$
3. Return 1 if all checks pass, else return 0

Simulated proof: On statement a, b, c we can simulate the proof in the following way

```
1. If $a = 1$
2.      $\pi_1 = \mathrm{com}(1; 0, 0), \pi_m = c$
3. Else if $a \neq 1$
4.      $r_1, s_1 \leftarrow \mathbb{Z}_p$
5.      $\pi_{1,1} = a^{xr_1}, \pi_{1,2} = a^{ys_1}, \pi_{1,3} = a^{r_1+s_1}$
6.      $\pi_{m,1} = b^{xr_1}, \pi_{m,2} = b^{ys_1}, \pi_{m,3} = b^{r_1+s_1}$
7. $\pi_{\pi_1} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, \pi_1 \mathrm{com}(-1; 0, 0))$
8. $\pi_{\pi_m} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, c\pi_m^{-1})$
9. The simulated proof is $\pi = (\pi_1, \pi_m, \pi_{\pi_1}, \pi_{\pi_m})$
```

Theorem 5 $(K, P_{\mathrm{expo}}, V_{\mathrm{expo}}, S_1, S_{\mathrm{expo}})$ *is an NIZK proof for* R_{expo} *with perfect completeness, perfect soundness and composable zero-knowledge with perfect simulation indistinguishability if the DLIN assumption holds for* $\mathcal{G}$*. A proof consists of 8 group elements. Verification consists of evaluating a set of pairing product equations.*

Proof.

Perfect completeness: Follows from perfect completeness of the NIZK proof for R_{zero}.

Perfect soundness: If $a \neq 1$ we learn from the bilinear map that $\exists m : b = a^m, \pi_m = \pi_1^m$. Perfect soundness of the NIZK proof for R_{zero} implies that π_1 is a commitment to 1 and $c\pi_1^{-m}$ is a commitment to 0. This implies that c is a commitment to m.

If $a = 1$ the bilinear map shows us that $1 = e(a, \pi_{m,1}) = e(b, \pi_{1,1}), 1 = e(a, \pi_{m,2}) = e(b, \pi_{1,2})$ and $1 = e(a, \pi_{m,3}) = e(b, \pi_{1,3})$. Since π_1 is a commitment to 1 at least one of $\pi_{1,1}, \pi_{1,2}, \pi_{1,3}$ must be non-trivial and therefore $b = 1$.

Composable zero-knowledge: Let us show that on a simulated common reference string, proofs and simulated proofs are perfectly indistinguishable. Observe first that on a simulated common reference string all valid commitments are on the form $f^{r_1}, h^{s_1}, g^{r_1+s_1}$. If $a \neq 1$, we pick r_1, s_1 at random and use them to generate two random commitments π_1, π_m such that $\pi_m = \pi_1^m$. So far, this gives us the same distribution as if they were generated by the prover. We conclude by noting that the NIZK proof for R_{zero} has perfect simulation of proofs on a simulated common reference string. If $a = 1$ the perfect simulation indistinguishability of the NIZK proof for R_{zero} also gives us a perfect simulation. □

4.4 NIZK Proof for Pedersen Commitment

Consider a Pedersen commitment $b = a^m g^t$ to m. We wish to make commitments to the opening m, t and make an NIZK proof that we have done so. In other words, we want an NIZK proof for $R_{\text{ped}} = \{((a, b, c_m, c_t), (m, t, r_m, s_m, r_t, s_t)) \mid b = a^m g^t, c_m = \text{com}(m; r_m, s_m), c_t = \text{com}(t; r_t, s_t)\}$.

The idea in the NIZK proof is as follows. We write

$$b = a^m g^t = a^{m+r_1 r+r_2 s} g^{r+s+t} (a^{r_1} g)^{-r} (a^{r_2} g)^{-s},$$

for randomly chosen r_1, r_2, r, s. We reveal a^{r_1}, a^{r_2} and make commitments π_{r_1}, π_{r_2} to r_1, r_2. Using the NIZK proof for R_{expo}, we can prove they have been correctly formed. We reveal $(a^{r_1} g)^r, (a^{r_2} g)^s$ and create commitments $\pi_{r,r_1} = \pi_{r_1}^r, \pi_{s,r_2} = \pi_{r_2}^s$. Using the bilinear map the verifier can check that the exponentiation is correct, so they contain rr_1 and sr_2. Using the homomorphic property, we have $c_m \pi_{r,r_1} \pi_{s,r_2}$ is a commitment to the exponent $m+r_1 r+r_2 s$. We also make commitments $\pi_{r,1}, \pi_{s,1}$ to r and s and prove they have been correctly formed. By the homomorphic property, we have that $c_t \pi_{r,1} \pi_{s,1}$ is a commitment to $r + s + t$. Revealing g^{r+s+t} indirectly also reveals $a^{m+r_1 r+r_2 s} = b(a^{r_1} g)^r (a^{r_2} g)^s g^{-(r+s+t)}$. Using NIZK proofs for R_{expo} we can prove it has been correctly formed. This demonstrates that $b = a^m g^t$ as required. Computational zero-knowledge will follow from the DLIN assumption, since g^{r+s} looks random given $(a^{r_1} g)^r$ and $(a^{r_2} g)^s$ and therefore in the simulation we can pick it a random exponent d instead of using $r + s$.

Proof for Pedersen commitment: The prover gets a statement (a, b, c_m, c_t) and a witness $(m, t, r_m, s_m, r_t, s_t)$ as input. The proof is constructed as follows

$r_1, r_2, r_3, s_3, \ldots, r_6, s_6, r, s \leftarrow \mathbb{Z}_p$ (If $a = 1$ let $r_1 = 0, \cdots, s = 0$)

$$\begin{array}{lll}
\pi_{a,1} = a^{r_1} & \pi_{1,1} = \text{com}(1; r_3, s_3) & \pi_{r_1} = \text{com}(r_1; r_4, s_4) \\
\pi_r = (\pi_{a,1} g)^r & \pi_{r,1} = \pi_{1,1}^r & \pi_{r,r_1} = \pi_{r_1}^r \\
 & & \\
\pi_{a,2} = a^{r_2} & \pi_{1,2} = \text{com}(1; r_5, s_5) & \pi_{r_2} = \text{com}(r_2; r_6, s_6) \\
\pi_s = (\pi_{a,2} g)^s & \pi_{s,1} = \pi_{1,2}^s & \pi_{s,r_2} = \pi_{r_2}^s \\
 & & \\
\pi_t = g^{r+s+t} & \pi_{t,1} = \text{com}(1; 0, 0)^{r+s+t} &
\end{array}$$

$$\begin{array}{l}
\pi_{\pi_{1,1}} \leftarrow P_{\text{zero}}(\sigma, \pi_{1,1}\text{com}(-1; 0, 0), (r_3, s_3)) \\
\pi_{\pi_{1,2}} \leftarrow P_{\text{zero}}(\sigma, \pi_{1,2}\text{com}(-1; 0, 0), (r_5, s_5)) \\
\pi_{\pi_0} \leftarrow P_{\text{zero}}(\sigma, c_t \pi_{r,1} \pi_{s,1} \pi_{t,1}^{-1}, (r_3 r + r_5 s + r_t, s_3 r + s_5 s + s_t)) \\
\pi_{\pi_{r_1}} \leftarrow P_{\text{expo}}(\sigma, (a, \pi_{a,1}, \pi_{r_1}), (r_1, r_4, s_4)) \\
\pi_{\pi_{r_2}} \leftarrow P_{\text{expo}}(\sigma, (a, \pi_{a,2}, \pi_{r_2}), (r_2, r_6, s_6)) \\
\pi_{\pi_a} \leftarrow P_{\text{expo}}(\sigma, (a, b\pi_r \pi_s \pi_t^{-1}, c_m \pi_{r,r_1} \pi_{s,r_2}), (r_1 r + r_2 s + m, r_4 r + r_6 s + r_m, s_4 r + s_6 s + s_m))
\end{array}$$

The proof is $\pi = (\pi_{a,1}, \ldots, \pi_{\pi_a})$.

Verification: On common input a, b, c_m, c_t and proof π do

1. Verify the NIZK proofs $\pi_{\pi_{1,1}}, \pi_{\pi_{1,2}}, \pi_{\pi_0}$ for $\pi_{1,1}, \pi_{1,2}$ being commitments to 1 and $c_t \pi_{r,1} \pi_{s,1} \pi_{t,1}^{-1}$ being a commitment to 0
2. Verify the NIZK proof $\pi_{\pi_{r_1}}$ for the existence of some r_1 so $\pi_{a,1} = a^{r_1}$ and π_{r_1} being a commitment to r_1

3. Verify the NIZK proof $\pi_{\pi_{r_2}}$ for the existence of some r_2 so $\pi_{a,2} = a^{r_2}$ and π_{r_2} being a commitment to r_2
4. Verify the NIZK proof π_{π_a} for the existence of some z so $b\pi_r\pi_s\pi_t^{-1} = a^z$ and $c_m\pi_{r,r_1}\pi_{s,r_2}$ being a commitment to z.
5. Using the bilinear map check $\exists r : \pi_r = (g\pi_{a,1})^r, \pi_{r,1} = \mathrm{com}(1;0,0)^r, \pi_{r,r_1} = \pi_{r_1}^r$
6. Using the bilinear map check $\exists s : \pi_s = (g\pi_{a,2})^s, \pi_{s,1} = \mathrm{com}(1;0,0)^s, \pi_{s,r_2} = \pi_{r_2}^s$
7. Using the bilinear map check $\exists z : \pi_t = g^z, \pi_{t,1} = \mathrm{com}(1;0,0)^z$
8. Return 1 if all checks pass, else return 0

Simulation: Given common input (a, b, c_m, c_t) we simulate the proof as follows

$$\alpha, \beta, r_3, s_3, \ldots, r_6, s_6, r, s, z \leftarrow \mathbb{Z}_p \qquad \text{(If } a = 1 \text{ let } \alpha = \beta = 1, r_3 = 0, \cdots, s = 0\text{)}$$

$$\begin{array}{lll}
\pi_{a,1} = g^{\alpha-1} & \pi_{1,1} = \mathrm{com}(0;\alpha r_3, \alpha s_3) & \pi_{r_1} = \mathrm{com}(0;\alpha r_4, \alpha s_4) \\
\pi_r = (\pi_{a,1}g)^r & \pi_{r,1} = \mathrm{com}(0;\alpha r r_3, \alpha r s_3) & \pi_{r,r_1} = \mathrm{com}(0;\alpha r r_4, \alpha r s_4) \\
 & & \\
\pi_{a,2} = g^{\beta-1} & \pi_{1,2} = \mathrm{com}(0;\beta r_5, \beta s_5) & \pi_{r_2} = \mathrm{com}(0;\beta r_6, \beta s_6) \\
\pi_s = (\pi_{a,2}g)^s & \pi_{s,1} = \mathrm{com}(0;\beta s r_5, \beta s s_5) & \pi_{s,r_2} = \mathrm{com}(0;\beta s r_6, \beta s s_6) \\
 & & \\
\pi_t = g^z & \pi_{t,1} = \mathrm{com}(1;0,0)^z & \text{(If } a = 1 \text{ let } \pi_t = b, \pi_{t,1} = (b^{xr_u}, b^{ys_v}, b^{r_u+s_v})\text{)}
\end{array}$$

$$\begin{array}{l}
\pi_{\pi_{1,1}} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, \pi_{1,1}\mathrm{com}(-1;0,0)) \\
\pi_{\pi_{1,2}} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, \pi_{1,2}\mathrm{com}(-1;0,0)) \\
\pi_{\pi_0} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, c_t\pi_{r,1}\pi_{s,1}\pi_{t,1}^{-1}) \\
\pi_{\pi_{r_1}} \leftarrow S_{\mathrm{expo}}(\sigma, \tau, (a, \pi_{a,1}, \pi_{r_1})) \\
\pi_{\pi_{r_2}} \leftarrow S_{\mathrm{expo}}(\sigma, \tau, (a, \pi_{a,2}, \pi_{r_2})) \\
\pi_{\pi_a} \leftarrow S_{\mathrm{expo}}(\sigma, \tau, (a, b\pi_r\pi_s\pi_t^{-1}, c_m\pi_{r,r_1}\pi_{s,r_2}))
\end{array}$$

The simulated proof is $\pi = (\pi_{a,1}, \ldots, \pi_{\pi_a})$

Theorem 6 $(K, P_{\mathrm{ped}}, V_{\mathrm{ped}}, S_1, S_{\mathrm{ped}})$ *is an NIZK proof system for* R_{ped} *with perfect completeness, perfect soundness and composable zero-knowledge if the DLIN assumption holds for* $\mathcal{G}$*. A proof consists of 59 group elements. Verification consists of evaluating a set of pairing product equations.*

Proof.

Perfect completeness: Follows from the perfect completeness of $(K, P_{\mathrm{zero}}, V_{\mathrm{zero}})$ and $(K, P_{\mathrm{expo}}, V_{\mathrm{expo}})$.

Perfect soundness: From the perfect soundness of $(K, P_{\mathrm{zero}}, V_{\mathrm{zero}})$ we get $\pi_{1,1}, \pi_{1,2}$ contain 1 and $c_t\pi_{r,1}\pi_{s,1}\pi_{t,1}^{-1}$ contains 0. This means $\pi_{r,1}$ is a commitment to r and $\pi_{s,1}$ is a commitment to s, and consequently $\pi_{t,1}$ is a commitment to $r + s + t$. This in turn means that $\pi_t = g^{r+s+t}$.

From the perfect soundness of $(K, P_{\mathrm{expo}}, V_{\mathrm{expo}})$ we see that there exists r_1, r_2 so $\pi_{a,1} = a^{r_1}$ and $\pi_{a,2} = a^{r_2}$, and at the same time π_{r_1} is a commitment to r_1 and π_{r_2} is a commitment to r_2. This means, $c_m\pi_{r,r_1}\pi_{s,r_2}$ contains $m + rr_1 + sr_2$.

From π_{π_a} we get

$$b(g\pi_{a,1})^r(g\pi_{a,2})^s\pi_t^{-1} = a^{m+r_1r+r_2s} \quad \text{so} \quad b = a^m g^t.$$

Composable zero-knowledge: The polynomial time adversary $\mathcal{A}(\sigma, \tau)$ produces a statement (a, b, c_m, c_t) and a witness $(m, t, r_m, s_m, r_t, s_t)$. We now have to argue it cannot distinguish a proof from P_{ped} from a simulated proof from S_{ped}.

By the composable zero-knowledge properties of the NIZK proofs for R_{zero} and R_{expo} we can simulate the proofs $\pi_{\pi_{1,1}}, \ldots, \pi_{\pi_a}$ without $\mathcal{A}$ being able to distinguish.

In case $a = 1$ straightforward verification shows that on a simulated reference string, proofs and simulated proofs are perfectly indistinguishable. This leaves the case where $a \neq 1$.

Let us start with the way a prover produces a proof. Since the commitments are perfectly hiding, we can select $\pi_{1,1}, \pi_{1,2}, \pi_{r_1}$ and π_{r_2} as random commitments to 0. Since a is a generator for $\mathbb{G}$, we get the same distribution if we select $\pi_{a,1} = g^{\alpha-1}$ and $\pi_{a,2} = g^{\beta-1}$ for random α, β. With overwhelming probability $\alpha, \beta \neq 0$ in which case r_3, s_3 and $\alpha r_3, \alpha s_3$ have the same distribution when picking r_3, r_4 random. In a similar fashion, we change the randomizers of the other commitments to have α or β factors.

We have now modified the proof, such that the only difference from the simulation is that we select $z = r + s + t$, while in the simulation we pick z at random. We will show that if $\mathcal{A}$ can distinguish a simulated proof from a partially simulated proof, then we can use it to break the DLIN assumption for $\mathcal{G}$.

Let $g^{\alpha}, g^{\beta}, g^{\alpha r}, g^{\beta s}, g^{d}$ be a DLIN challenge, where $d = r + s$ or d is random. Since we know the discrete logarithms of f, h, u, v, w with respect to g, we can given g^d but without knowing d still compute $\text{com}(1; 0, 0)^d = ((g^d)^{xr_u}, (g^d)^{ys_v}, (g^d)^{r_u+s_v})$. We can also given g^{α} compute a random commitment by selecting r_3, s_3 at random and letting the commitment be $\text{com}(0; \alpha r_3, \alpha s_3) = ((g^{\alpha})^{xr_3}, (g^{\alpha})^{ys_3}, g^{\alpha})^{r_3+s_3}$. Similar techniques explain how we compute the various commitments listed below.

Let us set $\pi_{a,1} = g^{-1}g^{\alpha}, \pi_r = g^{\alpha r}, \pi_{a,2} = g^{-1}g^{\beta}, \pi_s = g^{\beta s}$ and $\pi_t = g^d g^t$. We set $\pi_{1,1} = \text{com}(0; \alpha r_3, \alpha s_3), \pi_r = \text{com}(0; \alpha r r_3, \alpha r s_3)$ and $\pi_{1,2} = \text{com}(0; \beta r_5, s_5), \pi_s = \text{com}(0; \beta s r_5, \beta s s_5)$. We set $\pi_{r_1} = \text{com}(0; \alpha r_4, \alpha s_4), \pi_{r,r_1} = \text{com}(0; \alpha r r_4, \alpha r s_4)$ and $\pi_{r_2} = \text{com}(0; \beta r_6, \beta s_6), \pi_{s,r_2} = \text{com}(0; \beta s r_6, \beta s s_6)$. Finally, compute $\pi_t = g^d g^t$.

