"METAPROOFS" (and their Cryptographic Applications)

Alfredo De Santis and Moti Yung
IBM Research, T. J. Watson Research Center
Yorktown Heights, NY 10598

March 20, 1990

Abstract

We develop a non-interactive proof-system which we call "Metaproof" (μ-*NIZK proof system*); it provides a proof of "the existence of a proof to a statement". This metamathematical notion indeed seems redundant when we deal with proving $\mathcal{NP}$ statements, but in the context of zero-knowledge theory and cryptography it has a large variety of applications.

Combined with another tool we develop which we call "*on-line simulatable NIZK proof-system*", it is the key tool used to solve the open problem of the existence of a *many prover non-interactive zero-knowledge system* (MP-NIZK proof system). This problem was presented by Micali when the important notion of non-interactive zero-knowledge proofs (NIZK) was first suggested and implemented for a sole prover. The solution immensely enlarges the domain of applications of the NIZK model.

The work also provides a new connection between bounded (single-theorem) non-interactive zero-knowledge proofs and the unbounded (multi-theorem) one. This may help in reducing the complexity assumption upon which to base NIZK systems.

Remark: This is a full version (with more details, more material, and with proofs) of the Crypto 1990 paper on Metaproof. Over the years, the concept has been used and reinvented for specific settings beyond the original ones, by others; (which has made it more useful). Recently, we were asked about this paper and about details, so here they are! For historical reasons, except for this remark, this version is presented as it was in the above mentioned date under the above affiliations, though we did not pursue publication before!

1 Introduction

The development of zero-knowledge proof-systems introduced by Goldwasser, Micali, and Rackoff [GoMiRa] has revolutionized the field of cryptographic primitives and protocols design. Designing of basic primitives was made possible and proving security was made easy.

Two very useful and applicable notions of zero-knowledge proofs were given: interactive proofs (ZKIP) [GoMiRa, GoMiWi1, ImYu] and non-interactive proofs (NIZK) (introduced by Blum, Feldman, and Micali) [BlFeMi, DeMiPe1, BlDeMiPe]. Next we evaluate the current relative advantages and shortcomings of the two notions. Our work is motivated by the differences between the two models.

Comparison between the two models:

One of the strengths of the interactive scheme is the fact that it does not rely on public information, and thus every pair of users can apply it independently in a course of a multi-party computation. The major example is the protocol of Goldreich, Micali, and Wigderson for multi-party computation which can capture computation of any partial-information game [GoMiWi2]. In this procedure various pairs of users (that is, polynomial-time users assumed in cryptographic applications) are performing interactive zero-knowledge proofs. The disadvantage of this model is (naturally) its reliance on interaction of both parties which is an expensive resource and is lacking in certain settings (e.g. mail).

In the non-interactive model indeed interaction was shown not to be a necessary ingredient of zero-knowledge proofs. It was replaced by a shared public (short) string of random bits. The motivation for such a model is the availability of public random sources. For instance, a community in which everyone possess (in the local library) a copy of the same tables of random numbers prepared by RAND corporation, the RAND tables. This is essentially a short string *shared* by the community.

The problem with the original NIZK:

However, the NIZK notion as defined has a major shortcoming. The NIZK proofs by various users rely on the same string and therefore *are not independent of each other*. Thus the problem was formulated and solved first for one prover (or a small logarithmic number of provers).

The underlying technical problem of NIZK seems to be the fact that the simulation which is used in the zero-knowledge proof is very much dependent on the theorems to be proved by the system.

An "imaginary-life" scenario: The problem can be exemplified by the following problem faced by the "Society of Isolated Ancient Greek Mathematicians". In the Aegean islands live various talented mathematicians, residing in different islands: (Xenocrates the Mykonian, Erastosthenes from Lesbos, Ptolemy the Santorinian, Pappus the Rhodian, Hippocrates from Crete, and so on..). All these mathematicians know the last statement issued by the Oracle of Delphi (which is the only common reference string they have, which is, of course, random). The mathematicians are all working on the same set of open problems, sweating to get proofs. Naval transportation is quite rare and limited, thus interactive communication is practically impossible. The serious problem that any member of the society of Ancient Greek Mathematicians has is how to spread the fact that he possesses a recently discovered proof to a theorem, send it to others, and still claim it to himself. The obvious way to do it is to write down a letter and to send it with the next ship leaving for Athens. The letter will be publicly posted in the Agora and will convince everyone reading it that indeed a proof was found by the sender. However, if all details are published anyone can claim to have found the proof. Can the society as a whole send such convincing letters in such a way that no layman (or more importantly no other mathematician), seeing some proofs posted, will be able to get some details about the proof and thus may be able to reconstruct one of them and

claim it independently. By so doing, gaining an undeserved membership in the honorary society of "Somewhat Important Greek Mathematicians" (Sigma?)?

The current applicability of the NIZK procedure is limited. It cannot help the society of the Greek mathematicians, unless it is extended (as its interactive ZKIP counterpart) to enable polynomially many provers (Ancient Greek mathematicians) to use the shared string to prove various theorems in a zero-knowledge fashion. This open question of many-provers non-interactive zero-knowledge (MP-NIZK) proofs was posed by Silvio Micali (see [DeMiPe1]) and is essentially open from 1986.

Previous attempts to overcome the problem:

We remark that the difficulty of achieving the many-prover solution has led to the natural process of modifications of the model and relaxations of the security assumptions in order to achieve (at least in some weaker sense) a many-prover system. Bellare and Micali [BeMi] presented a model in which first there is an initialization of public files by the various users interacting with a trusted server, and then non-interactive proofs by the various users (which is non-transferable, unlike the original model) are possible. Feige and Shamir [FeSh], on the other hand, have defined the notion of Witness-Hiding (WH), and under which the system of [BlDeMiPe] can be used by many provers. (Note that Witness-Hiding (WH) seems weaker (i.e., a stronger secrity assumption) than the notion of Zero-Knowledge (ZK): since ZK implies WH (see lemma 2.1 below), and on the other hand, it is not clear whether under WH one can directly solve a problem like chosen-ciphertext secure Public-key systems, which was finally solved in the recent work [NaYu] by applying NIZK as was advocated in [BlFeMi]). The possibility of implementing the notion of Many Prover-NIZK remained a puzzle. It is interesting theoretically and has a large set of useful applications as efficient prover can apply it as well.

