Short Non-interactive Zero-Knowledge Proofs

Jens Groth[⋆]

University College London
j.groth@ucl.ac.uk

Abstract. We show that probabilistically checkable proofs can be used to shorten non-interactive zero-knowledge proofs. We obtain publicly verifiable non-interactive zero-knowledge proofs for circuit satisfiability with adaptive and unconditional soundness where the size grows quasi-linearly in the number of gates. The zero-knowledge property relies on the existence of trapdoor permutations, or it can be based on a specific number theoretic assumption related to factoring to get better efficiency. As an example of the latter, we suggest a non-interactive zero-knowledge proof for circuit satisfiability based on the Naccache-Stern cryptosystem consisting of a quasi-linear number of bits. This yields the shortest known non-interactive zero-knowledge proof for circuit satisfiability.

Keywords: Non-interactive zero-knowledge proofs, adaptive soundness, probabilistically checkable proofs, Naccache-Stern encryption.

1 Introduction

Zero-knowledge proofs introduced by Goldwasser, Micali and Rackoff [GMR89] are interactive protocols that enable a prover to convince a verifier about the truth of a statement without leaking any information but the fact that the statement is true. Blum, Feldman and Micali [BFM88] followed up by introducing non-interactive zero-knowledge (NIZK) proofs where the prover outputs just one message called a proof, which convinces the verifier of the truth of the statement. The central properties of zero-knowledge proofs and non-interactive zero-knowledge proofs are completeness, soundness and zero-knowledge.

Completeness: If the statement is true, the prover should be able to convince the verifier.

Soundness: A malicious prover should not be able to convince the verifier if the statement is false.

Zero-knowledge: A malicious verifier learns nothing except that the statement is true.

In this paper, we focus on the NP-complete language of circuit satisfiability, which is the most widely studied language in the context of non-interactive zero-knowledge proofs. The statement is a binary circuit C and a claim that there

[⋆] Supported by Engineering and Physical Sciences Research Council grant number EP/G013829/1.

exists an input, a witness w, such that the circuit outputs 1 when evaluated on w. The prover has the witness w as a private input, while the statement C is public and known both to the prover and the verifier.

Only languages in BPP have non-interactive zero-knowledge proofs in the plain model without any setup [Ore87,GO94,GK96]. Blum, Feldman and Micali [BFM88] therefore suggested the common reference string model, where the prover and the verifier have access to a trusted bit-string. The common reference string can for instance be generated by a trusted third party or by a set of parties executing a multi-party computation protocol. Groth and Ostrovsky [GO07] has as an alternative suggested NIZK proofs in the multi-string model, where many parties generate a random string and the security of the NIZK proof relies on a majority of the strings being honestly generated. In this paper, we work in the common random string model, where the common reference string is a trusted uniformly random bit-string.

NIZK proofs have many applications, ranging from early chosen-ciphertext secure public-key cryptosystems [DDN00,Sah01] to recent advanced signature schemes [BW06,CGS07]. They have therefore been studied carefully in the literature. Blum, Feldman and Micali [BFM88] proposed an NIZK proof for all of NP based on a number theoretic assumption related to factoring. Feige, Lapidot and Shamir [FLS99] gave an NIZK proof for all of NP based on the existence of trapdoor permutations. While these results established the existence of NIZK proofs based on general assumptions, other works [Dam92,DDP02,KP98] have aimed at reducing the complexity of NIZK proofs. Gentry's fully homomorphic cryptosystem based on lattices can reduce the complexity of an NIZK to grow linearly in the witness size as opposed to the circuit size [Gen09]. Groth, Ostrovsky and Sahai [GOS06b,GOS06a,Gro06,GS08] have constructed highly efficient NIZK proofs using techniques from pairing-based cryptography.

1.1 Our Contribution

We construct NIZK proofs for circuit satisfiability with a size that grows quasi-linearly in the size of the circuit. Our NIZK proofs have perfect completeness, statistical soundness, and computational zero-knowledge. The zero-knowledge property of the NIZK proofs can be based on trapdoor permutations or for higher efficiency on the semantic security of the Naccache-Stern cryptosystem based on higher residues [NS98].

The Naccache-Stern cryptosystem is based on a decisional assumption in RSA-type groups, which predates but is otherwise incomparable to the decisional assumptions used in pairing-based NIZK proofs. Surprisingly, the construction based on the Naccache-Stern cryptosystem has better asymptotic efficiency than the recent pairing-based NIZK proofs for circuit satisfiability [GOS06b,GOS06a,GS08] (although pairing-based NIZK proofs remain more efficient for practical purposes due to the large constants involved in our construction). With pairing group elements of size k_G and a circuit size that is polynomial in the security parameter k we get an asymptotic improvement over pairing-based NIZK proofs of a multiplicative factor $\frac{k_G}{\text{polylog}(k)}$. This brings the NIZK proof size within a polylog(k) factor of the size of the circuit itself.

In Table 1, we compare our NIZK proofs with the current state of the art NIZK proofs for circuit satisfiability based on respectively trapdoor permutations [KP98] and specific number theoretic assumptions [GOS06b,GOS06a]. All of these NIZK proofs have an efficient (probabilistic polynomial time) prover and they are all publicly verifiable.

	CRS size	Proof size	Assumption
Kilian-Petrank [KP98]	$\omega(\lvert C\rvert k_T k \log k)$	$\omega(\lvert C\rvert k_T k \log k)$	trapdoor perm.
This work	$\lvert C\rvert k_T \ \log^c(k) + \text{poly(k)}$	$\lvert C\rvert k_T \ \log^c(k) + \text{poly}(k)$	trapdoor perm.
Gentry [Gen09]	$\text{poly}(k)$	$\lvert w\rvert\text{poly}(k)$	lattice-based
GOS [GOS06b]	$\Theta(k_G)$	$\Theta(\lvert C\rvert k_G)$	pairing-based
This work	$\lvert C\rvert \log^c(k) + \text{poly}(k)$	$\lvert C\rvert \log^c(k) + \text{poly}(k)$	Naccache-Stern

Table 1. Comparison of NIZK proofs for security parameter k, circuit size $|C| = k^{O(1)}$, witness size $|w|$, trapdoor permutations over $\{0,1\}^{k_T}$, and pairing group size k_G (usually $k_G \approx k^3$ for 2^{-k} security [GPS08].)