In case, $d = r + s$ then this gives us a partially simulated proof with $z = r + s + t$. In case d is random, this gives us a simulated proof with z random. If $\mathcal{A}$ could distinguish between partially simulated proofs and simulated proofs, then we would be able to break the DLIN assumption. □

4.5 NIZK Proof for Multi-message Pedersen Commitment

Let us generalize the Pedersen commitment to a many messages $b = g^t \prod_{i=1}^{n} a_i^{m_i}$. We wish to make an NIZK proof for having committed to $t, m_1, \ldots, m_n$ so b is a multi-exponentiation to these messages. More precisely, we want an NIZK proof for the relation $R_{\text{m-ped}} = \{((a_1, \ldots, a_n, b, c_t, c_1, \ldots, c_n), (t, r_t, s_t, m_1, r_1, s_1, \ldots, m_n, r_n, s_n)) \mid b = g^t \prod_{i=1}^{n} a_i^{m_i}, c_t = \text{com}(t; r_t, s_t), c_i = \text{com}(m_i, r_i, s_i)\}$.

The idea in this NIZK proof is to split b into n Pedersen commitments and use NIZK proof from the previous section. Write

$$b = \prod_{i=1}^{n} (a_i^{m_i} g^{t_i}),$$

where $t = \sum_{i=1}^{n} t_i$, make commitments to the t_i's and make an NIZK proof for R_{ped} for each of these components.

Proof for multi-message Pedersen commitment: The prover gets a statement $(a_1, \dots, a_n, b, c_t, c_1, \dots, c_n)$ and a witness $(t, r_t, s_t, m_1, r_1, s_1, \dots, m_n, r_n, s_n)$ as input. The proof is constructed as follows

```
1. $t_1, r_{t_1}, s_{t_1}, \dots, t_{n-1}, r_{t_{n-1}}, s_{t_{n-1}} \leftarrow \mathbb{Z}_p, t_n = t - \sum_{i=1}^{n-1} t_i, r_{t_n} = r_t - \sum_{i=1}^{n-1} r_{t_i}, s_{t_n} = s_t - \sum_{i=1}^{n-1} s_{t_n}$
2. For $i = 1$ to $n$ do
3.     $\pi_i = a_i^{m_i} g^{t_i}$
4.     $\pi_{t_i} = \mathrm{com}(t_i; r_{t_i}, s_{t_i})$
5.     $\pi_{\pi_i} \leftarrow P_{\mathrm{ped}}(\sigma, (a_i, \pi_i, c_i, \pi_{t_i}), (m_i, t_i, r_i, s_i, r_{t_i}, s_{t_i}))$
```

The proof is $\pi = (\pi_1, \pi_{t_1}, \pi_{\pi_1}, \dots, \pi_{n-1}, \pi_{t_{n-1}}, \pi_{\pi_{n-1}}, \pi_{\pi_n})$

Verification: On common input $a_1, \dots, a_n, b, c_t, c_1, \dots, c_n$ and proof π do

```
1. Let $\pi_n = b \prod_{i=1}^{n-1} \pi_i^{-1}$ and $\pi_{t_n} = c_t \prod_{i=1}^{n-1} \pi_{t_i}^{-1}$
2. Verify the NIZK proofs $\pi_{\pi_1}, \dots, \pi_{\pi_n}$
3. Return 1 if all checks pass, else return 0
```

Simulation: Given common input $(a_1, \dots, a_n, b, c_t, c_1, \dots, c_n)$ we simulate the proof as follows

```
1. $t_1, r_{t_1}, s_{t_1}, \dots, t_{n-1}, r_{t_{n-1}}, s_{t_{n-1}} \leftarrow \mathbb{Z}_p$
2. For $i = 1$ to $n-1$ let $\pi_i = g^{t_i}$ and $\pi_{t_i} = \mathrm{com}(0; r_{t_i}, s_{t_i})$
3. Let $\pi_i = b \prod_{i=1}^{n-1} \pi_i^{-1}$ and $\pi_{t_n} = c_t \prod_{i=1}^{n-1} \pi_{t_i}^{-1}$
4. For $i = 1$ to $n$ simulate $\pi_{\pi_i} \leftarrow S_{\mathrm{ped}}(\sigma, \tau, (a_i, \pi_i, c_i, \pi_{t_i}))$
```

The simulated proof is $\pi = (\pi_1, \pi_{t_1}, \pi_{\pi_1}, \dots, \pi_{n-1}, \pi_{t_{n-1}}, \pi_{\pi_{n-1}}, \pi_{\pi_n})$

Theorem 7 $(K, P_{\mathrm{m-ped}}, V_{\mathrm{m-ped}}, S_1, S_{\mathrm{m-ped}})$ *is an NIZK proof system for* $R_{\mathrm{m-ped}}$ *with perfect completeness, perfect soundness and composable zero-knowledge if the DLIN assumption holds for* $\mathcal{G}$*. The proof consists of* $63n - 4$ *group elements. The verification consists of evaluating a set of pairing product equations.*

Proof.

Perfect completeness: Follows from the perfect completeness of $(K, P_{\mathrm{ped}}, V_{\mathrm{ped}})$.

Perfect soundness: From the perfect soundness of $(K, P_{\mathrm{ped}}, V_{\mathrm{ped}})$ we know that c_i, π_{t_i} are commitments to m_i, t_i so $\pi_i = a_i^{m_i} g^{t_i}$. Since $c_t = \prod_{i=1}^n \pi_{t_i}$ we see that $t = \sum_{i=1}^n t_i$. We now have

$$b = \prod_{i=1}^n \pi_i = \prod_{i=1}^n a_i^{m_i} g^{t_i} = g^{\sum_{i=1}^n t_i} \prod_{i=1}^n a_i^{m_i} = g^t \prod_{i=1}^n a_i^{m_i}$$

as required.

Composable zero-knowledge: On a simulated reference string, the adversary cannot distinguish between proofs and simulated proofs $\pi_{\pi_1}, \dots, \pi_{\pi_n}$. Since the commitment scheme is perfectly hiding the adversary also cannot distinguish between commitments $\pi_{t_i} \leftarrow \mathrm{com}(t_i)$ and $\pi_{t_i} \leftarrow \mathrm{com}(0)$, where we in both cases fit $\pi_{t_n} = c_t \prod_{i=1}^{n-1} \pi_{t_i}^{-1}$. The Pedersen commitment scheme is also perfectly hiding, so we cannot distinguish between picking $\pi_i = a_i^{m_i} g^{t_i}$ and $\pi_i = g^{t_i}$ for random t_i, where we fit $\pi_n = b \prod_{i=1}^{n-1} \pi_i^{-1}$. $\square$

4.6 NIZK Proof for Multiplicative Relationship

Consider three commitments c_a, c_b, c_c such that the corresponding messages have the relationship $m_c = m_a m_b$. We wish to construct an NIZK proof for such a multiplicative relationship. More precisely, a composable NIZK proof for $R_{\text{mult}} = \{((c_a, c_b, c_c), (m_a, r_a, s_a, m_b, r_b, s_b, r_c, s_c)) \mid c_a = \text{com}(m_a; r_a, s_a), c_b = \text{com}(m_b; r_b, s_b), c_c = \text{com}(m_a m_b; r_c, s_c)\}$.

If c_a, c_b, c_c have a multiplicative relationship, then

$$c_c = c_b^{m_a} \text{com}(0; r_c - m_a r_b, s_c - m_a s_b)$$

and vice versa. To prove the latter relationship, it suffices to reveal m_a, and prove that $c_a \text{com}(-m_a; 0, 0)$ and $c_c c_b^{-m_a}$ are commitments to 0. To get zero-knowledge, we want to somehow tweak this idea in a way such that m_a is not revealed directly.

The main trick in the following is to pick exponents r, s at random, which will be used to hide m_a. We want to prove

$$c_a \text{com}(1; 0, 0)^{-(r+s+m_a)} \text{com}(1; 0, 0)^r \text{com}(1; 0, 0)^s \quad \text{and} \quad c_c c_b^{-(r+s+m_a)} c_b^r c_b^s$$

are commitments to 0. If $\pi_{0,1}, \pi_{0,2}, \pi_{0,3}, \pi_{0,4}$ are commitments to 0 we can obscure things by instead proving that

$$c_a \text{com}(1; 0, 0)^{-(r+s+m_a)} (\text{com}(1; 0, 0)\pi_{0,1})^r (\text{com}(1; 0, 0)\pi_{0,3})^s \quad \text{and} \quad c_c c_b^{-(r+s+m_a)} (c_b \pi_{0,2})^r (c_b \pi_{0,4})^s$$

are commitments to 0.

Revealing the components $\text{com}(1; 0, 0)^{r+s+m_a}, c_b^{r+s+m_a}$, the verifier can use the bilinear maps to check that there exists some common exponent $t = r + s + m_a$, even though it cannot compute the exponent itself. Similarly, revealing $(\text{com}(1; 0, 0)\pi_{0,1})^r, (c_b \pi_{0,2})^r$ and $(\text{com}(1; 0, 0)\pi_{0,3})^s, (c_b \pi_{0,4})^s$ allows the verifier to check that there exist common exponents r, s. So far, we are performing the same exponentiations on $\text{com}(1; 0, 0)$ and c_b to get respectively c_a and c_c. This shows that

$$c_a \text{com}(1; 0, 0)^{r+s-t} \quad \text{and} \quad c_c c_b^{r+s-t}$$

are both commitments to 0. The only way this can be possible is when $m_a = t - r - s$.

Computational zero-knowledge will follow from the fact that while we use the same exponents, we use different bases. Therefore, at no point is any element itself raised to m_a, which the adversary could potentially use to detect whether it was a correct proof or one created by a simulator, which does not know m_a.

Proof of multiplicative relationship: $P_{\text{mult}}(\sigma, (c_a, c_b, c_c), (m_a, r_a, s_a, m_b, r_b, s_b, r_c, s_c))$ runs as follows

$$r_1, s_1, \dots, r_4, s_4, r, s \leftarrow \mathbb{Z}_p$$

$$\begin{array}{ll} \pi_{0,1} = \mathrm{com}(0; r_1, s_1) & \pi_{0,2} = \mathrm{com}(0; r_2, s_2) \\ \pi_r = (\mathrm{com}(1; 0, 0)\pi_{0,1})^r & \pi_{r,b} = (c_b \pi_{0,2})^r \end{array}$$

$$\begin{array}{ll} \pi_{0,3} = \mathrm{com}(0; r_3, s_3) & \pi_{0,4} = \mathrm{com}(0; r_4, s_4) \\ \pi_s = (\mathrm{com}(1; 0, 0)\pi_{0,3})^s & \pi_{s,b} = (c_b \pi_{0,4})^s \end{array}$$

$$\pi_t = \mathrm{com}(1; 0, 0)^{r+s+m_a} \quad \pi_{t,b} = c_b^{r+s+m_a}$$

$$\begin{array}{l} \pi_{\pi_{0,1}} \leftarrow P_{\mathrm{zero}}(\sigma, \pi_{0,1}, (r_1, s_1)) \\ \pi_{\pi_{0,2}} \leftarrow P_{\mathrm{zero}}(\sigma, \pi_{0,2}, (r_2, s_2)) \\ \pi_{\pi_{0,3}} \leftarrow P_{\mathrm{zero}}(\sigma, \pi_{0,3}, (r_3, s_3)) \\ \pi_{\pi_{0,4}} \leftarrow P_{\mathrm{zero}}(\sigma, \pi_{0,4}, (r_4, s_4)) \\ \pi_{\pi_{0,5}} \leftarrow P_{\mathrm{zero}}(\sigma, c_a \pi_r \pi_s \pi_t^{-1}, (r_1 r + r_3 s + r_a, s_1 r + s_3 s + s_a)) \\ \pi_{\pi_{0,6}} \leftarrow P_{\mathrm{zero}}(\sigma, c_c \pi_{r,b} \pi_{s,b} \pi_{t,b}^{-1}, (r_2 r + r_4 s + r_c - m_a r_b, s_2 r + s_4 s + s_c - m_a s_b)) \end{array}$$

The proof is $\pi = (\pi_{0,1}, \dots, \pi_{\pi_{0,6}})$.

Verification: $V_{\mathrm{mult}}(\sigma, (c_a, c_b, c_c), \pi)$ runs as follows

1. Verify the NIZK proofs $\pi_{\pi_{0,1}}, \dots, \pi_{\pi_{0,6}}$ for $\pi_{0,1}, \dots, \pi_{0,4}$ and $c_a \pi_r \pi_s \pi_t^{-1}, c_c \pi_{r,b} \pi_{s,b} \pi_{t,b}^{-1}$ being commitments to 0
2. Using the bilinear map[2], check $\exists r : \pi_r = (\mathrm{com}(1; 0, 0)\pi_{0,1})^r$ and $\pi_{r,b} = (c_b \pi_{0,2})^r$
3. Using the bilinear map, check $\exists s : \pi_s = (\mathrm{com}(1; 0, 0)\pi_{0,3})^s$ and $\pi_{s,b} = (c_b \pi_{0,4})^s$
4. Using the bilinear map, check $\exists t : \pi_t = \mathrm{com}(1; 0, 0)^t$ and $\pi_{t,b} = c_b^t$
5. Return 1 if all checks pass, else return 0

Simulated proof: $S_{\mathrm{mult}}(\sigma, \tau, (c_a, c_b, c_c))$ runs as follows

[2]Given n pairs (a_i, b_i) and the claim that $\exists r \forall i : b_i = a_i^r$ it can be checked by picking some $a_i \neq 1$ and check that for all $j \neq i$ we have $e(a_i, b_j) = e(b_i, a_j)$. In this particular case, where we claim $\exists r : \pi_r = (\mathrm{com}(1; 0, 0)\pi_{0,1})^r$ and $\pi_{r,b} = (c_b \pi_{0,2})^r$ there are 6 pairs, so we can verify the claim using 10 pairings. Note that since we see an NIZK proof for $\pi_{0,1}$ being a commitment to 0, we know that $\mathrm{com}(1; 0, 0)\pi_{0,1}$ has at least one non-trivial group element, so if the checks work out there is one uniquely defined exponent r. Later on, we will do the same operation on committed elements, where it is not straightforward to check whether some a_i is non-trivial. We solve that by checking $e(a_i, b_j) = e(b_i, a_j)$ for all $1 \leq i < j \leq n$, which corresponds to making 32 pairings.

$r_1, s_1, \ldots, r_4, s_4, r, s, t \leftarrow \mathbb{Z}_p$

$$\begin{array}{ll}
\pi_{0,1} = \mathrm{com}(0; r_1, s_1) & \pi_{0,2} = \mathrm{com}(0; r_2, s_2) \\
\pi_r = (\mathrm{com}(1; 0, 0)\pi_{0,1})^r & \pi_{r,b} = (c_b \pi_{0,2})^r \\
\\
\pi_{0,3} = \mathrm{com}(0; r_3, s_3) & \pi_{0,4} = \mathrm{com}(0; r_4, s_4) \\
\pi_s = (\mathrm{com}(1; 0, 0)\pi_{0,3})^s & \pi_{s,b} = (c_b \pi_{0,4})^s \\
\\
\pi_t = \mathrm{com}(1; 0, 0)^t & \pi_{t,b} = c_b^t
\end{array}$$

$$\begin{array}{l}
\pi_{\pi_{0,1}} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, \pi_{0,1}) \\
\pi_{\pi_{0,2}} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, \pi_{0,2}) \\
\pi_{\pi_{0,3}} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, \pi_{0,3}) \\
\pi_{\pi_{0,4}} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, \pi_{0,4}) \\
\pi_{\pi_{0,5}} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, c_a \pi_r \pi_s \pi_t^{-1}) \\
\pi_{\pi_{0,6}} \leftarrow S_{\mathrm{zero}}(\sigma, \tau, c_c \pi_{r,b} \pi_{s,b} \pi_{t,b}^{-1})
\end{array}$$

The simulated proof is $\pi = (\pi_{0,1}, \ldots, \pi_{\pi_{0,6}})$.

Theorem 8 $(K, P_{\mathrm{mult}}, V_{\mathrm{mult}}, S_1, S_{\mathrm{mult}})$ *is an NIZK proof for* R_{mult} *with perfect completeness, perfect soundness and composable zero-knowledge if the DLIN assumption holds for* $\mathcal{G}$*. A proof consists of 36 group elements. Verification corresponds to evaluating a set of pairing product equations.*

Proof.

Perfect completeness: Follows from the perfect completeness of the NIZK proof for R_{zero}.

Perfect soundness: Suppose $\mathcal{A}$ produces a statement (c_a, c_b, c_c) and a valid NIZK proof π.

From the verification, we learn that there exist r, s, t so

$$c_a(\mathrm{com}(1; 0, 0)\pi_{0,1})^r(\mathrm{com}(1; 0, 0)\pi_{0,3})^s \mathrm{com}(1; 0, 0)^{-t} \quad \text{and} \quad c_c(c_b\pi_{0,2})^r(c_b\pi_{0,4})^s c_b^{-t}$$

are commitments to 0. Since $\pi_{0,1}, \ldots, \pi_{0,4}$ are commitments to 0 this shows that $m_a = t - r - s$ and c_c is a commitment to $m_a m_b$.