Our results:

We develop two new non-interactive tools. The first one is a "Metaproof system". A meta-proof in terms of proof-theory is *a proof that there is a proof for a theorem.* Indeed the development of deductive systems and the notion of system which proves theorem and then another logical metasystem which proves theorems about theorems, is one of the major development of foundations of mathematics in the last century. However, for a language in $\mathcal{NP}$, in the usual sense a proof of an existence of a proof is just another way (possibly a weird one) to claim that the original theorem is in the language by claiming that there is a proof for it. Nevertheless, in the context of the theory of zero-knowledge proofs, we will show that this way of proving the existence of a proof is a very useful tool.

Intuitively, proving directly a theorem can indeed be done in a zero-knowledge fashion, but the NIZK proof system would be "too exhibitionistic" for our purposes. The additional meta layer (the μ-NIZK system) will provide the required cover. The exact advantage will be explained later.

The second tool presented is "On-line simulatable NIZK scheme". Proving zero-knowledgeness is done by exhibiting a simulator which without having special computational power can generate instances which look like the actual proof. In the "on-line" system, the zero-knowledge simulation seems stronger than the original definition. The simulator first prepares a random string and then the theorem(s) is/are given and a proof is generated on-line. In the original formulation of the property of NIZK, the simulator was given the theorem(s) off-line. This enables a proof system with a temporal constraint in which there are "sleepy mathematicians" each with a claim for a proof of a theorem. One of them wakes up and presents a zero-knowledge proof. We want to simulate this proof fast – before the next mathematician wakes up, so we have to be ready to simulate it on-line. This technical improvement of the definition (to require on-line simulation) is crucial in enabling solution to the MP-NIZK.

In this work we finally eliminate the inferiority of the non-interactive zero-knowledge notion and show how a community of many provers can indeed apply it and the resulting proofs are all together zero-knowledge. The solution combines the above tools developed in this work. As

motivated above, the possibility of MP-NIZK has major implication to the applicability of the NIZK model.

Finding the minimal needed complexity assumptions for cryptographic primitive is an important area of research. The solution to MP-NIZK gives a condition under which a bounded (single theorem) NIZK proof-system can be extended to an (unbounded multi-proof) NIZK proof-system (and the other systems we present). The condition is the existence of any one-way function. This extension is a useful construction which tries to minimize the requirements and to add modularity to the connection of bounded (single theorem) and unbounded (multi-theorem) NIZK schemes. The construction can be applied by a polynomial-time user, which is an important fact for applications. The original construction of general (unbounded) NIZK used a specific mechanism relying on a specific underlying hard number-theoretic problem. Our construction may yield a strategy for reducing complexity assumption for NIZK (implemented in [BlDeMiPe]), by attacking the more limited bounded (single theorem) NIZK (like the ones implemented in [BlFeMi, DeMiPe1, BlDeMiPe]).

A variety of applications of μ-NIZK and MP-NIZK will be presented; application to encryption schemes, and numerous cryptographic protocols and other primitives. Among our applications are extensions and improvements of the nice set of cryptographic applications suggested by the paradigm of Bellare and Goldwasser [BeGo], such as history-independent signature schemes and identification schemes.

The organization of the paper:

The rest of the paper is organized as following. Preliminaries will be presented in section 2. In section 3 we give the definition and implementation of the metaproof system while in section 4 the on-line simulatable system is given. In section 5 we present the MP-NIZK system. Applications are presented in section 6.

2 Preliminaries

2.1 Basic definitions.

Notations. We denote by $\mathcal{N}$ be the set of the natural numbers. If $n \in \mathcal{N}$, by 1^n we denote the concatenation of n 1's. We identify a binary string σ with the integer x whose binary representation (with possible leading zeroes) is σ.
By the expression $|x|$ we denote the length of x if x is a string, the length of the binary string representing x if x is an integer, the absolute value of x if x is a real number, or the cardinality of x if x is a set.
If σ and τ are binary strings, we denote their concatenation by either $\sigma \circ \tau$ or $\sigma\tau$.
A language is a subset of $\{0,1\}^*$. If L is a language and $k > 0$, we set $L_k = \{x \in L : |x| \leq k\}$.

Models of computation. An algorithm is a Turing machine. An *efficient* algorithm is a probabilistic Turing machine running in expected polynomial time.
We emphasize the number of input received by an algorithm as follows. If algorithm A receives only one input we write "$A(\cdot)$", if it receives two inputs we write "$A(\cdot,\cdot)$" and so on.
A sequence of probabilistic Turing machines $\{T_n\}_{n\epsilon\mathcal{N}}$ is an *efficient non-uniform algorithm* if there exists a positive constant c such that, for all sufficiently large n, T_n halts in expected n^c and the size of its program is $\leq n^c$. We use efficient non-uniform algorithms to gain the power of using different Turing machines for different input lengths. For instance, T_n can be used for inputs of length n. The power of non-uniformity lies in the fact that each Turing machine in the sequence

may have "wired-in" (i.e. properly encoded in its program) a small amount of special information about its own input length.

Algorithms and probability spaces. If $A(\cdot)$ is a probabilistic algorithm, then for any input x, the notation $A(x)$ refers to the probability space that assigns to the string σ the probability that A, on input x, outputs σ. Throughout this paper we do not mention explicitly the random coins used by probabilistic algorithms.
Following the notation of [GoMiRi], if S is a probability space, then "$x \stackrel{R}{\leftarrow} S$" denotes the algorithm which assigns to x an element randomly selected according to S. If F is a finite set, then the notation "$x \stackrel{R}{\leftarrow} F$" denotes the algorithm which assigns to x an element selected according to the probability space whose sample space is F and uniform probability distribution on the sample points.
If $p(\cdot,\cdot,\cdots)$ is a predicate, the notation $Pr(x \stackrel{R}{\leftarrow} S; y \stackrel{R}{\leftarrow} T; ... : p(x,y,\cdots))$ denotes the probability that $p(x,y,\cdots)$ will be true after the ordered execution of the algorithms $x \stackrel{R}{\leftarrow} S$, $y \stackrel{R}{\leftarrow} T, ...$
The notation $\{x \stackrel{R}{\leftarrow} S; y \stackrel{R}{\leftarrow} T; \cdots : (x,y,\cdots)\}$ denotes the probability space over $\{(x,y,\cdots)\}$ generated by the ordered execution of the algorithms $x \stackrel{R}{\leftarrow} S$, $y \stackrel{R}{\leftarrow} T, \cdots$.

A notion we use here is *a history-insensitive algorithm*. It is a probabilistic Turing machine R which has *one-way input tape*, *one-way output tape*, *one-way random tape*, and a regular *work tape*. A one-way tape is a tape in which after each read/write operation the head moves, always from left to right. R is called a history-insensitive algorithm if it works as follows. First, after copying the input to its work tape, R produces the output and writes it on the output tape. Then, R erases its work tape and returns to its initial state, without backtracking the one-way heads.