Soundness and zero-knowledge can be adaptive or non-adaptive. In non-adaptive soundness and zero-knowledge, the statement to be proven is chosen independently of the common reference string. Usually, NIZK proofs are used in a context where the common reference string is publicly available though, and it is therefore unreasonable to assume the statement is independent of the common reference string.[1] Adaptive soundness and adaptive zero-knowledge refers to the more realistic setting, where NIZK proofs need to be sound and zero-knowledge even when the common reference string is publicly available and the statement may depend on the common reference string. Our NIZK proofs are both adaptively sound and adaptively zero-knowledge, and in Table 1 we have compared the schemes in the efficient prover, adaptive soundness setting.

1.2 New Techniques

PCPs in NIZK. Probabilistically checkable proofs (PCPs) [AS98,ALM$^+$98,Din07] are proofs for a statement that can be verified by reading a few bits in the proof instead of reading the whole proof. A PCP for a circuit being satisfiable will be larger than the circuit itself; however, one only needs to read a few bits of the proof to get more than 50% chance of detecting an attempt to prove a false statement. By reading more bits, we can get exponentially small risk of wrongly being convinced by the PCP.

[1] We have a hard time thinking of any applications where non-adaptive soundness suffices, while non-adaptive zero-knowledge sometimes may be useful.

PCPs have been very useful in the context of zero-knowledge arguments. Kilian [Kil92] for instance suggested a sub-linear size zero-knowledge argument, where the prover commits to the bits of a PCP and the verifier asks the prover to reveal the content of a few of these commitments.

We use PCPs in a different way. In our NIZK proofs the prover will prove that all queries to the PCP have satisfactory answers. At first sight this may seem counterintuitive; the PCP will be larger than the statement itself and it is odd that increasing the statement size would help us in shortening the size of the NIZK proofs. Using a PCP for the statement, however, has the advantage that the verifier can grant the malicious prover a non-trivial chance of falsely answering some of the queries, as long as there are other queries where he will detect the attempt to cheat. This stands in contrast to traditional NIZK proofs, where the verifier needs high certainty for every single part of the statement being correct.

To illustrate our technique, consider an NIZK proof such as Kilian-Petrank [KP98]. They first reduce circuit satisfiability to 3SAT5; where each clause has three variables and each variable appears in at most 5 clauses. By choosing a trapdoor permutation and revealing hard-core bits related to the common reference string, the prover can demonstrate that each clause is satisfied. There is a risk of error though, and the prover therefore needs to repeat the proof many times for each clause to increase the soundness guarantee.

Our idea is to use a PCP in a pre-processing step before applying the techniques of Kilian and Petrank. The effect of the PCP (see Section 3) is to increase the gap between satisfiable and unsatisfiable statements. In a standard 3SAT5 statement there are unsatisfiable statements where all but one clause can be satisfied. With the PCP, however, we can ensure that either all clauses can be satisfied or alternatively a constant fraction of the clauses are unsatisfied. The advantage over Kilian and Petrank's NIZK proof is that now we have resilience towards errors in individual clauses. Even if a malicious prover succeeds in falsely creating NIZK proofs for some of the clauses being satisfied, we still get soundness as long as this only happens for a small constant fraction of clauses. We can therefore avoid the repetition of proofs that Kilian and Petrank needed.

Implementing a Hidden Random String. We construct our NIZK proofs in two steps. We use cryptographic techniques to convert the common reference string into a hidden string of random bits, where the prover may selectively disclose some of the bits and keep other bits secret. We then construct NIZK proofs that assume the existence of a string of hidden bits, where the prover may keep some of them secret and reveal others to the verifier.

Feige, Lapidot and Shamir [FLS99] suggested the following way of implementing the hidden bits model. When working with trapdoor permutations, we can interpret the common reference string as a string of images of the trapdoor permutation. The hidden random bits are hardcore bits of the pre-images. The prover may with the knowledge of the trapdoor learn all the hidden random bits. By revealing a pre-image to the trapdoor permutation, she can disclose the value of a particular hidden random bit. This is a costly approach, however, since we only get one hidden random bit per trapdoor permutation image. In general, the common reference string has to be a factor k_T larger than the hidden random string, where k_T is the size a trapdoor permutation value.

The second contribution of this paper is using the Naccache-Stern cryptosystem [NS98] to reduce the cost of implementing the hidden bits model. We interpret the common reference string as a series of ciphertexts, but with the Naccache-Stern cryptosystem each ciphertext may hold many hardcore bits. The message space is of the form $\mathbb{Z}_P$, where $P = \prod_{i=1}^{n} p_i$ is a product of small primes of size $\log k$. We will show that with the Naccache-Stern cryptosystem, it is possible to disclose the plaintext modulo p_i without revealing the rest of the plaintext. This means that we can have $\Omega(\frac{k_G}{\log k})$ hidden random bits in each ciphertext, giving a common reference string that is only a factor $O(\log k)$ larger than the hidden random string.

Combining PCPs and the Naccache-Stern cryptosystem, we get the asymptotically shortest known NIZK proofs for circuit satisfiability consisting of a quasi-linear number of bits.

1.3 Overview

We construct NIZK proofs for circuit satisfiability in three steps. In Section 3 we describe how a PCP can be used to convert the circuit into a Gap-3SAT5 formula, where either all clauses are satisfiable or alternatively there are at least a constant fraction of unsatisfiable formulae. In Section 4 we construct an NIZK proof in the hidden bits model, where it is assumed that the prover has access to a string of uniformly random bits and may reveal an arbitrary subset of these bits and their positions to the verifier. Finally, in Sections 5 and 6 we show how to implement the hidden bits model under the general assumption of the existence of trapdoor permutations and more efficiently under a concrete number theoretic assumption related to factoring. The two main contributions of the paper are the conceptual idea of using PCPs in a preprocessing step as described in Section 3 and the introduction of a new technique for efficiently implementing the hidden random bits model using the Naccache-Stern cryptosystem described in Section 6.

2 Preliminaries

NOTATION. Given two functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(k) \approx g(k)$ when $|f(k) - g(k)| = O(k^{-c})$ for every constant $c > 0$. We say that f is *negligible* if $f(k) \approx 0$ and that f is *overwhelming* if $f(k) \approx 1$.

We write $y = A(x; r)$ when the algorithm A on input x and randomness r, outputs y. We write $y \leftarrow A(x)$ for the process of picking randomness r at random and setting $y = A(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from the set S. We will for convenience assume uniform random sampling from various types of sets is possible; there are easy ways to amend our protocols to the case where the sets are only sampleable with a distribution that is statistically close to uniform.