Composable zero-knowledge: We wish to argue that $\mathcal{A}(\sigma, \tau)$ who produces a statement (c_a, c_b, c_c) and witness $(m_a, r_a, s_a, m_b, r_b, s_b, r_c, s_c)$ cannot distinguish between a proof π created by P_{mult} and a simulated proof created by S_{mult}.

Let us first look at the way P_{mult} operates. Because $(K, P_{\mathrm{zero}}, V_{\mathrm{zero}}, S_1, S_{\mathrm{zero}})$ is composable zero-knowledge, we can use S_{zero} to simulate $\pi_{\pi_{0,1}}, \ldots, \pi_{\pi_{0,6}}$ instead of using P_{zero} to make them, without $\mathcal{A}$ being able to distinguish.

This partially simulated proof runs exactly as the simulator, except we have $t = r+s+m_a$, whereas in the simulation t is random. We will show that if $\mathcal{A}(\sigma, \tau)$ can distinguish these two cases, then we can break the DLIN assumption for $\mathcal{G}$. Let therefore $g^\alpha, g^\beta, g^{\alpha r}, g^{\beta s}, g^d$ be a DLIN challenge, where we wish to decide whether $d = r + s$ or d is random.

Let us incorporate r, s from the challenge into the proof, such that distinguishing $t = r + s + m_a$ from t random corresponds to distinguishing between $d = r + s$ and d random. Since we know

τ, we know the discrete logarithms of f, h, u, v, w with respect to g. This means, given g^d we can compute

$$\text{com}(1;0,0)^d = ((g^d)^{xr_u}, (g^d)^{yr_v}, (g^d)^{r_u+s_v}).$$

We also know the witness $(m_a, r_a, s_a, m_b, r_b, s_b, r_c, s_c)$ so we can compute

$$c_b^d = ((g^d)^{xr_u m_b + xr_b}, (g^d)^{ys_v m_b + ys_b}, (g^d)^{(r_u+s_v)m_b+(r_b+s_b)}).$$

Given g^α and r_1, s_1 it is straightforward to compute the commitment

$$\text{com}(0;\alpha r_1, \alpha s_1) = ((g^\alpha)^{xr_1}, (g^\alpha)^{ys_1}, (g^\alpha)^{r_1+s_1}).$$

Similarly, we can compute the other elements in the following.

$$r_1, s_1, \ldots, r_4, s_4 \leftarrow \mathbb{Z}_p$$

$$\begin{array}{ll}
\pi_{0,1} = \text{com}(-1;0,0)\text{com}(0;\alpha r_1, \alpha s_1) & \pi_{0,2} = c_b^{-1}\text{com}(0;\alpha r_2, \alpha s_2) \\
\pi_r = \text{com}(0;\alpha r r_1, \alpha r s_1) & \pi_{r,b} = \text{com}(0;\alpha r r_2, \alpha r s_2) \\
 & \\
\pi_{0,3} = \text{com}(-1;0,0)\text{com}(0;\beta r_3, \beta s_3) & \pi_{0,4} = c_b^{-1}\text{com}(0;\beta r_4, \beta s_4) \\
\pi_s = \text{com}(0;\beta s r_3, \beta s s_3) & \pi_{s,b} = \text{com}(0;\beta s r_4, \beta s s_4) \\
 & \\
\pi_t = \text{com}(1;0,0)^d \text{com}(1;0,0)^{m_a} & \pi_{t,b} = c_b^d c_b^{m_a}
\end{array}$$

$$\begin{array}{l}
\pi_{\pi_{0,1}} \leftarrow S_{\text{zero}}(\sigma, \tau, \pi_{0,1}) \\
\pi_{\pi_{0,2}} \leftarrow S_{\text{zero}}(\sigma, \tau, \pi_{0,2}) \\
\pi_{\pi_{0,3}} \leftarrow S_{\text{zero}}(\sigma, \tau, \pi_{0,3}) \\
\pi_{\pi_{0,4}} \leftarrow S_{\text{zero}}(\sigma, \tau, \pi_{0,4}) \\
\pi_{\pi_{0,5}} \leftarrow S_{\text{zero}}(\sigma, \tau, c_a \pi_r \pi_s \pi_t^{-1}) \\
\pi_{\pi_{0,6}} \leftarrow S_{\text{zero}}(\sigma, \tau, c_c \pi_{r,b} \pi_{s,b} \pi_{t,b}^{-1})
\end{array}$$

The distribution of these values is statistically close to what a prover respectively simulator would produce on a simulated common reference string. With $d = r + s$, we have the case with $t = r + s + m_a$, and with d random it corresponds to t random. If $\mathcal{A}$ can distinguish real proofs from simulated proofs, we can therefore distinguish between $d = r + s$ and d random in the DLIN challenge. □

4.7 NIZK Proof for Committed Bilinear Product

Consider elements $a_1, b_1, \ldots, a_n, b_n \in \mathbb{G}$ such that $\prod_{i=1}^n e(a_i, b_i) = 1$. Suppose we have committed to a_i, b_i in the following way. We have $A_i = g^{r_i} a_i, B_i = g^{s_i} b_i$ and commitments to r_i, s_i and wish to make an NIZK proof for $\prod_{i=1}^n e(a_i, b_i) = 1$. More precisely, we want to make an NIZK proof for the following relation: $R_{\text{bil-prod}} = \{(A_1, c_{r_1}, B_1, c_{s_1}, \ldots, A_n, c_{r_n}, B_n, c_{s_n}), (r_1, r_{r_1}, s_{r_1}, s_1, r_{s_1}, s_{s_1}, \ldots, r_n, r_{r_n}, s_{r_n}, s_n, r_{s_n}, s_{s_n}) \mid A_i = g^{r_i} a_i, B_i = g^{s_i} b_i, c_{r_i} = \text{com}(r_i; r_{r_i}, s_{r_i}), c_{s_i} = \text{com}(s_i; r_{s_i}, s_{s_i})$ and $\prod_{i=1}^n e(a_i, b_i) = 1\}$.

For arbitrary $R_1, S_1, \ldots, R_n, S_n \in \mathbb{Z}_p$ we have

$$\prod_{i=1}^n e(A_i, B_i) = \prod_{i=1}^n e(g^{r_i} a_i, g^{s_i} b_i)$$

$$\begin{aligned}
&= (\prod_{i=1}^n e(g^{r_i}, g^{s_i}b_i)e(g^{r_i}a_i, g^{s_i})e(g^{r_i}, g^{s_i})^{-1})\prod_{i=1}^n e(a_i, b_i)\\
&= \prod_{i=1}^n e(g, B_i)^{r_i}e(A_i, g)^{s_i}e(g,g)^{-r_is_i}\\
&= e(g, g^{-\sum_{i=1}^n r_is_i}\prod_{i=1}^n A_i^{s_i}B_i^{r_i})\\
&= e(g, g^{-\sum_{i=1}^n (r_is_i+R_iS_i)}\prod_{i=1}^n A_i^{s_i}B_i^{r_i})\prod_{i=1}^n e(g^{R_i}, g^{S_i}),
\end{aligned}$$

if and only if $\prod_{i=1}^n e(a_i, b_i) = 1$.

In the NIZK proof, we pick $R_1, S_1, \ldots, R_n, S_n$ at random, their role is to give us zero-knowledge. We commit to R_i, S_i and we already have commitments to r_i, s_i. We reveal the $2n+1$ elements $g^{R_1}, g^{S_1}, \ldots, g^{R_n}, g^{S_n}$ and $g^{-\sum_{i=1}^n (r_is_i+R_iS_i)}\prod_{i=1}^n A_i^{s_i}B_i^{r_i}$. We then use NIZK proofs for $R_{\text{expo}}, R_{\text{mult}}, R_{\text{m-ped}}$ to prove that these elements have been formed correctly.

In the simulation, we observe that for arbitrary $R_1, S_1, \ldots, R_n, S_n$ we have

$$\begin{aligned}
\prod_{i=1}^n e(A_i, B_i) &= e(g,1)\prod_{i=1}^n e(A_i, B_i)\\
&= e(g, g^{-\sum_{i=1}^n R_iS_i}\prod_{i=1}^n A_i^{-S_i}B_i^{-R_i})\prod_{i=1}^n e(g^{R_i}A_i, g^{S_i}B_i).
\end{aligned}$$

Picking $R_1, S_1, \ldots, R_n, S_n$ at random means that all elements have the same distribution as in a real proof. We can then simulate the NIZK proofs for $R_{\text{expo}}, R_{\text{mult}}, R_{\text{m-ped}}$.

Proof for committed bilinear product: $P_{\text{bil-prod}}(\sigma, (A_1, c_{r_1}, B_1, c_{s_1}, \ldots, A_n, c_{r_n}, B_n, c_{s_n}), (r_1, r_{r_1}, s_{r_1}, s_1, r_{s_1}, s_{s_1}, \ldots, r_n, r_{r_n}, s_{r_n}, s_n, r_{s_n}, s_{s_n}))$ does the following

```
 1. For $i = 1$ to $n$ do
 2.     $R_i, S_i, r_{R_i}, s_{R_i}, r_{S_i}, s_{S_i}, r_{R_iS_i}, s_{R_iS_i}, r_{r_is_i}, s_{r_is_i} \leftarrow \mathbb{Z}_p$
 3.     $\pi_{R_i} = \text{com}(R_i; r_{R_i}, s_{R_i}), \pi_{S_i} = \text{com}(S_i; r_{S_i}, s_{S_i})$
 4.     $\pi_{R_iS_i} = \text{com}(R_iS_i; r_{R_iS_i}, s_{R_iS_i})$
 5.     $\pi_{g^{R_i}} = g^{R_i}, \pi_{g^{S_i}} = g^{S_i}$
 6.     $\pi_{r_is_i} = \text{com}(r_is_i; r_{r_is_i}, s_{r_is_i})$
 7. $\pi_{\text{m-ped}} = g^{-\sum_{i=1}^n (r_is_i+R_iS_i)}\prod_{i=1}^n A_i^{s_i}B_i^{r_i}$
 8. For $i = 1$ to $n$ do
 9.     $\pi_{\pi_{R_i}} \leftarrow P_{\text{expo}}(\sigma, (g, \pi_{g^{R_i}}, \pi_{R_i}), (R_i, r_{R_i}, s_{R_i}))$
10.     $\pi_{\pi_{S_i}} \leftarrow P_{\text{expo}}(\sigma, (g, \pi_{g^{S_i}}, \pi_{S_i}), (S_i, r_{S_i}, s_{S_i}))$
11.     $\pi_{\pi_{R_iS_i}} \leftarrow P_{\text{mult}}(\sigma, (\pi_{R_i}, \pi_{S_i}, \pi_{R_iS_i}), (R_i, r_{R_i}, s_{R_i}, S_i, r_{S_i}, s_{S_i}, r_{R_iS_i}, s_{R_iS_i}))$
12.     $\pi_{\pi_{r_is_i}} \leftarrow P_{\text{mult}}(\sigma, (c_{r_i}, c_{s_i}, \pi_{r_is_i}), (r_i, r_{r_i}, s_{r_i}, s_i, r_{s_i}, s_{s_i}, r_{r_is_i}, s_{r_is_i}))$
13. $\pi_{\pi_{\text{m-ped}}} \leftarrow P_{\text{m-ped}}(\sigma, (A_1, B_1, \ldots, A_n, B_n, \pi_{\pi_{\text{m-ped}}}, (\prod_{i=1}^n c_{r_is_i}c_{R_iS_i})^{-1}, c_{s_1}, c_{r_1}, \ldots, c_{s_n}, c_{r_n}),$
14.                 $(-\sum_{i=1}^n (r_is_i + R_iS_i), -\sum_{i=1}^n (r_{r_is_i} + r_{R_iS_i}),$
15.                 $-\sum_{i=1}^n (s_{r_is_i} + s_{R_iS_i}), s_1, r_{s_1}, s_{s_1}, \ldots, r_n, r_{r_n}, s_{r_n}))$
```

The proof is $\pi = (\pi_{R_1}, \dots, \pi_{\pi_{\mathrm{m-ped}}})$.

Verification: $V_{\mathrm{bil-prod}}(\sigma, (A_1, c_{r_1}, B_1, c_{s_1}, \dots, A_n, c_{r_n}, B_n, c_{s_n}), \pi)$ does the following

```
1. Check $\prod_{i=1}^n e(A_i, B_i) = e(g, \pi_{\mathrm{m-ped}}) \prod_{i=1}^n e(\pi_{g^{R_i}}, \pi_{g^{S_i}})$
2. Verify the proofs $\pi_{\pi_{R_1}}, \dots, \pi_{\pi_{\mathrm{m-ped}}}$
3. Return 1 if all checks pass, else return 0
```

Simulation: $S_{\mathrm{bil-prod}}(\sigma, \tau, (A_1, c_{r_1}, B_1, c_{s_1}, \dots, A_n, c_{r_n}, B_n, c_{s_n}))$ does the following

```
1.  For $i = 1$ to $n$ do
2.      $R_i, S_i, r_{R_i}, s_{R_i}, r_{S_i}, s_{S_i}, r_{R_iS_i}, s_{R_iS_i}, r_{r_is_i}, s_{r_is_i} \leftarrow \mathbb{Z}_p$
3.      $\pi_{R_i} = \mathrm{com}(0; r_{R_i}, s_{R_i}), \pi_{S_i} = \mathrm{com}(0; r_{S_i}, s_{S_i})$
4.      $\pi_{R_iS_i} = \mathrm{com}(0; r_{R_iS_i}, s_{R_iS_i})$
5.      $\pi_{g^{R_i}} = g^{R_i} A_i, \pi_{g^{S_i}} = g^{S_i} B_i$
6.      $\pi_{r_is_i} = \mathrm{com}(0; r_{r_is_i}, s_{r_is_i})$
7.  $\pi_{\mathrm{m-ped}} = g^{-\sum_{i=1}^n R_iS_i} \prod_{i=1}^n A_i^{-S_i} B_i^{-R_i}$
8.  For $i = 1$ to $n$ do
9.      $\pi_{\pi_{R_i}} \leftarrow S_{\mathrm{expo}}(\sigma, \tau, (g, \pi_{g^{R_i}}, \pi_{R_i}))$
10.     $\pi_{\pi_{S_i}} \leftarrow S_{\mathrm{expo}}(\sigma, \tau, (g, \pi_{g^{S_i}}, \pi_{S_i}))$
11.     $\pi_{\pi_{R_iS_i}} \leftarrow S_{\mathrm{mult}}(\sigma, \tau, (\pi_{R_i}, \pi_{S_i}, \pi_{R_iS_i}))$
12.     $\pi_{\pi_{r_is_i}} \leftarrow S_{\mathrm{mult}}(\sigma, \tau, (c_{r_i}, c_{s_i}, \pi_{r_is_i}))$
13. $\pi_{\pi_{\mathrm{m-ped}}} \leftarrow S_{\mathrm{m-ped}}(\sigma, \tau, (A_1, B_1, \dots, A_n, B_n, \pi_{\pi_{\mathrm{m-ped}}}, (\prod_{i=1}^n c_{r_is_i} c_{R_iS_i})^{-1}, c_{s_1}, c_{r_1}, \dots, c_{s_n}, c_{r_n}))$
```

The simulated proof is $\pi = (\pi_{R_1}, \dots, \pi_{\pi_{\mathrm{m-ped}}})$.

Theorem 9 $(K, P_{\mathrm{bil-prod}}, V_{\mathrm{bil-prod}}, S_1, S_{\mathrm{bil-prod}})$ *is an NIZK proof for* $R_{\mathrm{bil-prod}}$ *with perfect completeness, perfect soundness and composable zero-knowledge under the DLIN assumption for* $\mathcal{G}$*. Proofs consist of* $228n - 3$ *group elements and verification corresponds to evaluating a set of pairing product equations.*

Proof.

Perfect completeness: Perfect completeness follows from the perfect completeness of $P_{\mathrm{mult}}, P_{\mathrm{expo}}, P_{\mathrm{m-ped}}$.

Perfect soundness: From $\pi_{\pi_{R_i}}, \pi_{\pi_{S_i}}$ we learn that there exists R_i, S_i so $\pi_{g^{R_i}} = g^{R_i}, \pi_{g^{S_i}} = g^{S_i}$ and π_{R_i}, π_{S_i} are commitments to those R_i, S_i.

From $\pi_{\pi_{R_iS_i}}$ we learn that $\pi_{R_iS_i}$ is a commitment to R_iS_i. Likewise, $\pi_{\pi_{r_is_i}}$ is an NIZK proof that $\pi_{r_is_i}$ contains r_is_i, the product of the messages in c_{r_i}, c_{s_i}.