2.2 Bounded Non-Interactive Zero-knowledge Proof Systems

Bounded NIZK proof system were conceived by Blum, Feldman, and Micali, and were presented in [BlFeMi], [DeMiPe1], and [BlDeMiPe]. The term "Bounded" refers to the fact that the proof system is defined for a single theorem or a few "short" theorems.

The following notations and definitions are adopted from [BlDeMiPe].[1] Without loss of generality we use a complete language in $\mathcal{NP}$: the satisfiability of conjunctive normal-form boolean expressions with three literals per conjunct, called $3SAT$ [Co].

Definition 2.1 Let A_1 and A_2 be Turing Machines. We say that (A_1, A_2) is a *sender–receiver* pair if their computation on a *common input* x works as follows. First, algorithm A_1, on input x, outputs a string m_x. Then, algorithm A_2, computes on inputs x and m_x and outputs ACCEPT or REJECT.
A_1 is called the sender and A_2 the receiver. The running times of both machines is calculated in terms of the common input.

Thus, m_x can be interpreted as a message sent by A_1 to A_2.

Definition 2.2 Let (*Prover, Verifier*) be a sender–receiver pair, where *Prover* is history-insensitive and *Verifier* is polynomial-time. We say that (*Prover, Verifier*), is a Bounded Non-Interactive Zero-Knowledge Proof System (Bounded NIZK proof system) for $3SAT$ if there exists a positive constant c such that:

[1] We slightly modify the original definition for ease of exposition of the current paper, but we note that they are equivalent.

1. *Completeness.* $\forall \Phi \in 3SAT$ and satisfying assignments t,

$$Pr(\sigma \stackrel{R}{\leftarrow} \{0,1\}^{n^c}; \mathit{Proof} \stackrel{R}{\leftarrow} \mathit{Prover}(\sigma, \Phi, t) : \mathit{Verifier}(\sigma, \Phi, \mathit{Proof}) = 1) = 1.$$

2. *Soundness.* For all probabilistic algorithms *Adversary* outputting pairs (Φ, Proof), where $\Phi \notin 3SAT_n$, $\forall d > 0$, and all sufficiently large n,

$$Pr(\sigma \stackrel{R}{\leftarrow} \{0,1\}^{n^c}; (\Phi, \mathit{Proof}) \stackrel{R}{\leftarrow} \mathit{Adversary}(\sigma) : \mathit{Verifier}(\sigma, \Phi, \mathit{Proof}) = 1) < n^{-d}.$$

3. *Zero-Knowledge.* There exists an efficient algorithm S such that $\forall \Phi \in 3SAT_n$, for all satisfying assignments t for Φ, for all efficient non-uniform (distinguishing) algorithms D, $\forall d > 0$, and all sufficiently large n,

$$\left| Pr(s \stackrel{R}{\leftarrow} View(n, \Phi, t) : D_n(s) = 1) - Pr(s \stackrel{R}{\leftarrow} S(1^n, \Phi) : D_n(s) = 1) \right| < n^{-d},$$

where

$$View(n, \Phi, t) = \{\sigma \stackrel{R}{\leftarrow} \{0,1\}^{n^c}; \mathit{Proof} \stackrel{R}{\leftarrow} \mathit{Prover}(\sigma, \Phi, t) : (\sigma, \mathit{Proof})\}.$$

We call algorithm S the *Simulator*.

A sender–receiver pair (*Prover, Verifier*) is a Bounded Non-Interactive Proof System for $3SAT$ if there exists a positive constant c such that completeness and soundness hold (such a c will be referred as the *constant* of (*Prover, Verifier*)).
We call the "common" random string σ, input to both *Prover* and *Verifier*, the *reference string*. (Above σ and Φ are the common input.)
In the above definition, there is no limitation on the running time of *Prover*. In cryptographic applications it is required that the prover be expected polynomial time. A Bounded NIZK proof system (*Prover, Verifier*) with an efficient prover is a Bounded NIZK proof system where *Prover* on any common input (i.e. a reference string and a formula) runs in expected polynomial time.

Remark 2.1 Notice that an equivalent definition of *Completeness* is: For all probabilistic algorithms *Choose-in-*$3SAT(\cdot)$ that, on input a n^c-bit string, return elements in $3SAT_n$ along with a satisfying assignment, $\forall d > 0$, and all sufficiently large n,

$$Pr(\sigma \stackrel{R}{\leftarrow} \{0,1\}^{n^c}; (\Phi, t) \stackrel{R}{\leftarrow} \mathit{Choose\text{-}in\text{-}L}(\sigma); \mathit{Proof} \stackrel{R}{\leftarrow} \mathit{Prover}(\sigma, \Phi, t) : \mathit{Verifier}(\sigma, \Phi, \mathit{Proof}) = 1) > 1 - n^{-d}.$$

In fact, if the above condition holds, whenever the proof generated by the prover is not accepted by the verifier (something that the prover can compute) an $\mathcal{NP}$ witness may be provided instead. This will not affect the zero-knowledge property, as it happens with negligibly small probability.

Remark 2.2 Bounded NIZK proof systems with efficient prover exist if it is computationally infeasible to distinguish between quadratic and non quadratic residues modulo composite numbers which are product of two primes (see [BlDeMiPe]).

The following lemma is easily derived from the zero-knowledge property, but we explicitly state it for future reference.

Lemma 2.1 *Let (Prover,Verifier) be a bounded NIZK proof system. Then, for all efficient non-uniform algorithms D, for all $\Phi \in 3SAT_n$ and two satisfying assignments t and t' for Φ, $\forall d > 0$, and all sufficiently large n,*

$$\left|Pr(s \stackrel{R}{\leftarrow} View(n,\Phi,t) : D_n(s) = 1) - Pr(s \stackrel{R}{\leftarrow} View(n,\Phi,t') : D_n(s) = 1)\right| < n^{-d}.$$

Proof. Assume there is a constant $d > 0$, an infinite subset $\mathcal{I} \in \mathcal{N}$, a sequence $\{\Phi_n\}_{n\epsilon\mathcal{I}}$ where $\Phi_n \in 3SAT_n$, a sequence $\{(t_n^1, t_n^2)\}_{n\epsilon\mathcal{I}}$ where t_n^1 and t_n^2 are two satisfying assignments for Φ_n, and an efficient non-uniform algorithm $D = \{D_n\}_{n\epsilon\mathcal{I}}$ such that for all $n \in \mathcal{I}$,

$$|PV^1(n) - PV^2(n)| \geq n^{-d},$$

where $PV^i(n) = Pr(s \stackrel{R}{\leftarrow} View(n,\Phi_n,t_n^i) : D_n(s) = 1)$, $i = 1, 2$.