NIZK PROOFS. Let R be a polynomial time computable binary relation. For pairs $(C, w) \in R$ we call C the statement and w the witness. Let L be the NP-language consisting of statements with witnesses in R. In this paper, we will focus on the case where the statements are circuits and L is the language of satisfiable circuits, i.e., where there exists an input w so $C(w) = 1$. The size of the NIZK proofs will depend on the size of the statement. We will write L_n for the language of satisfiable circuit consisting of n binary gates and write R_n for the corresponding relation.

An efficient-prover non-interactive proof for the relation R consists of three probabilistic polynomial time algorithms (K, P, V). K is the common reference string generator that takes the security parameter written in unary 1^k and an intended statement size n as input and outputs a common reference string σ of length $\Omega(k)$. P is the prover algorithm that takes as input the common reference string σ, a statement C and a witness w so $(x, w) \in R$ and outputs a proof π. V is the verifier algorithm that on a common reference string σ, a statement C and a proof π outputs 0 or 1. We interpret a verifier output of 0 as a rejection of the proof and a verifier output of 1 as an acceptance of the proof. We call (K, P, V) a non-interactive proof system for R it is complete and sound as described below.

PERFECT COMPLETENESS. Completeness means that a prover with a witness can convince the verifier. For all adversaries $\mathcal{A}$ and $n = k^{O(1)}$ we have

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

STATISTICAL SOUNDNESS. Soundness means that it is impossible to convince the verifier of a false statement. For all non-uniform polynomial time adversaries $\mathcal{A}$ and $n = k^{O(1)}$ we have

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

COMPUTATIONAL ZERO-KNOWLEDGE. A non-interactive argument (K, P, V) is computationally zero-knowledge if it is possible to simulate the proof of a true statement without knowing the witness. Formally, we require the existence of a probabilistic polynomial time simulator $S = (S_1, S_2)$. S_1 outputs a simulated common reference string σ and a simulation trapdoor τ. S_2 takes the simulation trapdoor and a statement as input and produces a simulated proof π. We require for all non-uniform polynomial time stateful interactive adversaries $\mathcal{A}$ and $n = k^{O(1)}$ that

$$\Pr\left[\sigma \leftarrow K(1^k, n); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow P(\sigma, C, w) : (C, w) \in R_n \text{ and } \mathcal{A}(\pi) = 1\right]$$
$$\approx \Pr\left[(\sigma, \tau) \leftarrow S_1(1^k, n); (C, w) \leftarrow \mathcal{A}(\sigma); \pi \leftarrow S_2(\tau, C) : (C, w) \in R_n \text{ and } \mathcal{A}(\pi) = 1\right].$$

3 Preprocessing with Probabilistically Checkable Proofs

We start by giving a polynomial time reduction f from circuit satisfiability to Gap-3SAT5. The reduction f takes as its input a circuit with n binary gates and outputs a 3SAT formula with $N = n$ polylog n variables and $\frac{5}{3}N$ clauses. The 3SAT formula, will be such that each variable appears exactly 3 times as a positive literal and 2 times as a negative literal. If the input of f is a satisfiable circuit C, it will output a satisfiable 3SAT5 formula $\phi = f(C)$. If the circuit C is unsatisfiable, the reduction f will output a formula $\phi = f(C)$ such that all assignments have at least αN unsatisfied clauses for some constant $\alpha > 0$. We also need a polynomial time witness-reduction f_w, which on input C, w such that $C(w) = 1$ outputs a satisfying assignment $f_w(C, w)$ for the 3SAT5 formula $\phi = f(C)$.

The first step in our reduction is to map the circuit C to a constraint graph $G(C)$ with the following properties. The vertices of the constraint graph G may be assigned values from a constant size alphabet Σ, but each edge between two vertices describes a constraint on the values that they may be assigned. When starting with a satisfiable circuit, the output is a satisfiable constraint graph. However, on an unsatisfiable circuit the output is an unsatisfiable constraint graph where any assignment violates at least a α_0-fraction of the constraints for some constant $\alpha_0 > 0$. The polynomial time assignment tester [DR04] given by Dinur [Din07] in her proof of the PCP theorem has the properties described above. Moreover, given a witness for the satisfiability of the circuit C, we may in polynomial time compute a satisfying assignment for the constraint graph $G(C)$. Dinur's most efficient assignment tester building on work by Ben-Sasson and Sudan [BSS08] outputs a constraint graph $G(C)$ with n polylog n vertices and edges.

Given a constraint graph G over a constant size alphabet Σ, we assign a constant number of binary variables to each vertex such that it is possible to represent any element from the alphabet Σ. Each constraint between two vertices is of constant size Σ^2 and we can therefore write out a constant size 3SAT formula describing the constraint. Taking the conjunction of all these 3SAT formulas, we reduce the constraint graph to a 3SAT formula with n polylog n variables and clauses. A satisfying assignment for the constraint graph gives us a satisfying assignment to the 3SAT formula. Since each vertex has a constant number of variables associated with it, and each constraint has a constant number of clauses associated with it, a constraint graph with a constant fraction α_0 of unsatisfiable constraints reduces to a 3SAT formula with a α_1 fraction of unsatisfiable clauses for some constant $\alpha_1 > 0$.

Finally, we reduce the 3SAT formula to a 3SAT5 formula where each variable appears in the clauses as exactly 5 literals and each clause has exactly 3 literals. First we copy clauses and variables so each clause has exactly 3 literals and each variable appears at least 3 times. Then the ℓ appearances of a variable as a positive literal x_i or a negative literal $\neg x_i$ are replaced with copies $x_{i1}, \ldots, x_{i\ell}$. For each copy we add 4 clauses for consistency in the truth value assignment with the predecessor $x_{i,j-1 \bmod \ell}$ and the successor $x_{i,j+1 \bmod \ell}$ according to whether the original variable appeared as a positive or negative literal. In these consistency clauses the copy appears twice as a negative literal and twice as a positive literal, so all copies appear as exactly 3 positive literals and 2 negative literals in the resulting 3SAT5 formula. This is a linear size reduction, so we end up with n polylog n variables and clauses. A satisfying assignment for the 3SAT formula gives us a satisfying assignment for the resulting 3SAT5 formula. A 3SAT formula with a constant fraction α_1 of unsatisfiable clauses, gives a 3SAT5 formula with a α fraction of unsatisfiable clauses for some constant $\alpha > 0$.