The proof $\pi_{\pi_{\mathrm{m-ped}}}$ shows that $\pi_{\mathrm{m-ped}} = g^{-\sum_{i=1}^n (r_is_i + R_iS_i)} \prod_{i=1}^n A_i^{s_i} B_i^{r_i}$. Since

$$\prod_{i=1}^n e(A_i, B_i) = e(g, \pi_{\mathrm{m-ped}}) \prod_{i=1}^n e(\pi_{g^{R_i}} \pi_{g^{S_i}}) = e(g, g^{-\sum_{i=1}^n (r_is_i + R_iS_i)} \prod_{i=1}^n A_i^{s_i} B_i^{r_i}) \prod_{i=1}^n e(g^{R_i}, g^{S_i}),$$

we then have $\prod_{i=1}^n e(a_i, b_i) = 1$.

Composable zero-knowledge: Because, the NIZK proofs for $R_{\text{mult}}, R_{\text{expo}}$ and R_{ped} are all composable zero-knowledge, we may modify the prover in such a way that we simulate all the NIZK proofs for these relations. The adversary $\mathcal{A}(\sigma, \tau)$ cannot distinguish such a partially simulated proof from a real one.

Since the commitment scheme is perfectly hiding, we can now make commitments to 0, whenever we make a commitment. This is perfectly indistinguishable from the partially simulated proof described above. The partially simulated proof contains random values $\pi_{g^{R_i}}, \pi_{g^{S_i}}$ and $\pi_{\text{m-ped}}$ is the unique value so $\prod_{i=1}^n e(A_i, B_i) = e(g, \pi_{\text{m-ped}}) \prod_{i=1}^n e(\pi_{g^{R_i}}, \pi_{g^{S_i}})$. In the simulation, we also get completely random elements $\pi_{g^{R_i}}, \pi_{g^{S_i}}$ and the unique element $\pi_{\text{m-ped}}$ such that the equation holds. Therefore, proofs and simulated proofs are computationally indistinguishable. □

4.8 NIZK Proof for Satisfiability of Pairing Product Equations

In this section, we consider sets of pairing product equations over variables $a_1, \ldots, a_n$. Let us first recall the definition in the introduction of a pairing product equation. By a pairing product equation, we mean an equation on the form

$$eq(a_1, \ldots, a_n) : \prod_{j=1}^{\ell} e(q_{j,0}, q_{j,1}) = 1 \text{ , where } q_{j,b} = b_{j,b} \prod_{i=1}^{n} a_i^{e_{j,b,i}},$$

for known $b_{j,b} \in \mathbb{G}$ and $e_{j,b,i} \in \mathbb{Z}_p$. A set S of pairing product equations $eq_1, \ldots, eq_m$ is said to be satisfiable if there exists $(a_1, \ldots, a_n) \in \mathbb{G}^n$ such that all equations are satisfied.

Let $R_{\text{ppsat}} = \{ S \mid \exists (a_1, \ldots, a_n) \in \mathbb{G}^n \; \forall eq_k \in S : eq_k(a_1, \ldots, a_n) = \texttt{true}\}$. Using the NIZK proof for $R_{\text{bil-prod}}$ we can create an NIZK proof for R_{ppsat}. The idea is straightforward, we first commit to each a_i as $g^{t_i} a_i, \text{com}(t_i)$. Using homomorphic properties, it is straightforward for $q_{k,j,b}$ in equation eq_k to compute $g^{t_{k,j,b}} b_{k,j,b} \prod_{i=1}^n a_i^{e_{k,j,b,i}}, \text{com}(t_{k,j,b})$ as

$$b_{k,j,b} \prod_{i=1}^{n} (g^{t_i} a_i)^{e_{k,j,b,i}} = g^{\sum_{i=1}^n t_i e_{k,j,b,i}} (b_{k,j,b} \prod_{i=1}^{n} a_i^{e_{k,j,b,i}}) \text{ , } \prod_{i=1}^{n} \text{com}(t_i)^{e_{k,j,b,i}} = \text{com}(\sum_{i=1}^{n} t_i e_{k,j,b,i}).$$

For each equation we can now carry out an NIZK proof for $R_{\text{bil-prod}}$ that $\prod_{j=1}^{\ell_k} e(q_{k,j,0}, q_{k,j,1}) = 1$.

Proof for satisfiability of pairing product equations: On a set of pairing product equations $eq_1, \ldots, eq_m$ and witness $a_1, \ldots, a_n$ where eq_k is the equation $\prod_{j=1}^{\ell_k} e(q_{k,j,0}(a_1, \ldots, a_n), q_{k,j,1}(a_1, \ldots, a_n)) = 1$ with terms $q_{k,j,b}(a_1, \ldots, a_n) = b_{k,j,b} \prod_{i=1}^n a_i^{e_{k,j,b,i}}$ do

```
1. For $i = 1$ to $n$ do
2.     $t_i, r_i, s_i \leftarrow \mathbb{Z}_p$
3.     $\pi_{a_i} = g^{t_i} a_i$
4.     $\pi_{t_i} = \text{com}(t_i; r_i, s_i)$
5. For $k = 1$ to $m$ do
6.     $\pi_k \leftarrow P_{\text{bil-prod}}(\sigma, (b_{k,1,0} \prod_{i=1}^n \pi_{a_i}^{e_{k,1,0,i}}, \prod_{i=1}^n \pi_{t_i}^{e_{k,1,0,i}}, \ldots, b_{k,\ell_k,1} \prod_{i=1}^n \pi_{a_i}^{e_{k,\ell_k,1,i}},$
7.             $\prod_{i=1}^n \pi_{t_i}^{e_{k,\ell_k,1,i}}), (\sum_{i=1}^n t_i e_{k,1,0,i}, \sum_{i=1}^n r_i e_{k,1,0,i}, \sum_{i=1}^n s_i e_{k,1,0,i}, \ldots,$
8.             $\sum_{i=1}^n t_i e_{k,\ell_k,1,i}, \sum_{i=1}^n r_i e_{k,\ell_k,1,i}, \sum_{i=1}^n s_i e_{k,\ell_k,1,i}))$
```

The proof is $\pi = (\pi_{a_1}, \pi_{t_1}, \dots, \pi_{a_n}, \pi_{t_n}, \pi_1, \dots, \pi_m)$

Verification: Given the statement and proof π return 1 if and only if all proofs $\pi_1, \dots, \pi_m$ are valid.

Simulation: To simulate a proof do the following

```
1. For $i = 1$ to $n$ do
2.     $t_i, r_i, s_i \leftarrow \mathbb{Z}_p$
3.     $\pi_{a_i} = g^{t_i}$
4.     $\pi_{t_i} = \mathrm{com}(0; r_i, s_i)$
5. For $k = 1$ to $m$ do
6.     $\pi_k \leftarrow S_{\mathrm{bil-prod}}(\sigma, \tau, (b_{k,1,0} \prod_{i=1}^n \pi_{a_i}^{e_{k,1,0,i}}, \prod_{i=1}^n \pi_{t_i}^{e_{k,1,0,i}}, \dots, b_{k,\ell_k,1} \prod_{i=1}^n \pi_{a_i}^{e_{k,\ell_k,1,i}},$
7.             $\prod_{i=1}^n \pi_{t_i}^{e_{k,\ell_k,1,i}}))$
```

The simulated proof is $\pi = (\pi_{a_1}, \pi_{t_1}, \dots, \pi_{a_n}, \pi_{t_n}, \pi_1, \dots, \pi_m)$.

Theorem 10 $(K, P_{\mathrm{ppsat}}, V_{\mathrm{ppsat}}, S_1, S_{\mathrm{ppsat}})$ *is an NIZK proof for* R_{ppsat} *with perfect completeness, perfect soundness and composable zero-knowledge if the DLIN assumption holds for* $\mathcal{G}$. *Proofs consist of* $4n + 228\ell - 3m$ *group elements, where* $\ell = \sum_{k=1}^m \ell_k$. *Verification consists of evaluating a set of pairing product equations.*

Proof.

Perfect completeness: Perfect completeness follows from the perfect completeness of the NIZK proofs for $R_{\mathrm{bil-prod}}$.

Perfect soundness: Since the commitments are perfectly binding, π_{t_i} contain a t_i that uniquely defines an a_i so $\pi_{a_i} = g^{t_i} a_i$. Since the NIZK proofs for $R_{\mathrm{bil-prod}}$ are perfectly sound, $a_1, \dots, a_n$ is a satisfying assignment to the equations $eq_1, \dots, eq_m$.

Composable zero-knowledge: On a simulated reference string, it is indistinguishable to the adversary whether it sees proofs or simulated proofs $\pi_1, \dots, \pi_m$. Since the commitment scheme is perfectly hiding on a simulated reference string, this implies simulation indistinguishability. □

NESTING NIZK PROOFS. It is interesting to observe that verification of an NIZK proof for R_{ppsat} itself consists of verifying a set of pairing product equations. This means that we can nest NIZK proofs, i.e., prove that there exists a proof such that there exists a proof, etc. Each level of nesting will cause a blowup by a constant factor. This is something that is much more expensive to do with other known NIZK proofs, and impossible to do in the random oracle model.

REDUCING THE NUMBER OF VARIABLES. We will argue that if $n > 2\ell$ then we can combine or remove some of the variables such that we get an equivalent set of equations over variables $a'_1, \dots, a'_{2\ell}$.

We have a total of 2ℓ monomials $q_{k,j,b}/b_{k,j,b}$. We can represent each as a row-vector $(e_{k,j,b,1}, \dots, e_{k,j,b,n})$. Let M be an $2\ell \times n$ matrix with these row-vectors. Let $\vec{a} = (\log a_1, \dots, \log a_n)$ and $\vec{q} = (\log(q_{1,1,0}/b_{1,1,0}), \dots, \log(q_{m,\ell_m,1}/b_{m,\ell_m,1}))$. We have $M\vec{a} = \vec{q}$. Since $n > 2\ell$, we can do column-reduction on M to get a matrix with at most 2ℓ non-zero columns. This means there exists an invertible $n \times n$ transformation matrix T such that only the 2ℓ left columns of MT are non-zero. We have $(MT)(T^{-1}\vec{a}) = \vec{q}$. Let $\vec{a}$ correspond to a satisfying assignment for the set of equations. Then $\vec{a'} = T^{-1}\vec{a}$ corresponds to a satisfying assignment for the set of pairing product equations given by MT. Note that given $a_1, \ldots, a_n$ it is straightforward to compute $a'_1, \ldots, a'_n$. On the other hand, since T is invertible if there is no $(a_1, \ldots, a_n)$ satisfying the equations, then there is no $(a'_1, \ldots, a'_n)$ satisfying the equations given by representing the $q_{k,j,b}$'s with the rows in MT. We now observe that since only the left 2ℓ columns of MT are non-zero, we can ignore $a'_{2\ell+1}, \ldots, a'_n$ when considering satisfiability of the equations. Therefore, in practice we can always get by with 2ℓ variables and the NIZK proof will consist of $\mathcal{O}(\ell)$ group elements.

PAIRING PRODUCT EQUATIONS FOR IDENTICAL PLAINTEXT. Here is another motivational example for being interested in satisfiability of pairing product equations that will be useful later on. Let (f_1, h_1) and (f_2, h_2) be two public keys for the cryptosystem described in Section 1.1. Let (u_1, v_1, w_1) and (u_2, v_2, w_2) be two ciphertexts. We are interested in the question whether they encrypt the same message m. The ciphertexts uniquely define m_1, r_1, s_1 and m_2, r_2, s_2 such that

$$u_1 = f_1^{r_1}, v_2 = h_1^{s_1}, w_1 = g^{r_1+s_1} m_1 \qquad \text{and} \qquad u_2 = f_2^{r_2}, v_2 = h_2^{s_2}, w_2 = g^{r_2+s_2} m_2.$$

Observe, $m_1 = m_2$ if and only if $w_1/w_2 = g^{r_1+s_1-r_2-s_2}$. Therefore, $a_1 = g^{r_1}, a_2 = g^{s_1}, a_3 = g^{r_2}, a_4 = g^{s_2}$ is a witness for the two ciphertexts encrypting the same message. It is straightforward to use the bilinear map to check the pairing product equations

$$e(g, u_1) = e(a_1, f_1), \qquad e(g, v_1) = e(a_2, h_1), \qquad e(g, u_2) = e(a_3, f_2),$$
$$e(g, v_2) = e(a_4, h_2) \qquad \text{and} \qquad e(g, a_1 a_2 a_3 a_4) = e(g, w_1 w_2^{-1}),$$

where the latter equation implies $w_1/w_2 = g^{r_1+s_1-r_2-s_2}$.

DISJUNCTION OF SETS OF PAIRING PRODUCT EQUATIONS. Suppose that we have L sets $S_1, \ldots, S_L$ of pairing product equations over $a_1, \ldots, a_n$ and want to argue that at least one of the sets is satisfiable. One can compile these sets of equations into one set S, which is satisfiable if and only if one of $S_1, \ldots, S_L$ is satisfiable. The compilation, which we describe below is witness preserving in the sense that a satisfying assignment for a set S_l allows one to compute a satisfying assignment for S easily.

We introduce variables $a_{l,k,j}$, where $l \leq L, k \leq m_l, j \leq \ell_{l,k}$ and also $A_1, \ldots, A_L$. The role of $A_1, \ldots, A_L$ is to point to some set that is satisfiable. S will contain the equation $e(g^{-1} \prod_{l=1}^{L} A_l, g) = 1$. This equation guarantees at least one $A_l \neq 1$, indicating the satisfiable set. The prover can choose this $A_l = g$, while the others can be 1. We add to S the set equations $e(A_l, a_{l,k,j}^{-1} b_{l,k,j,0} \prod_{i=1}^{n} a_i^{e_{l,k,j,0}}) = 1$. These equations ensure that if $A_l \neq 1$ then $a_{l,k,j} = q_{l,k,j,0}$ and if $A_l = 1$, then we are free to choose $a_{l,k,j} = 1$. Finally, add all the original equations from the sets $S_1, \ldots, S_L$ replacing $q_{l,k,j,0}$ with $a_{l,k,j}$. This construction works because for l, where $A_l = g^0$ we have that all the variables representing $q_{k,l,j,0}$'s are 1 so the equations in this set are satisfied. On the other hand, for the set S_l where $A_l \neq 1$ we have preserved the original equations. If the sets of pairing product equations $S_1, \ldots, S_L$ have combined length $\ell = \sum_{l=1}^{L} \sum_{k=1}^{m_l} \ell_{l,k}$, then the set $\mathcal{S}$ has length $\ell_S = 1 + 2\ell$ and contains pairing product equations over $n + L + \ell$ variables.

5 Cryptographic Tools

5.1 A One-time Signature Scheme

Suppose we want to make a one-time signature on an element $m \in \mathbb{G}$. The verification key will consist of a common reference string σ as well as two commitments c, c_1 to respectively z, z_1. A signature on m is $m^z g^{z_1}$ and an NIZK proof that it has been correctly formed. The intuition behind this signature scheme is that even if an adversary sees one signature, there are still two unknowns z, z_1 so he cannot determine what a signature on another element should look like. We will extend this scheme in the natural way to sign multiple elements.

Verification key: On input n and $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ do

```
1. $\sigma \leftarrow K(p, \mathbb{G}, \mathbb{G}, e, g)$
2. $z, z_1, \ldots, z_n \leftarrow \mathbb{Z}_p$
3. $r_z, s_z, \ldots, r_{z_n}, s_{z_n} \leftarrow \mathbb{Z}_p$
4. $c = \mathrm{com}(z; r_z, s_z), \ldots, c_n = \mathrm{com}(z_n; r_{z_n}, s_{z_n})$
```

The verification key is $vk = (\sigma, c, c_1, \ldots, c_n)$.

The signing key is $sk = (vk, z, r_z, s_z, \ldots, z_n, r_{z_n}, s_{z_n})$

Signature: To sign $(m_1, \ldots, m_n) \in \mathbb{G}^n$ do

```
1. For $i = 1$ to $n$ do
2.     $s_i = m_i^z g^{z_i}$
3.     $\pi_i \leftarrow P_{\mathrm{ped}}(\sigma, (m_i, s_i, c, c_i), (z, r_z, s_z, z_i, r_{z_i}, s_{z_i}))$
```

The signature is $s = (s_1, \pi_1, \ldots, s_n, \pi_n)$

Verification: To verify the signature s on message $(m_1, \ldots, m_n)$ check the proofs $\pi_1, \ldots, \pi_n$.

Theorem 11 *Assuming $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ the scheme $(K_{\mathrm{ots}}, \mathrm{Sign}, \mathrm{Ver})$ described above is a one-time signature scheme with perfect correctness. To sign n elements from $\mathbb{G}$ both the verification key and the signatures have $\mathcal{O}(n)$ group elements. We note that the verification procedure consists of evaluating a set of pairing product equations.*

Proof. Perfect correctness follows from the perfect completeness of the NIZK proof for R_{ped}.

To argue existential unforgeability under a one-time chosen message attack consider the probability of an adversary creating a forged signature on $m_1, \ldots, m_n$ after seeing a signature on $m'_1, \ldots, m'_n$. By the perfect soundness of the NIZK proof, the adversary must produce a signature on the form $s_1 = m_1^z g^{z_1}, \ldots, s_n = m_n^z g^{z_n}$.

Let us change the key generation such that we simulate the common reference string and we simulate the NIZK proofs on the adversary's chosen message attack. A successful adversary must with overwhelming probability still produce a signature on $m_1, \ldots, m_n$ where $s_i = m_i^z g^{z_i}$. Otherwise, we could break the unbounded zero-knowledge property of the NIZK proof with the knowledge of $z, z_1, \ldots, z_n$.