The algorithm D can be used to distinguish between the view of the verifier and the output of the simulator. Indeed, let S be a simulator for (*Prover,Verifier*), Then, for each $n \in \mathcal{I}$ there is an $i = i(n) \in \{1, 2\}$ such that

$$|PV^i(n) - Pr(s \stackrel{R}{\leftarrow} S(1^n,\Phi_n) : D_n(s) = 1)| \geq 1/(2n^d),$$

violating the zero-knowledgeness of (*Prover,Verifier*). ∎

This proves that a zero-knowledge property implies a property suggested by Feige and Shamir which they call Witness-Hiding [FeSh]. This property seems to be a relaxation of zero-knowledge.

The next lemma shows that the property of zero-knowledge is independent of the choice of the formula. This simple technical fact is used later on.

Lemma 2.2 *Let (Prover,Verifier) be a bounded NIZK proof system and S be a simulator. Then, for all efficient non-uniform algorithms D, for all probabilistic algorithms $Gen(\cdot)$ that on input n, return elements in $3SAT_n$ along with a satisfying assignment, $\forall d > 0$, and all sufficiently large n,*

$$\Big|Pr((\Phi,t) \stackrel{R}{\leftarrow} Gen(n); s \stackrel{R}{\leftarrow} View(n,\Phi,t) : D_n(s) = 1) - Pr((\Phi,t) \stackrel{R}{\leftarrow} Gen(n); s \stackrel{R}{\leftarrow} S(1^n,\Phi) : D_n(s) = 1)\Big| < n^{-d}.$$

Proof. Assume there is a constant $d > 0$, an infinite subset $\mathcal{I} \in \mathcal{N}$, a probabilistic algorithm $Gen(\cdot)$, and an efficient non-uniform algorithm $D = \{D_n\}_{n\epsilon\mathcal{I}}$ such that for all $n \in \mathcal{I}$,

$$\Big|Pr((\Phi,t) \stackrel{R}{\leftarrow} Gen(n); s \stackrel{R}{\leftarrow} View(n,\Phi,t) : D_n(s) = 1) - Pr((\Phi,t) \stackrel{R}{\leftarrow} Gen(n); s \stackrel{R}{\leftarrow} S(1^n,\Phi) : D_n(s) = 1)\Big| > n^{-d}.$$

Above inequality can be rewritten as

$$\sum_{(\Phi,t)} Pr(r \stackrel{R}{\leftarrow} Gen(n) : r = (\Phi,t))\Big|Pr(s \stackrel{R}{\leftarrow} View(n,\Phi,t) : D_n(s) = 1) - Pr(s \stackrel{R}{\leftarrow} S(1^n,\Phi) : D_n(s) = 1)\Big| > n^{-d}.$$

Then, for each $n \in \mathcal{I}$ there is a formula $\Phi_n \in 3SAT_n$ and a satisfying assignment t_n such that

$$\left|Pr(s \stackrel{R}{\leftarrow} View(n,\Phi_n,t_n) : D_n(s) = 1) - Pr(s \stackrel{R}{\leftarrow} S(1^n,\Phi_n) : D_n(s) = 1)\right| > n^{-d}.$$

This violates the zero-knowledgeness of (*Prover,Verifier*). ∎

2.3 Secure probabilistic encryption schemes

Another basic tool we use is the following.

Definition 2.3 A *secure probabilistic encryption scheme* is a efficient Turing machine E that, on input b and internal coin tosses r, outputs an encryption $E(b,r)$, such that

1. {The ciphertext can be uniquely decoded.}

   Whatever are the coin tosses r, s and the inputs b_1, b_2, then $E(b_1,r) = E(b_2,s)$ implies $b_1 = b_2$.

2. {It is computationally hard to distinguish encryptions of 0 from encryptions of 1.}

   Let $E_n(b)$ be the probability space $\{r \stackrel{R}{\leftarrow} \{0,1\}^n : E(b,r)\}$. Then, for all efficient non-uniform algorithms $C = \{C_n\}$, $\forall d > 0$, and for all sufficiently large n,

$$\left| Pr(a \stackrel{R}{\leftarrow} E_n(0) : C_n(a) = 1) - Pr(a \stackrel{R}{\leftarrow} E_n(1) : C_n(a) = 1) \right| < n^{-d}.$$

Goldwasser and Micali [GoMi1] and Yao [Ya] introduced the concept and property 2 is called *indistinguishability of encryption*. If property 2 does not hold, we say that there is a "substantial advantage" in breaking the encryption scheme.

Remark 2.3 Naor [Na] has designed a bit-commitment protocol based on any pseudorandom number generator (that is, assuming any one-way function [Ha, ImLeLu]). The commitment is based on a random challenge given to the commiter. For a random challenge and a random generator, there is no way to cheat in a commit phase. In our model we can implement the encryption function by applying Naor's protocol by choosing the generator (one way function) and the long-enough challenging string *non-interactively* from some portion of the random string (used just for this purpose). All the encryption will, then, use this choice. For more details see [Na].

3 Metaproofs

Roughly speaking, the "metaproof of a theorem T" is a NIZK proof that "there is a NIZK proof of T". Assume that, for a formula Φ, there is a NIZK proof *pf* computed using a reference string σ_1. The *metaprover* μP on input a formula Φ and *pf*, computes a NIZK proof *Metaproof* using a different reference string σ_2, that there is indeed a string *pf* such that *Verifier* would accept as proof of Φ. The *metaverifier* μV checks that *Metaproof* has been correctly computed, but has no access whatsoever to *pf*.