In summary, there is a pair of polynomial time algorithms (f, f_w) and a constant $\alpha > 0$ so:

Reduction: f takes input a circuit C with n gates and outputs a 3SAT5 formula $f(C)$ with $N = n\ \text{poly}\log\ n$ variables. Each variable appears as 3 positive literals and 2 negative literals, and each clause has exactly 3 literals. If C is satisfiable, then $f(C)$ is satisfiable. If C is unsatisfiable, then all assignments to the variables of $f(C)$ leave at least αN clauses unsatisfied.

Witness-preservation: f_w takes as input a circuit C with n gates and a witness for C being satisfiable and outputs a truth value assignment satisfying the 3SAT5 formula $f(C)$.

4 NIZK Proofs in the Hidden Bits Model

We will now give an NIZK proof in the hidden-bits model for Gap-3SAT5-satisfiability. The ideas in this section are quite similar to Kilian and Petrank [KP98], although our setting allows us to simplify their scheme.

Let L_N be the language of satisfiable 3SAT5 formulae with N variables and $\frac{5}{3}N$ clauses, where each variable appears as 3 positive literals and 2 negative literals. Let R_N be the corresponding relation of formulae and satisfying assignments. Further, define L_N^α as the language of formulae in L_N that have a truth-value assignment to the variables that leaves at most αN clauses unsatisfied. We will be interested in a hidden-bits NIZK proof $(\ell_H(N), P_H, V_H)$ for R with perfect completeness, $(\alpha, \epsilon_H(N))$-soundness, and perfect zero-knowledge as described below.

Perfect completeness. For all $N \in 3\mathbb{N}$ and all $(\phi, w) \in R_N$ we have

$$\Pr\Big[\rho \leftarrow \{0,1\}^{\ell_H(N)}; (i_1, \ldots, i_t) \leftarrow P_H(\rho, \phi, w) : V_H(\phi, i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) = 1\Big] = 1.$$

Statistical soundness. For all $N \in 3\mathbb{N}$ and all adversaries $\mathcal{A}$

$$\Pr\Big[\rho \leftarrow \{0,1\}^{\ell_H(N)}; (\phi, i_1, \ldots, i_t) \leftarrow \mathcal{A}(\rho) : \\ \phi \notin L_N^\alpha \text{ and } V_H(\phi, i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) = 1\Big] \leq \epsilon_H(N).$$

Perfect zero-knowledge. There exists a probabilistic polynomial time simulator S_H such that for all $N \in 3\mathbb{N}$ and all $(\phi, w) \in R_N$ the distribution

$$\{\rho \leftarrow \{0,1\}^{\ell_H(N)}; (i_1, \ldots, i_t) \leftarrow P_H(\rho, \phi, w) : (i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t})\}$$

is identical to the distribution

$$\{(i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) \leftarrow S_H(\phi) : (i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t})\}.$$

4.1 Hidden-bits NIZK proof for Gap-3SAT5

Let ϕ be a 3SAT5 formula with N variables and $\frac{5}{3}N$ clauses, where each variable appears exactly thrice as a positive literal x_i and twice as a negative literal $\neg x_i$ in the clauses. The verifier has the promise that either there is an assignment to the variables so all clauses are satisfied, or all assignments of truth values to the variables lead to more than αN unsatisfied clauses. The prover has a satisfying assignment and wants to give an NIZK proof in the hidden bits model for ϕ being satisfiable.

We first sketch the NIZK proof and then afterwards explain the main ideas in the construction. There is some freedom in the choice of parameters; for concreteness we suggest $a = \lceil \frac{8}{\alpha} \rceil, b = \lceil \log N \rceil, \Delta = \lceil \frac{N}{\log N} \rceil$.

Statement: A 3SAT5 formula $\phi \in L_N$.

Prover's input: A string ρ of $6a2^{6a}(bN + \Delta)$ hidden bits. A truth-value assignment to the variables $x_1, \ldots, x_N$ so $\phi(x_1, \ldots, x_N) = 1$.

Proof:

1. Interpret the hidden bits as $6a2^{6a-1}(bN + \Delta)$ consecutive pairs of bits. Each pair of bits is interpreted as one of three possible characters according to the following scheme

   $$00 = 0 \qquad 01 = W \qquad 10 = W \qquad 11 = 1.$$

   Later the prover may reveal one of the bits in a character. In a wildcard character W the prover can reveal either 0 or 1, whereas 0 can only be revealed as 0 and 1 can only be revealed as 1.
2. Interpret the characters as $2^{6a-1}(bN+\Delta)$ consecutive $6a$-character blocks. Call a block good if it has exactly $3a$ W-characters and they are either all in the first half of the block or all in the second half of the block. Otherwise, call the block bad.
3. A block has 2^{1-6a} chance of being good, so we expect on average $(bN + \Delta)$ good blocks. If the number of good blocks is outside the interval $[bN; bN + 2\Delta]$ reveal all hidden bits and halt.
4. Reveal to the verifier all the hidden bits associated with bad blocks.
5. Assign the first b good blocks to the first variable, etc., so each variable has b blocks assigned to it. The remaining good blocks will not be used.
6. Interpret each good block as a set of 6 consecutive a-character strings (see examples in Figure 1). Assign in the order of appearance, 5 of these a-character strings to the 5 appearances of their variable x_i in the clauses as follows. If the witness has $x_i = 1$, assign the 3 wildcard strings to the 3 appearances of x_i, and the first 2 0/1-strings to the 2 appearances of $\neg x_i$. If the witness has $x_i = 0$, assign the first 2 wildcard strings to the 2 appearances of $\neg x_i$ and the 3 0/1-strings to the 3 appearances of x_i. Taking all good blocks into account, each appearance of x_i or $\neg x_i$ has b a-character strings assigned to it.

7. Each clause has 3 literals, and each literal has a corresponding tuple of b a-character strings. Pick at random a literal for which the b a-character strings only contain wildcard characters. Such a literal must exist since the clause is satisfied by the truth-value assignment. For the other two literals reveal a random bit in each character's bit pair, thereby revealing two ab-bit strings. In the remaining wildcard literal, reveal in each wildcard character one of the bits, such that the revealed ab-bit string is the exclusive-or of the two other ab-bit strings.
8. The proof consists of the revealed bits. If the number of good blocks is outside the interval $[bN; bN + 2\Delta]$ the proof reveals all hidden bits. Else, the proof reveals the hidden bits of the bad blocks and a $\frac{5}{12}$-fraction of the hidden bits in the first bN good blocks.