We are now in the simulation case, where the NIZK proofs are simulated. There are $n + 1$ unknowns $z, z_1, \ldots, z_n$ and in the chosen message attack the adversary may learn n equations $(m'_1)^z g^{z_1}, \ldots, (m'_n)^z g^{z_n}$. This still leaves one unknown variable. The probability of the adversary producing the correct $m_1^z g^{z_1}, \ldots, m_n^z g^{z_n}$ is therefore negligible unless $m_1 = m'_1, \ldots, m_n = m'_n$. □

5.2 RCCA-secure Public-Key Encryption

Canetti, Krawczyk and Nielsen [CKN03] suggest a useful relaxation of chosen ciphertext attack security. Informally, their notion captures the case where an adversary may be able to rerandomize a ciphertext such that it still has the *same* plaintext, however, in all other cases the cryptosystem is CCA-secure. They call this security against replayable chosen ciphertext (RCCA) attack.

Definition 12 (RCCA-security) *A cryptosystem $(K_{\mathrm{rcca}}, E, D)$ is RCCA-secure if for any non-uniform polynomial time adversary $\mathcal{A}$ we have*

$$\begin{aligned} & P[(pk, sk) \leftarrow K_{\mathrm{rcca}}(1^k); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_1}(pk); c \leftarrow E_{pk}(m_0) : \mathcal{A}^{\mathcal{O}_2}(c) = 1] \\ \approx\ & P[(pk, sk) \leftarrow K_{\mathrm{rcca}}(1^k); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_1}(pk); c \leftarrow E_{pk}(m_1) : \mathcal{A}^{\mathcal{O}_2}(c) = 1], \end{aligned}$$

where

- $\mathcal{O}_1(\cdot) = D_{sk}(\cdot)$.
- $\mathcal{O}_2(\cdot) = D_{sk}(\cdot)$ *except when the plaintext is* m_0 *or* m_1. *On plaintext* m_0 *or* m_1 *the oracle outputs* `test`.

It is clear that any CCA-secure cryptosystem is also RCCA-secure. [CKN03] show a separation, if CCA-secure cryptosystems exist then RCCA-secure cryptosystems that are not CCA-secure exist. On the other hand, it is also the case that if RCCA-secure cryptosystems exist, then CCA-secure cryptosystems exist.

We will suggest a public key cryptosystem, which is RCCA-secure. The construction resembles the constructions of CCA-secure encryption by Lindell [Lin03], who builds on previous work by Naor and Yung [NY90] and Sahai [Sah01]. Since we only aim for RCCA-security, we can obtain a few simplifications though. We first present the general construction and then plug in our tools afterwards.

Public key generation: To generate keys do

1. Generate two keys, $(pk_1, sk_1), (pk_2, sk_2) \leftarrow K_{\mathrm{cpa}}(1^k)$, for a CPA-secure cryptosystem
2. Generate a key, $ck \leftarrow K_{\mathrm{binding}}(1^k)$, for a perfectly binding commitment scheme
3. Generate a key $(vk', sk') \leftarrow K_{\mathrm{ots}}(1^k)$ for a one-time signature scheme
4. Let $c_{vk} = \mathrm{com}_{ck}(vk'; r)$ be a commitment to vk' using randomness r
5. Generate a common reference string, $\sigma \leftarrow K(1^k)$, for an NIZK proof[3]

The public key is $pk = (pk_1, pk_2, ck, c_{vk}, \sigma)$ and the secret key is $sk = (pk, sk_1)$.

Encryption: To encrypt a message m.

1. Pick randomness r_1, r_2 and encrypt m twice as $c_1 = E_{pk_1}(m; r_1), c_2 = E_{pk_2}(m; r_2)$
2. Select a key $(vk, sk) \leftarrow K_{\mathrm{ots}}(1^k)$ for the one-time signature scheme
3. Sign c_1, c_2 as $s \leftarrow \mathrm{Sign}_{sk}(c_1, c_2)$
4. $\pi \leftarrow P_{\mathrm{sor}}(\sigma, (pk_1, pk_2, ck, c_{vk}, c_1, c_2, vk), (m, r_1, r_2))$. This is an NIZK proof for c_1, c_2 containing the same plaintext or c_{vk} containing vk, i.e., $vk' = vk$.

The ciphertext is $c = (c_1, c_2, vk, s, \pi)$

Decryption: Verify the signature s and the proof π. If both are ok, return $m = D_{sk_1}(c_1)$.

Theorem 13 *The cryptosystem described above is RCCA-secure.*

[3]Zero-knowledge implies witness-indistinguishability, which is sufficient here.

Proof. In either experiment in the definition of RCCA-security if the adversary ever recycles the one-time signature verification key of the challenge in a decryption query with a valid signature s and a valid proof π, then with overwhelming probability it must also recycle c_1, c_2, since otherwise we would have a one-time signature forgery. We can therefore with only negligible change in success-probability for the adversary let the oracle answer `test` on any query with a valid (by valid we mean correct signature and correct proof) ciphertext reusing vk from the challenge.

When making the challenge encryption we can pick vk' as our one-time signature verification key. The hiding property of the commitment scheme ensures that the adversary's success probability only changes negligibly. In particular, the adversary does not use vk' in any valid query to $\mathcal{O}_1$, and in case it uses it in a query to $\mathcal{O}_2$, then it recycles c_1, c_2 from the challenge and we answer `test`.

We can now switch the witness we use in making the proof π. Instead of using (m_b, r_1, r_2) as the witness, we can use vk', r as the witness. The zero-knowledge property ensures that this does not change the adversary's success probability significantly.

We now have a situation where the adversary does not ask valid decryption queries using vk', and in the challenge we encrypt the message under both pk_1, pk_2 and then make an NIZK proof using witness vk', r. We have to argue that in this setting the adversary cannot distinguish a challenge encryption of m_0 from a challenge encryption of m_1.

Since the decryption key sk_2 is never used, the semantic security under chosen plaintext attack gives us that we can change c_2 to be an encryption of m_1 without the adversary noticing it.

Next, let us argue that we can switch from using sk_1 to using sk_2 when decrypting oracle queries. Remember, we already modified the oracle so that if vk' is recycled in a valid query, then we answer `test`, and we don't need either decryption key. Consider any other decryption query, here $vk \neq vk'$. Soundness of the NIZK proof implies that c_1, c_2 in any valid query contain the same message. Using sk_1 or sk_2 therefore gives the same answer, and therefore the adversary cannot distinguish whether we use sk_1 or sk_2 for decryption.

We are now in a situation, where the decryption key sk_1 is never used. Semantic security under chosen plaintext attack therefore means that we can switch the plaintext of c_1 from m_0 to m_1. We now have that in the challenge we encrypt m_1 in both c_1 and c_2.

Since the soundness of the NIZK proof implies that in any valid query c_1, c_2 contain the same plaintext, we can switch back to using sk_1 for decryption. $\square$

Let us suggest a concrete implementation of the above-mentioned scheme. We work over a DLIN group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ and wish to encrypt messages on the form $m = (m_1, \dots, m_n) \in \mathbb{G}^n$.

We pick $x_1, y_1, x_2, y_2 \leftarrow \mathbb{Z}_p$ at random and use $f_1 = g^{x_1}, h_1 = g^{y_1}$ and $f_2 = g^{x_2}, h_2 = g^{x_2}$ as two public keys for the CPA-secure cryptosystem described in Section 1.1. To encrypt a message $m = (m_1, \dots, m_n) \in \mathbb{G}^n$ under either key, we simply encrypt $m_1, \dots, m_n$ one by one.

To set up a perfectly binding commitment scheme, we pick yet another encryption key f_c, h_c. Commitment corresponds to encryption under this key.

We want to use the one-time signature scheme from the previous section. Let us consider how long the ciphertexts c_1, c_2 can be. Suppose we want to encrypt tuples of messages $(m_1, \dots, m_n) \in \mathbb{G}^n$. The size of two ciphertexts will be $6n$ elements in $\mathbb{G}$, so we will set up the one-time signature scheme such that we can sign messages consisting of $6n$ group elements. Such a one-time verification key will specify a common reference string $\sigma_{vk} = (f_{vk}, h_{vk}, u_{vk}, v_{vk}, w_{vk})$ as well as the committed signing key $c, c_1, \dots, c_{6n}$. We will therefore let c_{vk} be the encryption of these $18n + 9$ elements.

Finally, we need an NIZK proof for $R_{\text{sor}} = \{((pk_1, pk_1, ck, c_{vk}, c_1, c_2, vk), w) \mid (w = (r_1, r_2, m) : c_1 = E_{pk_1}(m; r_1), c_2 = E_{pk_2}(m; r_2)) \vee (w = r : c_{vk} = \text{com}_{ck}(vk; r))\}$. We have in Appendix 4.8 argued that there exists a set of pairing product equations that are satisfiable if and only if two ciphertexts encrypt the same message under their respective public keys. In the introduction, we described a set of pairing product equations corresponding to encryption of a particular message. In our case, this ciphertext is c_{vk}, which has the plaintext $vk' = (f_{vk'}, \ldots, c'_n)$. Finally, we also argued in Appendix 4.8 that we can compile two sets of pairing product equations into one set corresponding to the statement that at least one of the original set of equations is satisfiable. We observe, given a witness in the form of plaintexts and randomness for c_1, c_2 or vk' and randomness for c_{vk} one can actually find $(a_1, a_2, \ldots)$ to satisfy the set of pairing product equations. This shows that we can use the NIZK proof for $R_{\rm ppsat}$ to implement an NIZK proof for $R_{\rm sor}$. With the choices above, the size of the NIZK proof will be $\mathcal{O}(n)$ group elements.

Corollary 14 *If $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ is a DLIN-group, we can built an RCCA-secure cryptosystem $(K_{\rm rcca}, E, D)$. The cryptosystem permits encryption of messages on the form $(m_1, \ldots, m_n) \in \mathbb{G}^n$. The cryptosystem has perfect decryption and perfect decryption verification as defined in the next section. The public key and the ciphertexts both consist of $\mathcal{O}(n)$ group elements. It is publicly verifiable whether the ciphertext is valid, and such verification corresponds to evaluating a set of pairing product equations.*

Kiltz [Kil06] has recently suggested a simple CCA2-secure cryptosystem based on the DLIN assumption. However, in his scheme verifying correctness of the ciphertext does not correspond to evaluating a set of pairing product equations. Since we will need this property later on, we use the more complicated construction above.

5.3 Signature Scheme Secure against Chosen Message Attack

In this section, we construct a signature scheme that is secure against chosen message attack. We first describe the signature scheme in general terms, then suggest a concrete implementation based on the DLIN assumption.

The signature scheme based on the DLIN assumption is not practical due to the large constant. Nonetheless, it does have independent interest since it is based on a simple cryptographic assumption and can be used to sign group elements $m \in \mathbb{G}$. Other signature schemes such as Boneh and Boyen's [BB04] needs the message to be an exponent $m \in \mathbb{Z}_p$, which makes it hard to make proofs of knowledge for instance since we do not know how to compute discrete logarithms.

We will need three tools in the construction. One tool is an RCCA-secure cryptosystem[4] with decryption verification. By decryption verification, we mean that there should exist two algorithms $W_{\rm dec}, V_{\rm dec}$. The role of $W_{\rm dec}$ is to convert the plaintext m and the randomness r for a ciphertext $c = E_{pk}(m; r)$ into a decryption witness w for $D_{sk}(c) = m$. The role of $V_{\rm dec}$ is to verify a witness w that indeed a ciphertext will decrypt to m. Formally, we require that for all adversaries $\mathcal{A}$ we have

$$\Pr\Big[(pk, sk) \leftarrow K_{\rm rcca}(1^k); (m, r) \leftarrow \mathcal{A}(pk, sk); c = E_{pk}(m; r); w \leftarrow W_{\rm dec}(pk, m, r) : V_{\rm dec}(pk, m, c, w) = 1\Big]$$

and

$$\Pr\Big[(pk, sk) \leftarrow K_{\rm rcca}(1^k); (m, c, w) \leftarrow \mathcal{A}(pk, sk) : V_{\rm dec}(pk, m, c, w) = 1 \text{ and } D_{sk}(c) \neq m\Big] = 0.$$

Decryption verification implies that the cryptosystem has perfect decryption. On the other hand, for a cryptosystem with perfect decryption, we could simply let $w = r$ and let the verification algorithm check whether $c = E_{pk}(m; w)$.

[4]Actually, we do not need full RCCA-security. It suffices that the cryptosystem is RPA0-secure [Gro03]. In an RPA0-attack the adversary does not have access to $\mathcal{O}_1$. It can only query $\mathcal{O}_2$ once, but in this one query is allowed to ask for decryption of many ciphertexts. It can be shown with techniques from [BS99] that this kind of attack corresponds to a notion of non-malleability, where the adversary may be able to modify a ciphertext into one that contains the same message, but cannot maul the ciphertext in a non-trivial way.

The second tool is a gap problem. By this we mean that we have a generator K_{gap} that generates (a,b) and a verification algorithm V_{gap} such that it is easy to verify (a,b) but hard to compute b from a. For all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Pr\left[(a,b)\leftarrow K_{\text{gap}}(1^k):V_{\text{gap}}(a,b)=1\right]=1 \text{ and } \Pr\left[(a,b)\leftarrow K_{\text{gap}}(1^k);b'\leftarrow\mathcal{A}(a):V_{\text{gap}}(a,b')=1\right]\approx 0.$$

It is easy to come up with examples of gap-problems, one could for instance let f be a one-way function, choose b at random and let $a=f(b)$, which is easily verifiable.

The third tool is an NIZK proof system for a ciphertext decrypting to m,b such that b is the hard part of a gap-problem. More precisely, we need a $R_{\text{encgap}}=\{((pk,a,m,c),(b,w))\mid V_{\text{dec}}(pk,(m,b),c,w)=1,V_{\text{gap}}(a,b)=1\}$.

Key generation: To generate keys do

1. Pick $(a,b)\leftarrow K_{\text{gap}}(1^k)$
2. Generate keys for the RCCA-secure cryptosystem, $(pk,sk_{\text{rcca}})\leftarrow K_{\text{rcca}}(1^k)$
3. Generate a common reference string for the NIZK proof system, $\sigma\leftarrow K(1^k)$

The verification key is $vk=(pk,a,\sigma)$.

The signing key is $sk=(vk,b)$.

Signing: To sign a message m do

1. Encrypt m,b as $c=E_{pk}(m,b;r)$
2. Let $w=W_{\text{dec}}(pk,(m,b),r)$
3. Make an NIZK proof $\pi\leftarrow P_{\text{encgap}}(\sigma,(pk,a,m,c),(b,w))$ for c decrypting to m and b such that $V_{\text{gap}}(a,b)=1$.

The signature is $s=(c,\pi)$.

Verification: To verify signature s check the proof π

Theorem 15 *The signature scheme described above is existentially unforgeable under chosen message attack.*

Proof. Let $\mathcal{A}^{\text{Sign}_{sk}(\cdot)}(vk)$ be an adversary with access to a signing oracle and a randomly generated verification key vk. We wish to argue that it has negligible probability of finding a valid signature on a message m that it has not queried the oracle.

Observe first that by the soundness of the NIZK proof and the decryption verifiability property the ciphertext in $\mathcal{A}$'s forgery must contain m,b' so $V(a,b')=1$. We can therefore modify the game such that we decrypt the resulting ciphertext in $\mathcal{A}$'s forgery and consider the adversary unsuccessful if such m,b' is not the plaintext.

Let us modify the game such that we create $(\sigma,\tau)\leftarrow S_1(1^k)$ and simulate the NIZK proofs in the signing oracle. By the unbounded zero-knowledge property of the NIZK proof, the adversary cannot distinguish between this game and the previous one, so the success probability is changed negligibly. In particular, it must still produce a forgery where the plaintext is m,b' to be successful.

Let us make another modification, instead of encrypting m,b when making signatures, we encrypt $m,1$. To argue that this does not change the success probability of the adversary we will make a hybrid argument, so let $\text{Sign}[i]_{sk}(\cdot)$ be an oracle that on queries $1,\ldots,q$ responds with a signature where it encrypts $m_j, 1$ on a query m_j, where $j < i$ and encrypts m_j, b if $j \geq i$. Let $q(k)$ be an upper bound on the number of signing queries $\mathcal{A}$ makes. Since $\mathcal{A}$ is a polynomial time adversary, $q(k)$ is polynomial. We now have Sign$_{sk}(\cdot)$ =Sign$[0]_{sk}(\cdot)$, and we are switching to use the signing oracle Sign$[q(k)]_{sk}(\cdot)$ where we encrypt $(\cdot, 1)$.