More formally, let (*Prover, Verifier*) be a Bounded NIZK proof system for $3SAT$. Let $Prover(\sigma, \Phi, t)$ be an efficient *Prover*'s program that uses σ as its reference string and the satisfying assignment t to prove $\Phi \in 3SAT_n$. Thus, there is a constant $c > 0$ such that on input $r \in \{0,1\}^{n^c}$, $\Phi \in 3SAT_n$, and a satisfying assignment t, *Prover* computes a string *pf*, $|pf| \leq n^c$, such that $Verifier(r, \Phi, pf) = 1$. Let $L = \bigcup_n L(n)$ be the language where

$$L(n) = \{(r,\Phi) : |r| = n^c,\ \Phi \in 3SAT_n,\ \text{and } \exists pf, |pf| \leq n^c \text{ such that } Verifier(r,\Phi,pf) = 1\}.$$

Then $\Phi \in 3SAT_n$ iff $(r, \Phi) \in L(n)$ for all strings r. Moreover $L \in \mathcal{NP}$ and thus there is a fixed polynomial-time computable reduction $REDUCE$ such that

$$(r, \Phi) \in L(n) \iff \Psi = REDUCE(r, \Phi) \in 3SAT_{n^b}$$

where $b > 0$ is a fixed constant depending only on the reduction $REDUCE$. More precisely, the formula Ψ is obtained by encoding the computation of *Verifier* on input r, Φ, pf as in Cook's Theorem, and then reducing it to a 3-satisfiable formula, as in [Co]. A well known property of this reduction is that to each "witness" *pf* one can associate in polynomial-time a satisfying assignment α for Ψ. We call $Witness(r, \Phi, pf)$ the poly-time procedure that returns the satisfying assignment α for $\Psi = REDUCE(r, \Phi)$.

Now, we formally describe the programs for the metaprover $\mu P(\cdot, \cdot, \cdot)$ and the metaverifier $\mu V(\cdot, \cdot)$

The sender–receiver pair $(\mu P, \mu V)$

Input to μP and μV:

- A random string $\sigma_1 \circ \sigma_2$, where $|\sigma_1| = n^c$ and $|\sigma_2| = n^{bc}$.
- $\Phi \in 3SAT_n$.

Instructions for μP

Private Input: a string *pf* such that $Verifier(\sigma_1, \Phi, pf) = 1$.

$\mu P.1$ Compute $\Psi = REDUCE(\sigma_1, \Phi)$ and $\alpha = Witness(\sigma_1, \Phi, pf)$.

$\mu P.2$ Run $Prover(\sigma_2, \Psi, \alpha)$. Call *Metaproof* the output and send it to μV.

Instructions for μV

Input from μP: a string *Metaproof*.

$\mu V.0$ Compute n from $\sigma_1 \circ \sigma_2$.
Verify that Φ has at most n clauses with 3 literals each. If not, REJECT.

$\mu V.1$ Compute the formula $\Psi = REDUCE(\sigma_1, \Phi)$.

$\mu V.2$ If $Verifier(\sigma_2, \Psi, Metaproof) = 1$ then ACCEPT. Else, REJECT.

Theorem 3.1 *The metaproof system $(\mu P, \mu V)$ above is a Bounded NIZK proof system for $3SAT$.*

The theorem can be verified as it can be shown that $(\mu P, \mu V)$ enjoys the following properties.

1. *Completeness.* $\forall \Phi \in 3SAT$, $\forall \sigma_1$, and $\forall pf$, such that $Verifier(\sigma_1, \Phi, pf) = 1$,

$$Pr(\sigma_2 \stackrel{R}{\leftarrow} \{0,1\}^{n^{bc}}; Metaproof \stackrel{R}{\leftarrow} \mu P(\sigma_1 \circ \sigma_2, \Phi, pf) : \mu V(\sigma_1 \circ \sigma_2, \Phi, Metaproof) = 1) = 1.$$

2. *Soundness.* For all probabilistic algorithms *Adversary* outputting pairs (Φ, pf), where $\Phi \notin 3SAT_n$, $\forall d > 0$, and all sufficiently large n,

$$Pr(\sigma_1 \stackrel{R}{\leftarrow} \{0,1\}^{n^c}; \sigma_2 \stackrel{R}{\leftarrow} \{0,1\}^{n^{bc}}; (\Phi, \mathit{Metaproof}) \stackrel{R}{\leftarrow} \mathit{Adversary}(\sigma_1 \circ \sigma_2) : \mu V(\sigma_1 \circ \sigma_2, \Phi, \mathit{Metaproof}) = 1) < n^{-d}.$$

3. *Zero-Knowledge.* There exists an efficient algorithm S such that $\forall \Phi \in 3SAT_n$, $\forall \sigma_1$, and $\forall pf$, such that $\mathit{Verifier}(\sigma_1, \Phi, pf) = 1$, for all efficient non-uniform (distinguishing) algorithms D, $\forall d > 0$, and all sufficiently large n,

$$\left| Pr(s \stackrel{R}{\leftarrow} View(n, \Phi, pf) : D_n(s) = 1) - Pr(s \stackrel{R}{\leftarrow} S(1^n, \Phi) : D_n(s) = 1) \right| < n^{-d},$$

where

$$View(n, \Phi, pf) = \{\sigma_2 \stackrel{R}{\leftarrow} \{0,1\}^{n^{bc}}; \mathit{Metaproof} \stackrel{R}{\leftarrow} \mu P(\sigma_1 \circ \sigma_2, \Phi, pf) : (\sigma_1 \circ \sigma_2, \mathit{Metaproof})\}.$$

Notice that the metaprover needs to be computationally as strong as *Prover* and not stronger, up to a polynomial factor.

As a non-interactive proof system we can use $(\mu P, \mu V)$ for different theorems with the same reference string. The properties of Completeness and Soundness will still hold. Maintaining the zero-knowledge property is more involved, we need an exact definition for many proofs, however it does not seem that we can infer the zero-knowledge of many theorems from the guaranteed zero-knowledge of a single one.

In the description of μP's program, we did not specify how *pf* is computed. In some applications it is the prover itself that computes *pf*, on input a satisfying assignment t for Φ: for instance, this is the case of the many-provers of Section 5. In other applications the prover gets *pf* as an additional input: for instance, this is the case of the oblivious warden of Section 6.3.

Observe that μP only proves that *pf* is a proof that *Verifier* would accept. He does not prove that *pf* is zero-knowledge! For all the applications we show, the zero-knowledgeness of *pf* is in the interest of the party that computes *pf* and supplies it to μP.

Remark 3.1 Metaproofs can be cascaded to obtain "metametaproofs", "metametametaproofs", and so on, each time increasing the meta-level of the proof.

Remark 3.2 So far the μ-NIZK proof system is merely an alternative to the regular NIZK for $\mathcal{NP}$; the metaprover might as well convince the metaverifier just by forwarding the original proof (which was already zero-knowledge)! Furthermore, an intriguing property is that an efficient metaprover has no knowledge on why the formula Φ is true. Nevertheless he is able to convince the metaverifier that $\Phi \in 3SAT$. This seemingly bizarre property raises further questions on the purpose of defining the metaproof. However, we will show that this is indeed a powerful concept, with many applications.