Verification:

1. If the proof reveals all hidden bits, return 1 if the number of good blocks is outside the range $[bN; bN + 2\Delta]$ and else return 0.
2. Verify that there are no good blocks among the blocks where all bits have been revealed.
3. Verify that there are at most $bN + 2\Delta$ blocks where some of the bits remain hidden. Associate the first bN blocks with the variables in the order of appearance.
4. Verify that in each of the bN blocks corresponding to variables, exactly 5 of the 6 a-character strings have one revealed bit in each character. Verify also that in each block either the last a-character string in the first half of the block, or the last a-character string in the second half of the block has no revealed bits. Based on this, each revealed a-bit string can be uniquely associated with a corresponding literal in a clause.
5. For each clause, verify that the exclusive-or of the two first ab-bit strings corresponding to the first 2 literals equals the ab-bit string corresponding to the third literal.
6. Return 1 if all verifications passed, else return 0.

In the first step, note that there is 50% chance that a character is a wildcard and 50% chance that it is a 0 or a 1. Later, the prover will open some of the characters by revealing one of the bits. Wildcards can be opened as 0 or 1, whereas 0 can only be opened as 0 and 1 can only be opened as 1. The prover sets up the strings so wildcards correspond to true literals and non-wildcards correspond to false literals. In satisfied clauses there is a true literal, which can be opened at will. This is what gives the prover with a satisfying assignment the power to convince the verifier. On the other hand, in an unsatisfied clause there will only be non-wildcard characters associated with the false literals, which will reduce the power of the prover and make it hard to convince the verifier of a false statement. Finally, for zero-knowledge we can set more of the characters to be wildcard characters. This will make it possible to simulate a proof without knowing a satisfying truth assignment for the statement.

We interpret the string of characters as blocks of $6a$ characters. There will be an expected number of $bN + \Delta$ good blocks. We can use Chernoff-bounds to see that there is high probability that most of the hidden blocks that have not been revealed are indeed good blocks. The point of sampling good blocks is that they represent a consistent view of a variable. All true literals are assigned wildcard strings, all false literals are assigned non-wildcard string.

010 . . . 101	100 . . . 011	010 . . . 110	$WW\ldots W$	$WW\ldots W$	$WW\ldots W$

$WW\ldots W$	$WW\ldots W$	$WW\ldots W$	010 . . . 110	000 . . . 111	100 . . . 101

Fig. 1. Two examples of good blocks.

The important thing to note is that with wildcard string, the prover may open the true literals to any a-bit string. For the false literals, however, the prover is bound to a particular a-bit string. We require that in each clause, the prover should open 3 a-bit strings, such that they exclusive-or to 0. In clauses with a true literal this is easy to accomplish, since the prover may open the corresponding wildcard string to any a-bit string. This gives us completeness. In unsatisfied clauses, however, the prover has 3 fixed a-bit strings and the probability of their exclusive-or being 0 is 2^{-a}. For each unsatisfied clause, we therefore get a good chance of catching a cheating prover.

We have now described the main idea in the construction. The prover has some degrees of freedom in choosing the statement, taking advantage of a few bad blocks that may be camouflaged as good blocks, etc. However, by repeating the proof b times in parallel and using the fact that for unsatisfiable statement there is actually a constant fraction of unsatisfied clauses no matter what the truth assignment is, we can ensure that a cheating prover still has very small chance of convincing the verifier on a false statement.

Theorem 1. *For sufficiently large N the protocol given above is a hidden-bits NIZK proof for 3SAT5 with perfect completeness, $(\alpha, 2^{-\frac{N}{6\log^3 N}})$-soundness and perfect zero-knowledge with a hidden string of size $\ell_H(N) = O(N \log N)$.*

We refer to the full paper for a proof.

5 Implementing the Hidden Bits Proof with Trapdoor Permutations

Trapdoor permutations. We will now implement the hidden bits NIZK proof using trapdoor permutations. A trapdoor permutation is a triple of algorithms $(K_{\mathrm{T}}, F, F^{-1})$. K_{T} generates a public key pk, which we for convenience will assume has k bits, and a secret key sk for the trapdoor permutation. F_{pk} and F_{sk}^{-1} are efficiently computable permutations of k-bit strings, such that

$F_{pk}(F_{sk}^{-1}(y)) = y$. We will assume it is hard to compute F_{sk}^{-1} without knowledge of sk. All trapdoor permutations can easily be converted into trapdoor permutations with a hardcore predicate [GL89] so we will assume the existence of a hardcore predicate B for the trapdoor permutation. If $y \leftarrow \{0,1\}^k$ is chosen uniformly at random then $B(F_{sk}^{-1}(y))$ is uniformly random in $\{0,1\}$ and given only (pk, y) it is computationally hard to decide $B(F_{sk}^{-1}(y))$.

Implementing the hidden bit string. To implement a hidden bit string with N' random bits, we generate a common reference string σ consisting of $k(4N' + 4\Delta')$ uniformly random bits. There is a range of choices of Δ', for the sake of concreteness let us say $\Delta' = \lceil (N')^{\frac{3}{4}} \rceil$. The prover picks a trapdoor permutation and interprets σ as $4N' + 4\Delta'$ images of the trapdoor permutation. This gives the prover $4N' + 4\Delta'$ secret hardcore bits. The prover can selectively open some of the hardcore bits by computing the corresponding preimages and giving them to the verifier. This idea first described in [FLS99] indicates how we can generate hidden random bits that the prover can see and selectively disclose to the verifier.

Our hidden bits proof has perfect zero-knowledge if the simulator can choose the hidden bits itself. Once a trapdoor permutation has been chosen we cannot alter the preimages though so we have not yet implemented the hidden bits model in the *adaptive* zero-knowledge sense. The problem is that the common reference string is chosen before the adversary picks the statement and therefore the simulator needs to get hidden bits out of the simulated common reference string that can be revealed as both 0 and 1 depending on what is needed in the simulation. Our solution is to interpret pairs of hardcore bits as hidden bits as follows:

$$00 = \mathbf{0} \qquad 01 = S \qquad 10 = S \qquad 11 = \mathbf{1}.$$

The prover reveals a hidden bit by revealing one of the two preimages associated with it. This means it is bound to open $\mathbf{0}$ as 0 and open $\mathbf{1}$ as 1, but it can open a soft bit S as either 0 or 1. In the zero-knowledge simulation, we will set up the common reference string such that all hidden bits are soft. When all hidden bits are soft, the zero-knowledge simulator can open them as it likes and simulate the hidden bits proof.