What we need to observe is that for all $0 \leq i < q(k)$ the adversary's success probability using oracle Sign$[i]_{sk}(\cdot)$ is almost the same as when using the oracle Sign$[i+1]_{sk}(\cdot)$. Let us see that RCCA-security of the cryptosystem implies this property. Given a public key pk for the RCCA-security we choose $(a, b) \leftarrow K_{\text{gap}}(1^k)$ and we choose $(\sigma, \tau) \leftarrow S_1(1^k)$ and give the adversary the verification key $vk = (pk, a, \sigma)$. We answer queries $1, \ldots, i-1$ by encrypting $m_j, 1$ and simulating π. Now $\mathcal{A}$ produces query m_i and we let our two challenge messages be $m_0 = (m_i, 1)$ and $m_1 = (m_i, b)$. We receive a challenge encryption of one of these messages, and our goal is to break RCCA-security by distinguishing which one we received. We answer query i by simulating the proof and returning the challenge ciphertext together with the simulated proof. In all future signing queries, we encrypt (m_j, b) and simulate the proofs. Notice, if the challenge encryption has the plaintext $m_i, 1$, then this corresponds exactly to the adversary running with oracle Sign$[i]_{sk}$, while if the challenge encryption has plaintext m_i, b, then it corresponds exactly to the adversary running with oracle Sign$[i+1]_{sk}$. The adversary now produces a forged signature c, π on message m. We give c to the decryption oracle and receive a response m', b'. In case $m' = m, V_{\text{gap}}(a, b') = 1$ we output 1, else we output 0. In case the response is `test` we also output 0. By definition, to be successful $\mathcal{A}$ had to produce a ciphertext where m was inside (i.e., it is never successful if the answer is `test`) and $V_{\text{gap}}(a, b') = 1$. Therefore, if $\mathcal{A}$ has more than negligible difference in success probability with respectively oracle Sign$[i]_{sk}$ and Sign$[i+1]_{sk}$, then we can break the RCCA-security of the cryptosystem.

To conclude, observe that now we have a game where the adversary sees encryptions of $m_i, 1$ on query i, yet to be successful has to produce an encryption of m, b' such that $V_{\text{gap}}(a, b') = 1$. Since (a, b) is a gap problem, the adversary has negligible success probability. □

We can instantiate the general signature scheme above using a DLIN group. The gap problem will be the following: We pick d at random and let $a = g^d, b = g^{d^2}$. It is straightforward to verify the correctness of such a pair by checking whether $e(g, b) = e(a, a)$. Lemma 16 states that given a it is hard to compute b.

Lemma 16 *If the DLIN assumption holds for $\mathcal{G}$ then for all non-uniform polynomial time adversaries $\mathcal{A}$ we have*

$$\Pr\left[(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k); d \leftarrow \mathbb{Z}_p; b \leftarrow \mathcal{A}(p, \mathbb{G}, \mathbb{G}_1, e, g, a = g^d) : b = g^{d^2}\right] \approx 0.$$

Proof. If we can solve the computational Diffie-Hellman problem with more than negligible probability then we can break DLIN. In other words, given g, g^α, g^β for random $\alpha, \beta \leftarrow \mathbb{Z}_p$ it is hard to compute $g^{\alpha\beta}$.

Assume that we have more than negligible chance of computing g^{d^2} in the game above. This means, given g^α, g^β we have more than negligible chance of computing $g^{(\alpha+\beta)^2}, g^{(\alpha-\beta)^2}$. Rewriting $g^{\alpha\beta} = g^{((\alpha+beta)^2-(\alpha-\beta)^2)4^{-1}}$ we see that this violates the hardness of the computational Diffie-Hellman problem. □

To make a signature on $m = (m_1, \ldots, m_n) \in \mathbb{G}^n$ we encrypt $m_1, \ldots, m_n, b$ with the RCCA-secure cryptosystem constructed in the previous section. This encryption consists of $\mathcal{O}(n)$ group elements.

We need to make an NIZK proof that indeed the ciphertext contains the correct $m_1, \ldots, m_n$ and b such that $e(g, b) = e(a, a)$. Recall from the last section that the RCCA-secure encryption contains as a part of it a CPA-secure ciphertext c_1 which encrypts $m_1, \ldots, m_n, b$ as

$(f_1^{r_{1,1}}, h_1^{s_{1,1}}, g^{r_{1,1}+s_{1,1}}m_1, \ldots, f_1^{r_{1,n+1}}, h_1^{s_{1,n+1}}, g^{r_{1,n+1}+s_{1,n+1}}b)$. If the ciphertext is valid, which is publicly verifiable, the decryption operation gives us $m_1, \ldots, m_n, b$. The RCCA-secure cryptosystem therefore has simple decryption verification where the witness is on the form $g^{r_1}, \ldots, g^{r_{n+1}}$. All we have to do now is to check whether indeed $e(a,a) = e(g,b)$. All equations we have to evaluate are pairing product equations. This means that we can build an NIZK proof for R_{encgap} of length $\mathcal{O}(n)$.

Corollary 17 *Under the DLIN assumption there exists a CMA-secure digital signature scheme* $(K_{\text{sign}}, \text{Sign}, \text{Ver})$ *for signing* n *group elements with perfect correctness. The verification key and the signatures consist of* $\mathcal{O}(n)$ *group elements and the verification process consists of evaluating a set of pairing product equations.*

5.4 Strong One-Time Signature

The idea for our strong one-time signature scheme is to set up a Pedersen trapdoor commitment to 0. When we receive a message $m \in \mathbb{Z}_p$ to be signed, we make a trapdoor opening of the commitment to m. Only the signer knows the trapdoor, so only he can sign messages. Security of this scheme comes from the hardness of computing discrete logarithms. Note, the hardness of the discrete logarithm problem is implied by the DLIN assumption.

We want the signature scheme to be secure against a one-time chosen message attack. Therefore, we set up the Pedersen commitment with two trapdoors such that it is impossible for the adversary to see which trapdoor we used. This way, if the adversary can forge a signature we can find a way to break one of the trapdoors, i.e., compute a discrete logarithm.

Key generation: On the bilinear group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ we generate the verification key and the signing key, as follows. We choose $x_s, y_s \leftarrow \mathbb{Z}_p^*$ and set $f_s = g^{x_s}, h_s = g^{y_s}$. We pick $r_s, s_s \leftarrow \mathbb{Z}_p$ and set $c_s = f_s^{r_s} h_s^{s_s}$. We pick a collision-free hash function $H : \{0,1\}^* \to \mathbb{Z}_p$. The verification key is $vk = (p, \mathbb{G}, \mathbb{G}_1, e, g, f_s, h_s, c_s, H)$ and the secret key is $sk = (vk, x_s, y_s)$.

Signature: To sign a message $m \in \{0,1\}^*$ pick $r \leftarrow \mathbb{Z}_p$ and reveal the signature $(r, (x_s(r_s - r) + y_s s_s - H(m))/y_s)$.

Verification: To verify a signature (r, s) on m we check that $c_s = g^{H(m)} f_s^r h_s^s$.

Theorem 18 *Assuming hardness of computing discrete logarithms and collision-freeness of the hash-function, the protocol* $(K_{\text{sots}}, \text{Sign}, \text{Ver})$ *described above is a strong one-time signature scheme for signing messages* $m \in \{0,1\}^*$ *with perfect correctness.*

Proof. Let us say the adversary $\mathcal{A}$ queries for a signature (r, s) on m and then forges a signature (r', s') on m', where $s \neq s'$ with more than negligible probability. We will use it to compute a discrete logarithm or break the collision-freeness of the hash-function. Let $h_s = g^{y_s}$ be a challenge for the DL problem, we wish to compute y_s so $h_s = g^{y_s}$. We choose $x_s \leftarrow \mathbb{Z}_p^*$ (we can also easily check that $y_s \neq 0$) and let $f_s = g^{x_s}$. We then form $c_s = f_s^{r_s} h^s$ for $r_s, s \leftarrow \mathbb{Z}_p$. On query m we return the signature $(r, s) = (r_s - H(m)/x_s, s)$. This looks exactly like a real verification key and a standard signature. $\mathcal{A}$ therefore produces a forgery $m', (r', s')$ with $s' \neq s$ with more than negligible probability. We see now that $c = g^H(m) f_s^r h_s^s = g^{H(m')} f_s^{r'} h_s^{s'}$ and therefore $h_s = g^{(H(m') + x_s r' - H(m) - x_s r)/(s - s')}$, so we have computed the discrete logarithm of h_s. In a similar fashion, we can argue that $\mathcal{A}$ must reuse $r' = r$. But if $r' = r, s' = s$ the only possibility is $H(m') = H(m)$ and $\mathcal{A}$ has found a collision if $m' \neq m$. $\square$

Since hardness of computing discrete logarithms implies the existence of collision-free hash-functions this is not really an extra assumption.

6 Simulation-Sound NIZK Proof of Knowledge for Satisfiability of Pairing Product Equations

In this section, we combine the tools of the previous section with the NIZK proofs to construct an unbounded simulation-sound extractable NIZK proof for satisfiability of pairing product equations. We first describe the idea in general terms of proving a statement x belongs to some language L.

The prover will pick random keys $(vk_{\mathrm{sots}}, sk_{\mathrm{sots}})$ for a strong one-time signature scheme, and vk_{sots} will be part of the NIZK proof. Another part of the NIZK proof will be a strong one-time signature on the statement to be proven and the NIZK proof. Since the adversary does not know the secret signing key associated with vk_{sots} he must pick a different verification key in his forged NIZK proof.

The common reference string will contain a verification key vk for a digital signature scheme that is secure against adaptive chosen message attack. The prover will prove that $x \in L$ *or* that he knows a signature on vk_{sots}. In simulations we can set it up such that we do know the secret signing key associated with vk. This allows us to sign any vk_{sots} and therefore make a convincing NIZK proof even though we do not know a witness for $x \in L$. On the other hand, the adversary even after seeing many such proofs will not be able to forge a signature on a new vk_{sots} and therefore he cannot make a valid NIZK proof for a false statement.

Obviously, we need to hide the digital signature on vk_{sots}, otherwise it would be easy to see whether an NIZK proof was real or simulated. To do this, we let the common reference string contain a public key for a CPA-secure cryptosystem. In a real proof we encrypt some dummy message, while in the simulation we encrypt a digital signature on vk_{sots}. We then proceed with an NIZK proof that $x \in L$ or the ciphertext contains a signature on vk_{sots}.

One small issue remains. A computationally unbounded adversary can of course forge signatures under vk and therefore prove false statements. To have perfect soundness, we will include an encryption c_1 of some non-trivial element in the common reference string. Furthermore, in the NIZK proofs we require an NIZK proof for the prover having both encrypted a digital signature on vk_{sots} as well as c_1 having plaintext 1. Since c_1 does not have plaintext 1 even an unbounded prover cannot cheat. On the other hand, in the simulation we will set up c_1 so that it does contain 1.

To make the NIZK proof a proof of knowledge, instead of proving directly that $x \in L$, we will encrypt a witness and prove that we have encrypted the witness, or we have encrypted a signature on vk_{sots} and c_1 has 1 as plaintext.

Let in the following $(K, P_{\mathrm{ssor}}, V_{\mathrm{ssor}}, S_1, S_{\mathrm{ssor}})$ be an NIZK proof for R_{ssor}, the relation for the statement that c_w contains a satisfying $(a_1, \ldots, a_n)$ or c_s contains a signature on vk_{sots} and c_1 contains 1. More formally, $R_{\mathrm{ssor}} = \{((S, c_w, c_s), w) \mid (w = (a_1, \ldots, a_n, R_w) : c = E_{(f_e,h_e)}(a_1, \ldots, a_n; R_w), S(a_1, \ldots, a_n) = \mathtt{true}) \vee (w = (s, R_s, r_c, s_c) : c_s = E_{(f_e,h_e)}(s; R_s), \mathrm{Ver}_{vk}(vk_{\mathrm{sots}}, s) = 1, c_1 = E_{(f_e,h_e)}(1; r_c, s_c))\}$.

Common reference string generation and simulation: On group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ do

1. $(vk, sk) \leftarrow K_{\mathrm{sign}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$
2. $x_e, y_e \leftarrow \mathbb{Z}_p^*$
3. $f_e = g^{x_e}, h_e = g^{y_e}$
4. $r_c, s_c \leftarrow \mathbb{Z}_p$
5. $c_1 = E_{(f_e,h_e)}(g; r_c, s_c) = (f_e^{r_c}, h_e^{s_c}, g^{r_c + s_c g})$
6. $\sigma \leftarrow K(p, \mathbb{G}, \mathbb{G}_1, e, g)$

The common reference string is $\Sigma = (vk, f_e, h_e, c_1, \sigma)$

The extraction algorithm E_1 generates the common reference string as above and outputs the extraction key $\xi = (x_e, y_e)$

The simulation-extractor SE_1 generates the common reference string as described above except it sets $c_1 = E_{(f_e,h_e)}(1; r_c, s_c)$. It outputs $(\Sigma, \tau, \xi) = ((vk, f_e, h_e, c_1, \sigma), (sk, r_c, s_c), (x_e, y_e))$.

Proof of satisfiability of pairing product equations: On a set S of pairing product equations over n variables and a witness, namely a satisfying assignment $(a_1, \ldots, a_n)$, do

1. $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow K_{\text{sots}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$
2. $c_w = E_{(f_e,h_e)}(a_1, \ldots, a_n; R_w)$, where $R_w = (r_{w,1}, s_{w,1}, \ldots, r_{w,n}, s_{w,n})$
3. $c_s = E_{(f_e,h_e)}(1, \ldots, 1; R_s)$, where $R_s = (r_{s,1}, \ldots)$
4. $\pi_{\text{ssor}} \leftarrow P_{\text{ssor}}(\sigma, (S, c_w, c_s), (a_1, \ldots, a_n, R_w))$
5. $s_{\text{sots}} \leftarrow \text{Sign}_{sk_{\text{sots}}}(S, vk_{\text{sots}}, c_w, c_s, \pi_{\text{ssor}})$

The proof is $\pi = (vk_{\text{sots}}, c_w, c_s, \pi_{\text{ssor}}, s_{\text{sots}})$

Extraction: Given a valid proof π as above, the extraction algorithm uses $\xi = (x_e, y_e)$ to decrypt c_w to get a witness $a_1, \ldots, a_n$.

Verification: To verify proof π check $\text{Ver}_{vk_{\text{sots}}}((S, vk_{\text{sots}}, c_w, c_s, \pi_{\text{ssor}}), s_{\text{sots}}) = 1$ and that the ciphertexts c_w, c_s have the right lengths and that the proof π_{ssor} is valid.

Simulation: To simulate a proof for S do

1. $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow K_{\text{sots}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$
2. $s \leftarrow \text{Sign}_{sk}(vk_{\text{sots}})$
3. $c_w = E_{(f_e,h_e)}(1, \ldots, 1; R_w)$, where $R_w = (r_{w,1}, s_{w,1}, \ldots, r_{w,n}, s_{w,n})$
4. $c_s = E_{(f_e,h_e)}(s; R_s)$, where $R_s = (r_{s,1}, s_{s,1}, \ldots)$
5. $\pi_{\text{ssor}} \leftarrow P_{\text{ssor}}(\sigma, (S, c_w, c_s), (s, R_s, r_c, s_c))$
6. $s_{\text{sots}} \leftarrow \text{Sign}_{sk_{\text{sots}}}(S, vk_{\text{sots}}, c_w, c_s, \pi)$

The simulated proof is $\pi = (vk_{\text{sots}}, c_w, c_s, \pi_{\text{ssor}}, s_{\text{sots}})$

Theorem 19 *If $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ is a DLIN group then $(K_{\text{sse}}, P_{\text{sse}}, V_{\text{sse}}, S_{1,\text{sse}}, S_{\text{sse}}, E_{1,\text{sse}}, E_{\text{sse}}, SE_{1,\text{sse}})$ as described above is an NIZK proof for satisfiability of pairing product equations. It has perfect completeness, perfect soundness, perfect knowledge extraction and composable zero-knowledge and unbounded simulation soundness extraction. The size of the common reference string is $\mathcal{O}(1)$ group elements, while the NIZK proofs consist of $\mathcal{O}(n + \ell)$ group elements.*

Proof. Let us first compute the size of the common reference string and the NIZK proofs. vk_{sots} consists of $\mathcal{O}(1)$ group elements, no matter the size of the message to be signed. We use (vk, sk) when signing vk_{sots} in the simulation, but since vk_{sots} has constant size, we can make do with a constant size vk, and in simulations the signature s will also have constant size. Since the public key (f_e, h_e) has 2 group elements, and since σ is of constant size as well, we see that the entire common reference string Σ has $\mathcal{O}(1)$ group elements. In a simulated proof, the ciphertext c_s only need to encrypt the constant size signature s. On the other hand in an NIZK proof, c_w needs to encrypt a witness containing n elements, so it will have size $\mathcal{O}(n)$ group elements. Setting up product pairing equations for a plaintext $a_1, \ldots, a_n$ being inside uses $\mathcal{O}(n)$ group elements. With the combined length of the pairing products equations being ℓ the proof π_{ssor} therefore consists of $\mathcal{O}(n + \ell)$ group elements. Since s_{sots} is of constant size, we conclude that the NIZK proof has size $\mathcal{O}(n + \ell)$ group elements.

Perfect completeness: It follows from the perfect completeness of the NIZK proof for R_{ppsat} that we get perfectly complete NIZK proof for R_{ssor}. This combined with the perfect decryption property of the cryptosystem and the perfect correctness of the signature schemes gives us perfect completeness.