4 On-line Simulatable Non-Interactive Zero-knowledge Proof Systems

The original notion of Bounded-NIZK was define zero-knowledge by exhibiting a simulator which generates transcript of reference strings and proofs which a polynomial machine cannot tell apart

from a real proof. The simulator was defined as a machine which first gets the theorem to be proved and then starts the computation. Next we define and implement a simulator which works in a different mode. In a preprocessing stage the processor prepare a prefix of a simulation (the reference string); when given a theorem, the simulated proof with respect to the reference string is generated.

This fashion of simulation resembles ideas presented in simulations in [DeMiPe2, ImYu]. It is not clear that it is a stronger definition, however, this simulation mode will be instrumental in constructing a many-provers NIZK proof system using metaproofs.

Definition 4.1 Let (*Prover, Verifier*) be a Bounded NIZK proof system for $3SAT$. A simulator $M(\cdot,\cdot)$ for (*Prover, Verifier*) is an *on-line simulator* if it consists of a pair of efficient algorithms $M = (M_1, M_2)$ that work as follows:

Description of M's program.

First input: 1^n.

Compute: $(\sigma, state) \stackrel{R}{\leftarrow} M_1(1^n)$.

Second input: $x \in 3SAT_n$.

Compute: $Proof \stackrel{R}{\leftarrow} M_2(state, x)$.

Output: $(\sigma, Proof)$.

A Bounded NIZK proof system for $3SAT$ is *on-line simulatable* if it has an on-line simulator.

An overview of the construction.

The idea is to prepare a machinery for the proof based on ciphertexts of an encryption function in an off line fashion, independently of the proof. That is, the statement of the proof is reduced to claims about certain encrypted values. The simulation public string can therefore be prepared independently of the theorem to be prooved. This enables the on-line simulatable proof-system as the public string is independent of the proof itself.

Let (*Prover, Verifier*) be a Bounded NIZK proof system for $3SAT$ with constant c with efficient prover and simulator denoted *Simulator*. Let $Prover(\sigma, \Phi, t)$ be an efficient algorithm that uses σ as its reference string and the satisfying assignment t to prove that $\Phi \in 3SAT_n$.

Let $E(\cdot,\cdot)$ be a secure probabilistic encryption scheme. Define $F = \bigcup_n F(n)$ as the set of all ciphertext pairs of different bit values, formally

$$F(n) = \{(w, \overline{w}) : \exists r, \overline{r} \in \{0,1\}^n, b \in \{0,1\} \text{ such that } w = E(b, r), \overline{w} = E(1-b, \overline{r})\}.$$

Since $F \in \mathcal{NP}$, there is a fixed polynomial-time reduction DIF such that

$$(w, \overline{w}) \in F(n) \iff \Pi = DIF(w, \overline{w}) \in 3SAT_{n^e}$$

where $e > 0$ is some positive constant depending only on the reduction DIF.
Define $T = \bigcup_n T(n)$ as the set of all triplets of ciphertexts such that at least one of them encrypts 1, formally

$$\begin{aligned} T(n) = \{(\alpha, \beta, \gamma) : \exists r_\alpha, r_\beta, r_\gamma, \in \{0,1\}^n, b_\alpha, b_\beta, b_\gamma \in \{0,1\} \text{ such that} \\ \alpha = E(b_\alpha, r_\alpha), \beta = E(b_\beta, r_\beta), \gamma = E(b_\gamma, r_\gamma) \text{ and } b_\alpha \vee b_\beta \vee b_\gamma = 1)\}. \end{aligned}$$

Since $T \in \mathcal{NP}$, there is a fixed polynomial-time reduction RED such that

$$(\alpha, \beta, \gamma) \in T(n) \iff \Psi = RED(\alpha, \beta, \gamma) \in 3SAT_{n^g}$$

where $g > 0$ is some positive constant depending only on the reduction RED.

By the "witness translating" property of $\mathcal{NP}$-reductions, the above two reductions associate in polynomial-time each witness $r, \overline{r}$ $(r_\alpha, r_\beta, r_\gamma)$ with a satisfying assignment t_Π for Π (t_Ψ for Ψ), respectively. The poly-time procedure that returns the satisfying assignment t_Π for $\Pi = DIF(w, \overline{w})$ (t_Ψ for $\Psi = RED(\alpha, \beta, \gamma)$) is called *Witness*-$\Pi(w, \overline{w}, r, \overline{r})$ (*Witness*-$\Psi(\alpha, \beta, \gamma, r_\alpha, r_\beta, r_\gamma)$).

Next we describe a sender–receiver pair (A, B), which then we will prove to be an on-line simulatable NIZK proof system for $3SAT$.

4.1 Description of (A,B)

Input to A and B:

- A random string σ consisting of $\sigma = \rho_1 \cdots \rho_{3n} \tau_1 \cdots \tau_n$, where $|\rho_i| = n^{ec}$ and $|\tau_j| = n^{gc}$.
- A 3-satisfiable formula $\Phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$ over the variables $u_1, ..., u_k$, $k \leq 3n$. (We assume w.l.o.g. that the number of clauses is n, instead of $\leq n$.)

Instructions for A

Private input: a satisfying assignment $t : \{u_1, ..., u_k\} \to \{0, 1\}$ for Φ.

A.1 {Label the formula Φ.}

For each variable u_j, randomly choose two n-bit strings $r_j, \overline{r}_j$ and compute the pairs (u_j, w_j) and $(\overline{u}_j, \overline{w}_j)$, where

$$w_j = E(t(u_j), r_j) \quad \text{and} \quad \overline{w}_j = E(t(\overline{u}_j), \overline{r}_j).$$

{These pairs are the *labeling* of Φ; w_j ($\overline{w}_j$) is the *label* of u_j ($\overline{u}_j$).}

A.2 {Choose proofs of correctness for the labeling of Φ.}

For $i = 1, ..., k$ do:

Compute $\Pi_i = DIF(w_i, \overline{w}_i)$ and $t_{\Pi_i} = \textit{Witness-DIF}(w_i, \overline{w}_i, r_i, \overline{r}_i)$.

Run $\textit{Prover}(\rho_i, \Pi_i, t_{\Pi_i})$ obtaining $\textit{Proof}\,\Pi_i$.

A.3 {Prove that each clause is satisfiable.}

For each clause ϕ_i of Φ do:

Let (α, β, γ) be the labels of the three literals of ϕ_i.

Let $r_\alpha, r_\beta, r_\gamma$ be the coins used in the encryptions α, β, γ, respectively.

Compute $\Psi_i = RED(\alpha, \beta, \gamma)$ and $t_{\Psi_i} = \textit{Witness-RED}(\alpha, \beta, \gamma, r_\alpha, r_\beta, r_\gamma)$.