When half the hidden bits are soft we have to be careful to preserve soundness though. We therefore require that the prover reveals the preimages corresponding to soft hidden bits. On average the prover should reveal $N' + \Delta'$ soft hidden bits; and the verifier checks that at last N' soft hidden bits are revealed. This leaves the prover with approximately N' hidden bits, which mostly will be hard hidden bits which can only be opened as 0 or only be opened as 1. Soundness will now follow from the fact that most of the remaining hidden bits are uniformly random hard bits.

NIZK proof. We will now give the full NIZK proof for circuit satisfiability. The statement will be a circuit C and the prover will have a satisfying witness w so $C(w) = 1$. We have to be careful that the prover chooses a well-formed public key for the trapdoor permutation and will therefore use an NIZK proof $(\ell_{\text{well}}, P_{\text{well}}, V_{\text{well}})$ for well-formedness. This NIZK proof could for instance be

Kilian and Petrank's original NIZK proof [KP98], which would have a cost of $\mathrm{poly}(k)$ bits. Alternatively, we could assume the existence of certifiable trapdoor permutations where the well-formedness of the public key is directly verifiable. Or we could use Bellare and Yung's [BY92] method of sampling preimages to show that the public key describes a function close to a trapdoor permutation and then give a more careful probability analysis that deals with the small statistical bias this might introduce in the hidden bits. We will in the following let $N' = O(N \log N) = n$ polylog(n) be the number of bits needed in the hidden bits model for circuits with n gates.

CRS: $\sigma = (\sigma_{1,1}, \ldots, \sigma_{2N'+2\Delta',2}, \sigma_{\mathrm{well}}) \leftarrow \{0,1\}^{k(4N'+4\Delta')+\ell_{\mathrm{well}}(k)}$.
Proof:
1. Generate keys for the trapdoor permutation $(pk, sk) \leftarrow K_{\mathrm{T}}(1^k)$.
2. Compute an NIZK proof π_{well} for pk being a valid public trapdoor permutation key.
3. Compute the hardcore bits $h_{1,1}, h_{1,2}, \ldots, h_{2N'+2\Delta',1}, h_{2N'+2\Delta',2}$ as $h_{i,j} = B(F_{sk}^{-1}(\sigma_{i,j}))$.
4. If there are less than N' pairs or more than $N' + 2\Delta'$ pairs where $h_{i,1} = h_{i,2}$ return the proof $(pk, \pi_{\mathrm{well}}, F_{sk}^{-1}(\sigma_{1,1}), \ldots, F_{sk}^{-1}(\sigma_{2N'+2\Delta',2})$ and halt.
5. For each pair $h_{i,1} \neq h_{i,2}$ include preimages $\pi_{i,1} = F_{sk}^{-1}(\sigma_{i,1})$ and $\pi_{i,2} = F_{sk}^{-1}(\sigma_{i,2})$ in the proof.
6. Let $\rho = (\rho_1, \ldots, \rho_{N'})$ be the values of the first N' remaining pairs of hardcore bits.
7. Run the hidden bit string proof on ρ to get $\pi_H \leftarrow P_H(\rho, f(C), f_w(w))$.
8. For all revealed bits ρ_i in the hidden bits proof π_H corresponding to hardcore bits $h_{j,1} = h_{j,2}$ choose at random to include either $\pi_{j,1} = F_{sk}^{-1}(\sigma_{j,1})$ or $\pi_{j,2} = F_{sk}^{-1}(\sigma_{j,2})$ in the proof.

The proof is of the form $(pk, \pi_{\mathrm{well}}, \pi_{i_1,j_1}, \ldots, \pi_{i_t,j_t})$.

Verification:
1. Verify the NIZK proof π_{well} for pk being a correctly generated public trapdoor permutation key.
2. Verify the correctness of all the preimages $\sigma_{i,j} = F_{pk}(\pi_{i,j})$.
3. Compute the corresponding hardcore bits $h_{i,j} = B(\pi_{i,j})$.
4. If all hardcore bits have been revealed, verify that there are less than N' or more than $N' + 2\Delta'$ pairs $h_{i,1} = h_{i,2}$ and accept if all verifications have succeeded.
5. Verify that all revealed pairs of hardcore bits have $h_{i,1} \neq h_{i,2}$ and that there are between N' and $N' + 2\Delta'$ pairs left in which at most one hardcore bit has been revealed.
6. Interpret the remaining hardcore bits as indices and revealed bits as a hidden bits proof $(i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t})$ and accept if all verifications have succeeded and $V_H(f(C), i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) = 1$.

The construction leads to the following theorem that we prove in the full paper.

Theorem 2. *Assuming the existence of trapdoor permutations on $\{0,1\}^k$ with k-bit keys there is an NIZK proof for circuit satisfiability with perfect completeness, statistical soundness and computational zero-knowledge. The size of the common reference string and the NIZK proof is $|C|$ polylog $|C| \cdot k + \mathrm{poly}(k)$ bits.*

6 Implementing the Hidden Bits Proof with Naccache-Stern Encryption

Naccache-Stern encryption. The Naccache-Stern cryptosystem based on higher residues [NS98] has message space $\mathbb{Z}_P$ where P is a product of small primes. We will show how to reveal the plaintext modulo a small prime factor p_i without revealing the rest of the plaintext. Interpreting even numbers as 0, odd numbers as 1, and $p_i - 1$ as "ignore" we get a uniform distribution of hardcore bits modulo p_i assuming the Naccache-Stern encryption is semantically secure. With Naccache-Stern's cryptosystem having constant expansion rate and each prime factor of the message space being of logarithmic size in the security parameter we can construct a hidden random bits implementation that is quasi-linear in the number of hidden bits.

In the Naccache-Stern cryptosystem the public key is of the form $pk = (M, P, g)$, where M is a k-bit RSA modulus, P is a product of small odd primes $p_1, \ldots, p_d$ so $\gcd(8P^2, \varphi(M)) = 4P$, and $g \in \mathbb{Z}_M^*$ is a group element with $\mathrm{ord}(g) = \frac{\varphi(M)}{4}$. The secret key is $sk = \varphi(M)$. Encrypting a message $m \in \mathbb{Z}_P$ with randomness $r \leftarrow \mathbb{Z}_M^*$ yields the ciphertext

$$c = g^m r^P \bmod M.$$

To decrypt a ciphertext c, compute $c^{\frac{\varphi(M)}{P}} = (g^{\frac{\varphi(M)}{P}})^m$ and use the Pohlig-Hellman algorithm for finding discrete logarithms in groups with a smooth order to compute $m \bmod P$.