Perfect soundness and perfect knowledge extraction: From the perfect soundness of the NIZK proof for R_{ssor} we know that either c_w encrypts a satisfying $(a_1, \ldots, a_n)$ or that c_1 is an encryption of 1. Since c_1 is not an encryption of 1 this means that c_w encrypts a satisfying $a_1, \ldots, a_n$. This implies that S is satisfiable. Moreover, given the decryption key (x_e, y_e) we can extract the witness $(a_1, \ldots, a_n) = D_{(x_e,y_e)}(c_w)$.

Composable zero-knowledge: We first have to argue common reference string indistinguishability. The only difference between common reference strings and simulated common reference strings is whether c_1 is an encryption of 1 or not. By the semantic security of the cryptosystem no non-uniform polynomial time adversary can distinguish between such ciphertexts with more than negligible probability, and therefore it cannot distinguish between real and simulated common reference strings.

Let now Σ be a simulated common reference string and consider a non-uniform polynomial time adversary $\mathcal{A}(\Sigma, \tau)$ that tries to distinguish between proofs and simulated proofs. It produces a set S of pairing product equations as well as a satisfiability witness $(a_1, \ldots, a_n)$. Given a proof π it has to distinguish whether π was simulated or not.

Let us start with the way the prover creates a proof π. By the semantic security of the cryptosystem, we can create a signature $s \leftarrow \text{Sign}_{sk}(vk_{\text{sots}})$ and let $c_s = E_{(f_e,h_e)}(s; R_s)$ instead of encrypting 1's without changing $\mathcal{A}$'s success probability more than negligibly.

The proof π_{ssor} proves that c_w contains a satisfying $(a_1, \ldots, a_n)$, or c_1 encrypts 1 and c_s contains a signature on vk_{sots}. Now both parts of this or-statement is true and we know a witness for both of them. Since π_{ssor} is a zero-knowledge proof it is also witness-indistinguishable. We can therefore switch to using the witness (s, R_s, r_c, s_c) in the proof π_{ssor} without $\mathcal{A}$ detecting the switch.

By the semantic security of the cryptosystem $\mathcal{A}$ does not notice it if we switch to creating c_w as an encryption of $(1, \ldots, 1)$. We are now creating the proof as the simulator does, but $\mathcal{A}$'s success probability has changed only negligibly.

Simulation-sound extractability: We will argue that it is infeasible for a non-uniform polynomial time adversary $\mathcal{A}^{S_2(\Sigma,\tau,\cdot)}(\Sigma, \xi)$ to create a statement and valid proof (x, π) such that we cannot extract a witness from it, unless π is one of the query-responses.

We first observe that the strong one-time signature scheme's existential unforgeability implies that it is infeasible for $\mathcal{A}$ to produce a valid proof π where it recycles vk_{sots} from one of the queries.

If c_w does not contain a satisfying $(a_1, \ldots, a_n)$ then by the soundness of the NIZK proof for R_{ssor} the ciphertext c_s must contain a signature s on vk_{sots}. Using the decryption key (x_e, y_e) we therefore obtain a forged signature under vk. By the CMA-security of the signature scheme this event has negligible probability of happening.

□

6.1 Universally Composable Non-interactive Zero-Knowledge

The goal of this section is to demonstrate how powerful NIZK proofs with simulation-sound extractability are. We will securely realize the NIZK-functionality $\mathcal{F}_{\text{NIZK}}$ from [GOS06b] in Canetti's UC framework

[Can01]. In [GOS06b] there is a construction of a UC NIZK protocol that can be used to realize $\mathcal{F}_{\mathrm{NIZK}}$ for Circuit Satisfiability in a model where the adversary is adaptive and the parties cannot erase data from their tapes. We consider a weaker model, where the parties *can* erase data from their tapes. On the other hand, whereas the UC NIZK protocol in [GOS06b] is inefficient, our protocol produces proofs of linear size in the circuit size. We note that [CLOS02] have already observed without proof that simulation-sound extractability gives you UC NIZK for non-adaptive adversaries, which in the case of UC NIZK is almost the same as adaptive adversaries where we allow erasures.

MODELING NON-INTERACTIVE ZERO-KNOWLEDGE PROOFS We refer to [Can01, GOS06b] for a description of the UC-framework and the modeling of NIZK proofs/arguments in the UC framework. Here we simply describe the ideal functionalities for Circuit Satisfiability and prove that we can realize it with linear size proofs under the DLIN assumption for $\mathcal{G}$.

Parametrized by relation R and running with parties $P_1, \dots, P_n$ 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 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 proof functionality $\mathcal{F}_{\mathrm{NIZK}}$.

Common reference string: On input (**start**,sid) run $\Sigma \leftarrow K(1^k)$.

Send (**crs**,sid, Σ) to all parties and halt.

Figure 4: Protocol for UC NIZK common reference string generation.

Proof: Party P waits until receiving (**crs**,sid, Σ) from $\mathcal{F}_{\mathrm{CRS}}$.

On input (**prove**,$sid, ssid, x, w$) run $\pi \leftarrow P(\Sigma, x, w)$. Erase intermediate data used in the computation of π. Output (**proof**,$sid, ssid, \pi$).

Verification: Party V waits until receiving (**crs**,sid, Σ) from $\mathcal{F}_{\mathrm{CRS}}$.

On input (**verify**,$sid, ssid, x, \pi$) run $b \leftarrow V(\Sigma, x, \pi)$. Output (**verification**,$sid, ssid, b$).

Figure 5: Protocol for UC NIZK proof using simulation-sound extractable NIZK proof $(K, P, V, S_1, S_2, E_1, E_2, SE_1)$ for relation R.

Theorem 20 *The protocol in Figure 5 securely realizes $\mathcal{F}_{\mathrm{NIZK}}$ in the $\mathcal{F}_{\mathrm{CRS}}$-model.*

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

$\mathcal{S}$ 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 $\mathcal{F}_{\text{CRS}}$. $\mathcal{S}$ chooses the common reference string as $(\Sigma, \tau, \xi) \leftarrow SE_1(1^k)$. $\mathcal{S}$ simulates $\mathcal{F}_{\text{CRS}}$ 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**,$sid, ssid, x$) from $\mathcal{F}_{\text{NIZK}}$. This means that some dummy party $\tilde{P}$ received input (**prove**,$sid, ssid, x, w$), where $(x, w) \in R$. We must simulate the output a real party P would make, however, we may not know w.

We create $\pi \leftarrow S_2(\Sigma, \tau, x)$ and return (**proof**,π) to $\mathcal{F}_{\text{NIZK}}$. $\mathcal{F}_{\text{NIZK}}$ 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**,x, π) from $\mathcal{F}_{\text{NIZK}}$. This means an honest dummy party $\tilde{V}$ has received (**verify**,$sid, ssid, x, \pi$) from the environment.

$\mathcal{S}$ checks the proof, $b \leftarrow V(\Sigma, x, \pi)$. If invalid, it sends (**witness**,`no witness`) to $\mathcal{F}_{\text{NIZK}}$ 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 x 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 $\mathcal{S}$ lets $w \leftarrow E_2(\Sigma, \xi, x, \pi)$. If $(x, w) \notin R$ it sets $w =$ `no witness`. It sends (**witness**,w) to $\mathcal{F}_{\text{NIZK}}$. It delivers the resulting output message to $\tilde{V}$ that outputs it to the environment.

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

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 x that it has proved together with the corresponding witnesses w. Recall, the UC NIZK arguments π have been provided by $\mathcal{S}$. Since we erased all other data, we can therefore simulate the tape of P_i.

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 $\mathcal{F}_{\text{CRS}}$-hybrid model and the ideal adversary $\mathcal{S}$ running in the $\mathcal{F}_{\text{NIZK}}$-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 $\mathcal{F}_{\text{CRS}}$-hybrid model running with adversary $\mathcal{A}$ and parties $P_1, \ldots, P_n$.

H1: We modify H0 by running $(\Sigma, \tau, \xi) \leftarrow SE_1(1^k)$ and creating the proofs of uncorrupted provers as $\pi \leftarrow S_2(\Sigma, \tau, x)$.

By the unbounded zero-knowledge property the adversary this experiment is indistinguishable from H0.

H2: Consider the case where an honest party V receives (**verify**,$sid, ssid, x, \pi$). Suppose π is indeed an acceptable UC NIZK proof and π is not one of the proofs we simulated. We run $w \leftarrow E_2(\Sigma, \xi, x, \pi)$. If $(x, w) \notin R$ give up in the simulation.

By the simulation-sound extractability property there is negligible probability that we will ever give up, so H2 is indistinguishable from H1.

H3: This is the ideal process running with $\mathcal{F}_{\text{NIZK}}$ and $\mathcal{S}$.

Inspection shows that in process H2 and H3 we are computing the different parts of the protocol in the same way. H2 and H3 are therefore perfectly indistinguishable to the environment.

□

Corollary 21 *If the DLIN assumption for $\mathcal{G}$ is true then there exists a UC NIZK proof for Circuit Satisfiability secure against adaptive adversaries where we allow erasures. The common reference string contains a constant number of group elements, while the proofs consist of $\mathcal{O}(|C|)$ group elements.*

7 Constant Size Group Signatures without Random Oracles

7.1 Group Signature Functionality

In a group signature scheme there is a group manager that controls the group. This group manager controls who can join the group. Once in the group members can sign messages on behalf of the group. Members' signatures are anonymous except to the group manager who can open a signature and see who signed the message. In some scenarios it is of interest to separate the group manager into two entities, an issuer who enrolls members and an opener who traces signers.

We will describe the algorithms the group signature scheme will support. We imagine that there is a PKI in place so that public keys can be trusted. We model this by having a public key registry reg where only user i has a one-time write access to $reg[i]$, we do not attempt to keep this information secret. User i's stores his secret key in $gsk[i]$, unless compromised only the user has access to this key. [BSZ05] model the PKI in a slightly more complicated way, but the difference between their definition and the present one is non-essential.

Key generation: GKg generates (gpk, ik, ok). Here gpk is a group public key, while ik and ok are respectively the issuer's and the opener's secret key.

Join/Issue: This is an interactive protocol between a user and the issuer. The user i registers a public key vk_i in $reg[i]$ and stores some corresponding private information sk_i. The issuer on detecting a new entry $reg[i]$ uses the issuer key ik to generate a response $cert_i$. The user verifies the correctness of the response, and in case it accepts it stores $gsk[i] = (sk_i, vk_i, cert_i)$.

The [BSZ05] definition allows for many rounds of secret communication. Our protocol is secure in this more restricted model where we have a simple 2-move interaction, which does not need to be secret.

Sign: A group member i can sign a message m by running picking randomness r and letting the signature be $s = \text{Gsig}(gpk, gsk[i], m; r)$.

Verify: To verify a signature s on message m we run $\text{GVf}(gpk, m, s)$. The signature is valid if and only if this verification algorithm outputs 1.

Open: The opener has read-access to the registration table reg. We have $(i, \psi) \leftarrow \text{Open}(gpk, ok, reg, m, s)$ gives an opening of a valid signature s on message m pointing to i. In case the signature points to no member, the opener will assume the issuer forged the signature and set $i = \texttt{issuer}$.

Judge: This algorithm is used to verify that openings are correct. We say the opening is correct if $\text{Judge}(gpk, i, reg[i], m, s, \psi) = 1$.

7.2 Group Signature Security Definitions

[BSZ05] define four security properties that the group signature must satisfy: correctness, anonymity, traceability and non-frameability. We refer to [BSZ05] for a discussion how this security definition covers and strengthens other security issues that have appeared in the literature.

PERFECT CORRECTNESS. On any adversarially chosen message, the verification should accept a group signature created with a correctly generated group signing key $gsk[i]$ for member i. Running the opening algorithm on this should identify i and make the Judge algorithm accept the opening. For all (unbounded) adversaries $\mathcal{A}$ we have

$$\Pr\Big[(gpk, ik, ok) \leftarrow \mathrm{GKg}(1^k); (i, m) \leftarrow \mathcal{A}^{\text{Join/Issue}}(gpk, ik, ok); s \leftarrow \mathrm{GSig}(gpk, gsk[i], m);\\ (j, \psi) \leftarrow \mathrm{Open}(gpk, ok, reg[], m, s) :\\ \text{if } i \in Q \text{ then } F = 0 \ \wedge\ i = j \ \wedge\ \mathrm{Judge}(gpk, i, reg[i], m, s, \psi) = 1\Big] = 1,$$

where the oracle works as follows

Join/Issue: On input i that has not been queried before run the Join/Issue protocol. This updates $reg[i]$ and $gsk[i]$. In case the user does not accept, set $F = 1$ and return 1, else set $F = 0$ and return $reg[i], gsk[i]$. Add i to the list of queries Q.

ANONYMITY. It should be infeasible for an adversary to identify the signer of a message if he does not know the opener's key ok. We require a strong sense of anonymity, which holds even when the adversary controls the issuer and that all the members' secret signing keys are exposed. We require for all non-uniform polynomial time $\mathcal{A}$ that

$$\Pr\Big[(gpk, ik, ok) \leftarrow \mathrm{GKg}(1^k) : \mathcal{A}^{\text{Ch}_0,\text{Open},\text{Issue},\text{ReadGsk},\text{JoinCorrupt},\text{JoinExposedHonest}}(gpk, ik) = 1\Big]\\ \approx \ \Pr\Big[(gpk, ik, ok) \leftarrow \mathrm{GKg}(1^k) : \mathcal{A}^{\text{Ch}_1,\text{Open},\text{Issue},\text{ReadGsk},\text{JoinCorrupt},\text{JoinExposedHonest}}(gpk, ik) = 1\Big]$$

where the oracles work as follows:

JoinExposedHonest: On input i where $reg[i]$ is empty first generate (vk_i, sk_i) as specified by the Join-algorithm. Then store $reg[i] \leftarrow vk_i$ and send (sk_i, vk_i) to the adversary. Add i to Q_{Join}.

Issue: On input $(i, cert_i)$ where $i \in Q_{\text{Join}}$ and yet not been answered check whether the answer is acceptable. In that case, store $gsk[i] = (sk_i, vk_i, cert_i)$.

JoinCorrupt: On input (i, vk_i) where $reg[i]$ is empty $reg[i] = vk_i$. This allows the adversary to enroll a corrupt member and register any public key of its own choosing.

Ch$_b$: On input (i_0, i_1, m) where i_0, i_1 are honest members with non-empty $gsk[i_0]$ and $gsk[i_1]$ return $s \leftarrow \mathrm{GSig}(gpk, gsk[i_b], m)$.

Open: On input (m, s) that has not been produced by Ch_b return $\mathrm{Open}(gpk, ok, reg, m, s)$.

ReadGsk: On input i return $gsk[i]$. Add i to the query list Q_{ReadGsk}.

TRACEABILITY. We want to avoid forged group signatures. The issuer can always make a dummy registration and create group signatures, so we cannot rule out the creation of group signatures. What we want to capture here is that if the issuer is honest, then it is infeasible to create a signature that does not belong to some member i. For all non-uniform polynomial time adversaries we have

$$\Pr\Big[(gpk, ik, ok) \leftarrow \mathrm{GKg}(1^k); (m, s) \leftarrow \mathcal{A}^{\mathrm{Join}}(gpk, ok); (i, \psi) \leftarrow \mathrm{Open}(gpk, ok, reg, m, s) : \mathrm{GVf}(gpk, m, s) = 1 \text{ and } \mathrm{Judge}(gpk, i, reg[i], m, s, \psi) = 1 \text{ and } i = \texttt{issuer}\Big] \approx 0,$$

where the oracle is

Join: On input (gpk, i, vk_i) register $reg[i] = vk_i$. Run the issuer's protocol on (i, vk_i) and return $cert_i$.

NON-FRAMEABILITY. We want to a void that an honest member is not falsely attributed a signature that it did not sign, even if both the issuer and opener are controlled by the adversary. We require that for all non-uniform polynomial time adversaries $\mathcal{A}$ we have

$$\Pr\Big[(gpk, ik, ok) \leftarrow \mathrm{GKg}(1^k); (m, s, i, \psi) \leftarrow \mathcal{A}^{\mathrm{JoinHonest,Issue,ReadGsk,GSig}}(gpk, ik, ok) : \mathrm{GVf}(gpk, m, s) = 1 \ \wedge \ i \text{ is an honest member with} reg[i] \neq \varepsilon, \mathrm{Judge}(gpk, i, reg[i], m, s, \psi) = 1 \ \wedge \ i \notin Q_{\mathrm{ReadGsk}} \ \wedge \ (m, s) \notin Q_{\mathrm{GSig}}\Big] \approx 0,$$

where the oracles not described before are as follows

JoinHonest: On input i, where $reg[i]$ is empty first generate (vk_i, sk_i) as specified in the Join-algorithm. Then store $reg[i] \leftarrow vk_i$ and send (i, vk_i) to the issuer. Add i to Q_{Join}.

ReadGsk: On input i return $gsk[i]$. Add i to $\mathrm{Q}_{\mathrm{ReadGsk}}$.

GSig: On input (i, m) check whether $gsk[i]$ is non-empty. In that case return $s \leftarrow \mathrm{GSig}(gpk, gsk[i], m)$. Add (m, s) to the query list Q_{GSig}.