Run $\textit{Prover}(\tau_i, \Psi_i, t_{\Psi_i})$ obtaining $\textit{Proof}\,\Psi_i$.

Send $\textit{Proof}\,\Phi = (w_1, \overline{w}_1, ..., w_k, \overline{w}_k, \textit{Proof}\,\Pi_1, ..., \textit{Proof}\,\Pi_k, \textit{Proof}\,\Psi_1, ..., \textit{Proof}\,\Psi_n)$.

Instructions for B

Input from A: a string $Proof\Phi$.

B.0 Compute n from σ. Verify that Φ has n clauses with 3 literals each. If not, REJECT.

B.1 {Verify the labeling of Φ.}

For $i = 1, ..., k$ do:

Compute $\Pi_i = DIF(w_i, \overline{w}_i)$. If $Verifier(\rho_i, \Pi_i, Proof\,\Pi_i) \neq 1$ then REJECT.

B.2 {Verify proofs of clauses of Φ.}

For each clause ϕ_i of Φ do:

Let (α, β, γ) be the labels of the three literals of ϕ_i.

Compute $\Psi_i = RED(\alpha, \beta, \gamma)$. If $Verifier(\tau_i, \Psi_i, Proof\,\Psi_i) \neq 1$ then REJECT.

B.3 If all checks have been successfully made then ACCEPT.

4.2 (A,B) is a non-interactive proof system

We show that the properties of completeness and soundness are satisfied.

(A, B) satisfies the completeness requirement. If $\Phi \in 3SAT$ and t is a satisfying assignment for Φ, A can always compute the labels w_j's and $\overline{w}_j$'s. Using the coins r_j's and $\overline{r}_j$'s, A can compute the Π_j's along with the t_{Π_j}'s. Using t_{Π_j}, A can run the efficient procedure *Prover* to get $Proof\,\Pi_j$. Since each clause is satisfied, at least one of its labels is an encryption of 1 and thus A can successfully compute the correspondent t_Ψ and $Proof\,\Psi_i$. All verifications of B are successful if the $Proof\,\Pi_i$'s and the $Proof\,\Psi_i$'s are correctly computed by A.

(A, B) satisfies the soundness requirement. Assume $\Phi \notin 3SAT$. We distinguish two cases. First, suppose that for some j, both w_j and $\overline{w}_j$ are encryption of the same bit. Then, because of the soundness of (*Prover, Verifier*), step B.1 is passed with negligible probability. Second, suppose that all $(w_j, \overline{w}_j)$ are encryptions of different bits but $\Phi \notin 3SAT$. Then, whatever is the labeling (u_j, w_j), $(\overline{u}_j, \overline{w}_j)$,..., (u_k, w_k), $(\overline{u}_k, \overline{w}_k)$ there is a clause ϕ of Φ whose three labels are all encryptions of 0. Thus, step B.2 for that clause is passed with negligible probability, because of the soundness of (*Prover, Verifier*).

4.3 Description of the on-line simulator (M_1, M_2)

Here, we exhibit an on-line simulator $M = (M_1, M_2)$. Then, we prove it satisfies the zero-knowledge requirement.

Instructions for M_1

Input: 1^n

M1.1 {Choose labels.}

For $j = 1, ..., 3n$, randomly choose two n-bit strings $r_j, \overline{r}_j$ and compute

$$w_j = E(1, r_j) \text{ and } \overline{w}_j = E(1, \overline{r}_j).$$

M1.2 {Choose σ and proofs of correctness for the labeling of Φ.}

Randomly choose n strings $\tau_1, ..., \tau_n$, where $|\tau_i| = n^{gc}$.

For $i = 1, ..., 3n$ do:

Compute $\Pi_i = DIF(w_i, \overline{w}_i)$. Run $Simulator(1^n, \Pi_i)$ obtaining ρ_i and $Proof\,\Pi_i$.

M1.3 Set $\sigma = \rho_1 \cdots \rho_{3n} \tau_1 \cdots \tau_n$.

Set $state = (w_1, \overline{w}_1, ..., w_{3n}, \overline{w}_{3n}, r_1, \overline{r}_1, ..., r_{3n}, \overline{r}_{3n}, Proof\,\Pi_1, ..., Proof\,\Pi_{3n}, \sigma)$.

Output: $(\sigma, state)$.

Instructions for M_2

Input: a string *state* and a 3-satisfiable formula $\Phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$.

M2.1 {Label the formula Φ.}

For each variable u_j, construct the pairs (u_j, w_j) and $(\overline{u}_j, \overline{w}_j)$.

{These pairs are the *labeling* of Φ; w_j ($\overline{w}_j$) is the *label* of u_j ($\overline{u}_j$).}

M2.2 {Prove that each clause is satisfiable.}

For each clause ϕ_i of Φ do:

Let (α, β, γ) be the labels of the three literals of ϕ_i.

Let $r_\alpha, r_\beta, r_\gamma$ be the coins used in the encryptions α, β, γ, respectively.

Compute $\Psi_i = RED(\alpha, \beta, \gamma)$ and $t_{\Psi_i} = Witness\text{-}RED(\alpha, \beta, \gamma, r_\alpha, r_\beta, r_\gamma)$.

Run $Prover(\tau_i, \Psi_i, t_{\Psi_i})$. (Call $Proof\,\Psi_i$ the output.)

Set
$Proof\,\Phi = (w_1, \overline{w}_1, ..., w_k, \overline{w}_k, Proof\,\Pi_1, ..., Proof\,\Pi_k, Proof\,\Psi_1, ..., Proof\,\Psi_n)$.

Output: $Proof\,\Phi$.

It is clear that if (M_1, M_2) is a simulator, then it is an on-line simulator.

4.4 (A,B) is zero-knowledge

Theorem 4.1 *If a secure probabilistic encryption scheme and a Bounded NIZK proof system for $3SAT$ with efficient prover exist, then there is a On-line Simulatable Bounded NIZK proof system for $3SAT$.*

Proof. It remains to prove that (A, B) is zero-knowledge, that is showing that $M = (M_1, M_2)$ is a simulator. We proceed by contradiction. Assume that there exist a constant $d > 0$, an infinite subset $\mathcal{I} \subseteq \mathcal{N}$, a set $\{\Phi_n\}_{n\epsilon\mathcal{I}}$ of 3-satisfiable formulae, a set $\{t_n\}_{n\epsilon\mathcal{I}}$, where Φ_n has n clauses and t_n is a satisfying assignment for Φ_n, and an efficient non-uniform algorithm $D = \{D_n\}_{n\epsilon\mathcal{I}}$ such that for all $n \in \mathcal{I}$

$$|P_V(n) - P_S(n)| \geq n^{-d},$$

where $P_V(n) = Pr(s \stackrel{R}{\leftarrow} View(n, \Phi_n, t_n) : D_n(s) = 1)$ and $P_S(n) = Pr(s \stackrel{R}{\leftarrow} S(1^n, \Phi_n) : D_n(s) = 1)$ (recall that *View* is the space of the pairs $(\sigma, Proof)$ where σ is a random string and $Proof \stackrel{R}{\leftarrow} Prover(\sigma, \Phi_n, t_n)$, whereas $S(1^n, \Phi_n)$ is the simulator S's output.)