The cryptographic assumption underlying our NIZK proof is that there is a probabilistic polynomial time key generator K_{NS} for generating Naccache-Stern keys (pk, sk) such that the cryptosystem is IND-CPA secure and the number of small prime factors in P is larger than $\beta\frac{k}{\log k}$ for some constant $\beta > 0$. We refer to Naccache and Stern [NS98] for concrete key generator suggestions and a proof that the resulting cryptosystem is IND-CPA secure under a computational intractability assumption related to higher residues.

Opening and simulating openings of hardcore bits. In the implementation of the Naccache-Stern cryptosystem, the prover will generate Naccache-Stern keys $pk = (M, P, g)$ and $sk = \varphi(M)$. The random string is interpreted as a series of k-bit integers where those outside $\mathbb{Z}_M^*$ are ignored. An integer in $\mathbb{Z}_M^*$ can be interpreted as a ciphertext encrypting some message $m \bmod P$ where $P = \prod_{i=1}^{t} p_i$. Since there are $d = \lceil \frac{\beta k}{\log k} \rceil$ prime factors in P, this gives the prover d residues $\{m \bmod p_i\}_{i=1}^{d}$, each of which is translated into a hardcore bit. The prover will use the first N' hardcore bits as the hidden bit string and since she gets $\Theta(\frac{k}{\log k})$ bits per element in $\mathbb{Z}_M^*$ she only looses a logarithmic factor in implementing the hidden bit string.

The key observation needed for using Naccache-Stern encryption in this way is that the prover may verifiably disclose $m_i = m \bmod p_i$ without revealing the other parts of the message. Consider a particular k-bit block $c \in \mathbb{Z}_M^*$, which the prover can decrypt to get the plaintext $m \in \mathbb{Z}_P$. All $c \in \mathbb{Z}_M^*$ are valid ciphertexts but there are P different $r \in \mathbb{Z}_M^*$ so $c = r^P g^m$ so we will for notational convenience fix an arbitrary such r in the following. To prove $m_i = m \bmod p_i$ is indeed part of the plaintext the prover gives a proof π satisfying

$$\pi^P = (cg^{-m_i})^{\frac{P}{p_i}}.$$

Raising both sides to the power $\frac{\phi(M)}{P}$ shows

$$1 = (\pi^P)^{\frac{\phi(M)}{P}} = (r^P g^{m-m_i})^{\frac{P}{p_i}\frac{\phi(M)}{P}} = (g^{\frac{\phi(M)}{p_i}})^{m-m_i}$$

telling the verifier that $m_i = m \bmod p_i$ since $P|\mathrm{ord}(g)$. The prover with the secret key $\phi(M)$ can compute a random π satisfying the equation by choosing $s \in \mathbb{Z}_M^*$ at random and setting

$$\pi = (cg^{-m_i})^{(P^{-1} \bmod \frac{\varphi(M)}{P})\frac{P}{p_i}} s^{\frac{\phi(M)}{P}}.$$

In the NIZK proof, we will generalize this idea to verifiably disclose $m \bmod P_I$ for arbitrary $P_I = \prod_{i \in I} p_i$. This makes it possible for the prover to reveal many values $\{m \bmod p_i\}_{i \in I}$ simultaneously.

There is a little variation in how many hardcore bits the prover gets out of a common reference string since not all k-bit integers will belong to $\mathbb{Z}_M^*$ and some hardcore bits are ignored but we can use Chernoff bounds to get a good estimate of how many hardcore bits the prover can extract and tune the proof accordingly. Since the verifier obtains proofs π for the correctness of the opened hardcore bits the soundness of the hidden bit proof system implies soundness of the full NIZK proof for circuit satisfiability.

The zero-knowledge property will come from using a different type of public key. Instead of using g that has order $\frac{\phi(M)}{4}$ the simulator will pick g with order $\frac{\phi(M)}{4P}$. As we shall see in the security proof, the semantic security of the Naccache-Stern cryptosystem implies that the two types of public keys are computationally indistinguishable. With the latter choice of public key $\mathrm{ord}(g) = \frac{\phi(M)}{4P}$ we can write $g = (g')^P$ and now a ciphertext is no longer binding since $c = r^P g^m = r^P (g')^{mP} = (r(g')^{m-m'})^P g^{m'}$ is at the same time an "encryption" of m and m'. The simulator sets up the common reference string so it contains ciphertexts that can be opened to any hardcore bits it chooses thereby allowing it to simulate the hidden bits proof.

NIZK PROOF BASED ON NACCACHE-STERN ENCRYPTION. We will now give the full NIZK proof for circuit satisfiability. The statement is a circuit C and the prover will have a satisfying witness w so $C(w) = 1$. Naccache-Stern keys are not directly verifiable, so we let $(\ell_{\mathrm{well}}, P_{\mathrm{well}}, V_{\mathrm{well}})$ be an NIZK proof system for well-formedness of a Naccache-Stern public key. This NIZK proof could for instance be Kilian and Petrank's original NIZK proof [KP98], which would have a cost of poly(k) bits. We will in the following let $N' = O(N \log N) = n$ polylog(n) be the number of bits needed in the hidden bits model for circuits with n gates and let $\Delta' = \Theta((N')^{\frac{3}{4}})$. For notational simplicity, we will assume $d|N'$ and $\frac{N'+\Delta'}{\delta} \in \mathbb{Z}$, where $d = \lceil \frac{\beta k}{\log k} \rceil$ for a constant $\beta > 0$ and δ is defined in the protocol.

Common reference string: $\sigma = (\sigma_1, \ldots, \sigma_{\frac{3N'}{d}}, \sigma_{\text{well}}) \leftarrow \{0,1\}^{k\frac{3N'}{d} + \ell_{\text{well}}(k)}$.