The above definition addresses a partially dynamic setting where members can be enrolled along the way. It also separates the roles of granting membership from opening signatures. In [BMW03] a simpler situation is considered. Only a single group manager that acts both as issuer and opener is considered. All members' keys are set up from the start, there is no enrollment. This relaxation permits the definitions of traceability and non-frameability to be combined into one requirement called full-traceability. In the following we concentrate on the stronger [BSZ05] model as described above and provide a secure implementation based on the DLIN assumption.

7.3 Construction of a Group Signature Scheme

Our construction is related to the constructions in [BMW03, BSZ05]. We use four tools that we have constructed earlier in the paper: CPA-secure encryption, CMA-secure signatures, NIZK proofs with simulation-sound extractability and a strong one-time signature scheme.

The public key will be on the form (vk, pk, Σ), where vk is a verification key for the CMA-secure signature scheme, pk is a public key for the CPA-secure cryptosystem and Σ is a common reference string. The issuer's key ik is the signing key corresponding to the signature scheme, while the opener's key ok is the decryption key for the cryptosystem.

To join the user i creates a signature key pair (vk_i, sk_i). He sends vk_i to the issuer who returns a signature $cert_i$ on vk_i. The user checks that $cert_i$ is a valid signature on vk_i. His group signing key is $(sk_i, vk_i, cert_i)$.

To sign a message the member creates a strong one-time signature key pair $(vk_{\text{sots}}, sk_{\text{sots}})$. Using sk_i he forms a signature s_i on vk_{sots}. He then creates an encryption c of $(vk_i, cert_i, s_i)$ and makes an NIZK proof π that the plaintext is correctly formed. Finally, he makes a strong one-time signature s_{sots} on $m, vk_{\text{sots}}, c, \pi$. The group signature on m is $s = (vk_{\text{sots}}, c, \pi, s_{\text{sots}})$. Verification of a group signature is straightforward, simply check the strong one-time signature and the NIZK proof.

To open a valid group signature we decrypt c. We get some $vk_*, cert_*, s_*$ and look up the member i who registered vk_*. In case no such member exists, we set $i = \texttt{issuer}$. We return an opening (i, ψ), where $\psi = (vk_*, cert_*, s_*)$. Anybody can check whether $cert_*$ is a signature on vk_* under vk, and whether s_* is a signature on vk_{sots} under vk_*. If vk_* has been registered for user i or no vk_* has been registered and $i = \texttt{issuer}$ we accept the opening.

The security intuition behind the group signature scheme is as follows. We get anonymity, because the information $(vk_i, cert_i, s_i)$ that could identify the signer is encrypted. Since the NIZK proof is zero-knowledge it does not reveal anything either. Even seeing openings of other group signatures does not help, because when a CPA-secure cryptosystem is combined with a simulation-sound proof of knowledge of the plaintext, then it becomes CCA2-secure, see also [DP92].

We get traceability because by the soundness of the NIZK proof system we must have a correct $vk_*, cert_*, s_*$ inside the ciphertext. Since only the issuer knows the signing key ik, nobody else can forge a certificate $cert_*$. This means, the group signature must point to some member i, not the issuer.

We have non-frameability because a valid signature and a valid opening pointing to i contains a signature s_* under vk_i on vk_{sots}, so vk_{sots} must have been signed by the member. Furthermore, since it is a strong one-time signature scheme and the public key vk_{sots} is used only once by i, it must also be this member that made the signature s_{sots} on $(m, vk_{\text{sots}}, c, \pi)$.

Let us consider how to make the NIZK proof. Let $S = S_{\text{gs}}(vk, pk, vk_{\text{sots}}, c)$ be a set of pairing equations that are satisfiable only on a witness $w = (vk_*, cert_*, s_*, \vec{a})$ such that $\text{Ver}_{vk}(vk_*, cert_*) = 1, \text{Ver}_{vk_*}(vk_{\text{sots}}, s_*) = 1$ and c has plaintext $(vk_*, cert_*, s_*)$. Correspondingly, let $w = W_{\text{gs}}(vk, pk, vk_{\text{sots}}, c, vk_*, cert_*, s_*, R)$ be such a witness w. Observe, since verification of the signatures consist of verifying a set of pairing equations, and since we know how to prove that something is a plaintext of c it is straightforward to compute the set S and the corresponding satisfiability witness. The witness will be $w = (vk_*, cert_*, s_*, g^{r_1}, g^{r_2}, \ldots)$, where $R = (r_1, s_1, r_2, s_2, \ldots)$. We can now invoke the simulation-sound extractable NIZK proof from Appendix 6.

Key generation: On input 1^k do

1. $(p, \mathbb{G}, \mathbb{G}_1, e, g) \leftarrow \mathcal{G}(1^k)$
2. $(vk, ik) \leftarrow K_{\text{sign}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$
3. $(pk, ok) \leftarrow K_{\text{cpa}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$
4. $\Sigma \leftarrow K_{\text{sse}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$

Return (gpk, ik, ok) where the group public key is $gpk = (p, \mathbb{G}, \mathbb{G}_1, e, g, vk, pk, \Sigma)$.

Join/Issue: On input (gpk, i) to Join generate $(vk_i, sk_i) \leftarrow K_{\text{sign}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$. Store vk_i in $reg[i]$ and send (i, vk_i) to the issuer.

The issuer on input (i, vk_i) checks that vk_i has been stored in $reg[i]$. In that it sends $cert_i \leftarrow \text{Sign}_{ik}(vk_i)$ to i.

User i on input $cert_i$ to Join verifies that $\text{Ver}_{vk}(vk_i, cert_i) = 1$. It accepts if this is the case and stores $gsk[i] = (sk_i, vk_i, cert_i)$. The user has now become a member.

Group signature: On input $(gpk, gsk[i], m)$ with non-empty $gsk[i]$ do

1. $(vk_{\text{sots}}, sk_{\text{sots}}) \leftarrow K_{\text{sots}}(p, \mathbb{G}, \mathbb{G}_1, e, g)$
2. $s_i \leftarrow \text{Sign}_{sk_i}(vk_{\text{sots}})$
3. $c = E_{pk}(vk_i, cert_i, s_i; R)$, with randomness $R = (r_1, s_1, \ldots)$
4. $\pi \leftarrow P_{\text{sse}}(\Sigma, S_{\text{gs}}(vk, pk, vk_{\text{sots}}, c), W_{\text{gs}}(vk, pk, vk_{\text{sots}}, vk_i, cert_i, s_i, R))$
5. $s_{\text{sots}} \leftarrow \text{Sign}_{sk_{\text{sots}}}(m, vk_{\text{sots}}, c, \pi)$

Return the group signature $s = (vk_{\text{sots}}, c, \pi, s_{\text{sots}})$.

Verify signature: On input (gpk, m, s) do

1. Check that $\text{Ver}_{vk_{\text{sots}}}((m, vk_{\text{sots}}, c, \pi), s_{\text{sots}})$
2. Check that $\text{V}_{\text{sse}}(\Sigma, S_{\text{gs}}(vk, pk, vk_{\text{sots}}, c), \pi) = 1$

Return 1 if both checks pass, else return 0.

Opening: On input (gpk, ok, reg, m, s) do

1. Return 0 if $V_{\text{sse}}(\Sigma, S_{\text{gs}}(vk, pk, vk_{\text{sots}}, c), \pi) = 0$
2. $(vk_*, cert_*, s_*) = D_{ok}(c)$
3. Look up vk_* in reg and find the corresponding i. If no such i exists set $i =$ `issuer`
4. Let $\psi = (vk_*, cert_*, s_*)$

Return (i, ψ).

Judge: On input $(gpk, m, s, reg[i], i, \psi)$ do

1. Verify $\text{GVf}(gpk, m, s) = 1$
2. Check that $reg[i] = vk_*$
3. Verify $\text{Ver}_{vk}(vk_*, cert_*) = 1$
4. Verify $\text{Ver}_{vk_*}(vk_{\text{sots}}, s_*) = 1$
5. Verify $\text{Ver}_{vk_{\text{sots}}}(m, vk_{\text{sots}}, c, \pi) = 1$

Return 1 if all these checks pass, else return 0.

Theorem 22 *The group signature scheme described above has perfect correctness, and anonymity, traceability and non-frameability if the DLIN assumption holds for $\mathcal{G}$. All keys contain $\mathcal{O}(1)$ group elements, openings contain $\mathcal{O}(1)$ group elements, and signatures contain $\mathcal{O}(1)$ group elements and elements from $\mathbb{Z}_p$.*

Proof.

Perfect correctness: This follows from the perfect correctness of the signature scheme and the perfect completeness of the NIZK proof.

Anonymity: By a hybrid argument it suffices to prove anonymity in the case where the challenge oracle Ch_b is only queried once. In other words, the adversary queries with i_0, i_1, m and receives a challenge signature $s \leftarrow \text{GSig}(gpk, gsk[i_b], m)$. It has access to arbitrary openings except of (m, s) and knows ik as well as all the members' group signature keys and must now try to guess b.

We start with the Ch_0 oracle and modify it such that we run SE_1 in the key generation algorithm to get Σ. In the challenge, we simulate the NIZK proof π for c encrypting $(vk_{i_0}, cert_{i_0}, s_{i_0})$. By the unbounded zero-knowledge of the NIZK proof the adversary's success probability changes negligibly.

Next, let us observe that the probability of reusing vk_{sots} as the verification key for the one-time signature in any valid query to the opening oracle is negligible since it is a strong one-time signature scheme. We can therefore from now on assume that does not happen.

This implies that the statement $(vk, pk, vk_{\text{sots}}, c)$ that is proved in the group signature in any query to the opening oracle is different from the statement in the challenge signature. Therefore, instead of using the opener's key ok to extract the plaintext $(vk_i, cert_i, s_i)$ we may as well use the knowledge extractor for the NIZK proof. By the simulation-sound extraction property this gives the right opening with overwhelming probability.

Since we are not decrypting any ciphertexts any more, we can use the semantic security to argue that the adversary's success probability is the same when we change the ciphertext in the challenge to encrypt $(1, \ldots, 1)$ instead of $(vk_{i_0}, cert_{i_0}, s_{i_0})$.

By a similar argument, we can argue that the game where the adversary has access to Ch_1 gives him similar success probability when seeing a challenge consisting of $c = E_{pk}(1, \ldots, 1)$ and a simulated proof π.

Traceability: We want a guarantee that if the issuer is honest, then every signature can be traced back to a member. Consider a valid group signature $s = (vk_{\text{sots}}, c, \pi, s_{\text{sots}})$ on m. By the perfect soundness of the proof π we know that c contains a plaintext $(vk_*, cert_*, s_*)$, where $\mathrm{Ver}_{vk}(vk_*, cert_*) = 1$. The opening points to i that has registered vk_*, unless no such registration took place. However, if no such registration took place, then the issuer never signed vk_*. This implies that we have created a forged signature on vk_* and therefore broken the CMA-security of the signature scheme.

Non-frameability: In this definition, both the issuer and the opener are corrupt. We want to guarantee that no uncorrupt member is framed. Consider an adversary that creates a valid signature $(vk_{\text{sots}}, c, \pi, s_{\text{sots}})$ on m and a valid opening $(i, (vk_i, cert_*, s_*)$ pointing to i. In all group signatures s' this member made, it generated a signature s'_{sots} using a random key pair $(vk'_{\text{sots}}, sk'_{\text{sots}})$. By the strong unforgeability the adversary cannot recycle vk'_{sots}. Therefore, the adversary must have chosen a new vk_{sots} that has never been signed by member i. This means s_* is a forged signature on vk_{sots} and we have broken the CMA-security of the signature scheme.

$\square$

KEY GENERATION. The security definitions in [BMW03, BSZ05] rely on a trusted key generation. Their security guarantees guard against key exposures, but not against a malicious key generator. Let us consider which parts need to be trusted.

Security for all parties rely on $(p, \mathbb{G}, \mathbb{G}_1, e, g)$ being a DLIN group. Assuming that it is a DLIN group we get non-frameability even if the issuer and opener cooperate to generate the rest of the group public key gpk.

The opener is in control of the anonymity, if it is corrupt there is no anonymity. We can therefore assume that the opener will try to help members get anonymity. One of the components in giving anonymity is the CPA-security of the cryptosystem. It is therefore reasonable to let the opener generate pk and in the process learn the corresponding decryption key ok. Another part is the zero-knowledge property and the simulation-sound extractability of the NIZK proof. We can let the opener generate the common reference string Σ.

The issuer can create arbitrary group signatures on its own. If it is dishonest we cannot protect ourselves against forgeries. Therefore, it is reasonable to let it aid us in guaranteeing traceability. One part of this is to let it generate (vk, ik) such that certificates cannot be forged. Another part of this is to let it verify that the opener knows the secret key corresponding to pk and has generated Σ so it is indeed perfectly sound. The issuer can for instance request an interactive zero-knowledge proof of knowledge from the opener to get these guarantees.

We conclude that the trust in the key generation algorithm boils down to trust in the DLIN problem being hard in the group $(p, \mathbb{G}, \mathbb{G}_1, e, g)$.

References

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/2005/385`.

[ACJT00] Guiseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure group signature scheme. In *proceedings of CRYPTO '00, LNCS series, volume 1880*, pages 255–270, 2000.

[AdM03] Giuseppe Ateniese and Breno de Medeiros. Efficient group signatures without trapdoors. In *proceedings of ASIACRYPT '03, LNCS series, volume 2894*, pages 246–268, 2003. Revised paper available at `http://eprint.iacr.org/2002/173`.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 56–73, 2004.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid encryption problem. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 171–188, 2004. Full paper available at `http://eprint.iacr.org/2003/077`.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *proceedings of CRYPTO '04, LNCS series, volume 3152*, pages 41–55, 2004.

[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, LNCS series, volume 3378*, pages 325–341, 2005.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 614–629, 2003.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS '93*, pages 62–73, 1993.

[BS99] Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In *proceedings of CRYPTO '99, LNCS series, volume 1666*, pages 519–536, 1999.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *proceedings of CT-RSA '05, LNCS series, volume 3376*, pages 136–153, 2005.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In *proceedings of EUROCRYPT '06, LNCS series, volume 4004*, pages 427–444, 2006.

[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`.

[CG04] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *proceedings of SCN '04, LNCS series, volume 3352*, pages 120–133, 2004. Full paper available at `http://www.brics.dk/∼jg/GroupSignFull.pdf`.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. In *proceedings of STOC '98*, pages 209–218, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes, 2004.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *proceedings of CRYPTO '03, LNCS series, volume 2729*, pages 565–582, 2003. Full paper available at `http://eprint.iacr.org/2003/174`.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *proceedings of CRYPTO '02, LNCS series, volume 2442*, pages 61–76, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *proceedings of CRYPTO '04, LNCS series, volume 3152*, pages 56–72, 2004.

[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`.

[CM98] Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In *proceedings of ASIACRYPT '98, LNCS series, volume 1514*, pages 160–174, 1998.

[CS97] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *proceedings of CRYPTO '97, LNCS series, volume 1294*, pages 410–424, 1997.

[CS98] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. In *proceedings of CRYPTO '98, LNCS series, volume 1462*, pages 13–25, 1998. Full paper available at `http://eprint.iacr.org/2001/108`.

[CvH91] David Chaum and Eugène van Heyst. Group signatures. In *proceedings of EUROCRYPT '91, LNCS series, volume 547*, pages 257–265, 1991.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM Journal of Computing*, 30(2):391–437, 2000.

[DDO$^+$02] 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.

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *proceedings of FOCS '92*, pages 427–436, 1992.

[FI05] Jun Furukawa and Hideki Imai. An efficient group signature scheme from bilinear maps. In *proceedings of ACISP '05, LNCS series, volume 3574*, pages 455–467, 2005.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal of Computing*, 29(1):1–28, 1999.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *proceedings of FOCS '03*, pages 102–, 2003. Full paper available at `http://eprint.iacr.org/2003/034`.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *proceedings of CRYPTO '06, LNCS series, volume 4117*, pages 97–111, 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero-knowledge for np. In *proceedings of EUROCRYPT '06, LNCS series, volume 4004*, pages 339–358, 2006.

[Gro03] Jens Groth. Rerandomizable and replayable chosen ciphertext attack secure public key encryption. In *proceedings of TCC '04, LNCS series, volume 2951*, pages 152–170, 2003.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In *proceedings of TCC '06, LNCS series, volume 3876*, pages 581–600, 2006.

[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.

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In *proceedings of EUROCRYPT '04, LNCS series, volume 3027*, pages 571–589, 2004. Full paper available at `http://eprint.iacr.org/2004/007`.

[KY05] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In *proceedings of EUROCRYPT '05, LNCS series, volume 3494*, pages 198–214, 2005. Full paper available at `http://eprint.iacr.org/345`.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant round secure two-party computation. In *proceedings of CRYPTO '01, LNCS series, volume 2139*, pages 408–432, 2001. Full paper available at `http://eprint.iacr.org/2001/107`.

[Lin03] Yehuda Lindell. A simpler construction of cca2-secure public-key encryption under general assumptions. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 241–254, 2003.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *proceedings of CRYPTO ’02, LNCS series, volume 2442*, pages 111–126, 2002.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *proceedings of STOC ’90*, pages 427–437, 1990.

[Sah01] Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *proceedings of FOCS ’01*, pages 543–553, 2001.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *proceedings of EUROCRYPT ’05, LNCS series, volume 3494*, pages 114–127, 2005.