Proof:

1. Generate Naccache-Stern keys $(pk, sk) = ((M, P, g), \phi(M)) \leftarrow K_{\text{NS}}(1^k)$ with $P = \prod_{i=1}^{d} p_i$.
2. Compute an NIZK proof π_{well} for the well-formedness of $pk = (M, P, g)$.
3. Define $\delta = d - \sum_{i=1}^{d} \frac{1}{p_i}$ and let $c_1, \ldots, c_{\frac{N'+\Delta'}{\delta}}$ be the first $\frac{N'+\Delta'}{\delta}$ of $\sigma_1, \ldots, \sigma_{\frac{3N'}{d}} \in \mathbb{Z}_M^*$.[2] If there are less than $\frac{N'+\Delta'}{\delta}$ of them return the proof $\pi = (pk, sk)$.
4. Decrypt $c_1, \ldots, c_{\frac{N'+\Delta'}{\delta}}$ to get plaintexts $m_1, \ldots, m_{\frac{N'+\Delta'}{\delta}}$. Define $m_{i,j} = m_j \bmod p_i$.
5. Define $h_{1,1}, \ldots, h_{d,\frac{N'+\Delta'}{\delta}}$ as $h_{i,j} = \bot$ if $m_{i,j} = -1$ and otherwise $h_{i,j} = 0$ if $m_{i,j}$ is even and $h_{i,j} = 1$ if $m_{i,j}$ is odd. If there are less than N' or more than $N' + 2\Delta'$ hardcore bits $h_{i,j} \in \{0,1\}$ return the proof $\pi = (pk, sk)$.
6. Define $\rho = (\rho_1, \ldots, \rho_{N'})$ as the first N' hardcore bits $h_{i,j}$.
7. Run the hidden bit string proof on ρ to get $\pi_H \leftarrow P_H(\rho, f(C), f_w(w))$.
8. Define $m_{i,j}$ as revealed if the hardcore bit $h_{i,j}$ is revealed in π_H or $h_{i,j} = \bot$.
9. Let for all j the set $I_j \subset \{1, \ldots, d\}$ be the indices i for which $m_{i,j}$ is revealed. Define $m_{I_j} = m_j \bmod P_{I_j}$ where $P_{I_j} = \prod_{i \in I_j} p_i$. Compute $\pi_j = (cg^{-m_{I_j}})^{(P^{-1} \bmod \frac{\phi(M)}{P})\frac{P}{P_{I_j}}} s_j^{\frac{\phi(M)}{P}}$ for a randomly chosen $s_j \leftarrow \mathbb{Z}_M^*$.

The proof is either $\pi = (pk, sk)$ or
$\pi = (pk, \pi_{\text{well}}, I_1, m_{I_1}, \pi_1, \ldots, I_{\frac{N'+\Delta'}{\delta}}, m_{I_{\frac{N'+\Delta'}{\delta}}}, \pi_{\frac{N'+\Delta'}{\delta}})$.

Verification:

1. If the proof is of the form $\pi = (pk, sk)$ accept it if and only if the key is well-formed (the secret key can be of a form so this can be verified) and there are less than $\frac{N'+\Delta'}{\delta}$ values in $\mathbb{Z}_M^*$ or the number of valid hardcore bits $h_{i,j} \in \{0,1\}$ is less than N' or higher than $N' + 2\Delta'$.
2. Verify the NIZK proof π_{well} for $pk = (M, P, g)$ being a correctly generated public Naccache-Stern key with d small odd primes $p_1, \ldots, p_d$.
3. Identify the first $\frac{N'+\Delta'}{\delta}$ values $c_1, \ldots, c_{\frac{N'+\Delta'}{\delta}} \in \mathbb{Z}_M^*$. Reject if there are less than $\frac{N'+\Delta'}{\delta}$ of them.
4. Verify the proofs $\pi_j^P = (cg^{-m_{I_j}})^{\frac{P}{P_{I_j}}} \bmod M$ and compute the hardcore bits $h_{i,j} \in \{0,1\}$ corresponding to $m_{I_1}, \ldots, m_{I_{\frac{N'+\Delta'}{\delta}}}$. Reject if the number of unopened hardcore bits plus opened valid hardcore bits $h_{i,j}$ is less than N' or more than $N' + 2\Delta'$.
5. Interpret the $h_{i,j} \in \{0,1\}$ as a hidden bits proof $(i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t})$. Accept if the verifications succeed and $V_H(f(C), i_1, \rho_{i_1}, \ldots, i_t, \rho_{i_t}) = 1$.

The construction gives us the following theorem that we prove in the full paper.

[2] We represent elements of $\mathbb{Z}_M^*$ as integers in the range $\{1, \ldots, M-1\}$.

Theorem 3. *Assuming the Naccache-Stern cryptosystem is IND-CPA secure, there is an NIZK proof for circuit satisfiability with perfect completeness, statistical soundness and computational zero-knowledge. The size of the common random string and the proof is* $|C|\ \mathrm{polylog}\ |C| + \mathrm{poly}(k)$ *bits.*

7 Conclusion

We have suggested the shortest known NIZK proofs based on standard intractability assumptions. Based on trapdoor permutations we get an NIZK proof and common reference string of size $|C|k\ \mathrm{polylog} k$ bits (where we use that $\mathrm{polylog}|C| = \mathrm{polylog} k$). This is a factor $\frac{k}{\mathrm{polylog} k}$ improvement over Kilian and Petrank's construction [KP98].

Based on a specific number-theoretic assumption related to factoring, we get a very efficient implementation of a hidden bit string and an even shorter NIZK proof with a complexity of $|C|\ \mathrm{polylog} k$ bits. This is asymptotically a factor $\frac{k^3}{\mathrm{polylog} k}$ more efficient than the pairing-based constructions by Groth, Ostrovsky and Sahai [GOS06b,GOS06a] (assuming the group elements have size $\frac{k^3}{\mathrm{polylog} k}$) although it remains an open problem to reduce the polylogarithmic factor to make our construction practical.

References

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

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

[BSS08] Eli Ben-Sasson and Madhu Sudan. Short pcps with polylog query complexity. *SIAM Journal of Computing*, 38(2):551–607, 2008.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444, 2006.

[BY92] Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 442–460, 1992.

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sublinear size without random oracles. In *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 423–434, 2007.

[Dam92] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with preprocessing. In *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 341–355, 1992.

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

[DDP02] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-optimal characterization of two NP proof systems. In *RANDOM*, volume 2483 of *Lecture Notes in Computer Science*, pages 179–193, 2002.

[Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3), 2007.

[DR04] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the pcp-theorem. In *FOCS*, pages 155–164, 2004.

[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.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal of Computing*, 25(1):169–192, 1996.

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

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proofs. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 323–341, 2007.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero-knowledge for NP. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, 2006.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT*, volume 4248 of *Lecture Notes in Computer Science*, pages 444–459, 2006.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, 2008.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, pages 723–732, 1992.

[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.

[NS98] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *ACM CCS*, pages 59–66, 1998.

[Ore87] Yair Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs. In *FOCS*, pages 462–471, 1987.

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