Now, we use the standard technique of a walking argument among probabilistic spaces. We define a sequence of experiments depending on the output of an efficient non-uniform algorithm $EXP_n(\Phi, t, \textit{label-th}, \textit{pf-lab-th}, \textit{pf-clause-th})$. This algorithm is an hybrid between the simulator M and the prover A. It takes as input a 3-satisfiable formula Φ with n clauses, a satisfying assignment t for Φ, and three "threshold" values. (Notice that the input enables the algorithm to behave as both M and A.) EXP_n labels the first *label-th* literals of Φ as S would do, and the remaining as A would do. Then, EXP_n chooses proofs of correctness as S would do for the first *pf-lab-th* labels, and as A would do for the remaining. Finally, EXP_n proves that each of the first *pf-clause-th* clauses is satisfiable as M would do, and the remaining clauses as A would do.

Now, we formally describe the algorithm EXP_n and its procedures.

Description of $EXP_n(\Phi, t, \textit{label-th}, \textit{pf-lab-th}, \textit{pf-clause-th})$.

$\{\Phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$ is a formula in $3SAT$ over the variables $u_1, ..., u_k$, $k \leq 3n$. t is a satisfying assignment for Φ. $0 \leq$ *label-th* $\leq$ *pf-lab-th* $\leq 3n$. *pf-clause-th* $\in [0, n]$.$\}$

EXP.1 {Choose labels.}

Run $Label(\Phi, t, \textit{label-th})$ obtaining *labeling* and *coins*.

EXP.2 {Choose ρ_i's and proofs of correctness for the labeling of Φ.}

Run $Prove\text{-}labeling(\Phi, t, labeling, coins, \textit{pf-lab-th})$ obtaining $(\rho_1, ..., \rho_{3n}, Proof\,\Pi_1, ..., Proof\,\Pi_k)$.

EXP.3 {Prove that each clause is satisfiable.}

Run $Prove\text{-}clauses(\Phi, t, labeling, coins, \textit{pf-clause-th})$ obtaining $(\tau_1, ..., \tau_n, Proof\,\Psi_1, ..., Proof\,\Psi_n)$.

EXP.4 Set $\sigma = \rho_1 \cdots \rho_{3n} \tau_1 \cdots \tau_n$.

Set $Proof\,\Phi = (w_1, \overline{w}_1, ..., w_k, \overline{w}_k, Proof\,\Pi_1, ..., Proof\,\Pi_k, Proof\,\Psi_1, ..., Proof\,\Psi_n)$.

Output: $(\sigma, Proof\,\Phi)$.

Description of *Label*$(\Phi, t, \textit{label-th})$

$\{\Phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$ is a 3-satisfiable formula over the variables $u_1, ..., u_k$, $k \leq 3n$. t is a satisfying assignment for Φ. *label-th* $\in [0, 3n].\}$

For $j = 1, ..., 3n$ do:

Randomly choose two n-bit strings $r_j, \overline{r}_j$ and compute the pairs (u_j, w_j) and $(\overline{u}_j, \overline{w}_j)$, where

$$\begin{array}{ll} w_j = E(1, r_j) \text{ and } \overline{w}_j = E(1, \overline{r}_j) & \text{if } j \leq \textit{label-th} \\ w_j = E(t(u_j), r_j) \text{ and } \overline{w}_j = E(t(\overline{u}_j), \overline{r}_j) & \text{if } j \geq \textit{label-th} + 1 \end{array}$$

Set *labeling* $= ((u_1, w_1), (\overline{u}_1, \overline{w}_1), ..., (u_k, w_k), (\overline{u}_k, \overline{w}_k))$ and *coins* $= (r_1, \overline{r}_1, ..., r_{3n}, \overline{r}_{3n})$.

Output: (*labeling, coins*).

Description of *Prove-labeling*$(\Phi, t, \textit{labeling}, \textit{coins}, \textit{pf-lab-th})$

{*labeling* and *coins* are output of *Label*$(\Phi, t, \cdot)$.}

For $i = 1, ..., 3n$ do:

Compute $\Pi_i = DIF(w_i, \overline{w}_i)$.

If $i \leq$ *pf-lab-th* then Run *Simulator*$(1^n, \Pi_i)$ obtaining ρ_i and *Proof* Π_i.

Else, Randomly choose ρ_i. Compute $t_{\Pi_i} =$ *Witness-DIF*$(w_i, \overline{w}_i, r_i, \overline{r}_i)$.

Run *Prover*$(\rho_i, \Pi_i, t_{\Pi_i})$ obtaining *Proof* Π_i.

Output: $(\rho_1, ..., \rho_{3n}, \textit{Proof}\,\Pi_1, ..., \textit{Proof}\,\Pi_k)$.

Description of *Prove-clauses*$(\Phi, t, \textit{labeling}, \textit{coins}, \textit{pf-clause-th})$

{*labeling* and *coins* are output of *Label*$(\Phi, t, \cdot)$.}

For each clause ϕ_i of Φ do:

Let (α, β, γ) be the labels of the three literals of ϕ_i. Compute $\Psi_i = RED(\alpha, \beta, \gamma)$.

If $i \leq$ *pf-clause-th* then Run *Simulator*$(1^n, \Psi_i)$ obtaining τ_i and *Proof* Ψ_i.

Else, Let $r_\alpha, r_\beta, r_\gamma$ be the coins used in the encryptions α, β, γ, respectively.

Compute $t_{\Psi_i} = Witness\text{-}RED(\alpha, \beta, \gamma, r_\alpha, r_\beta, r_\gamma)$.

Randomly choose a n^{gc}-bit string τ_i. Run *Prover*$(\tau_i, \Psi_i, t_{\Psi_i})$ obtaining *Proof* Ψ_i.

Output: $(\tau_1, ..., \tau_n, \textit{Proof}\,\Psi_1, ..., \textit{Proof}\,\Psi_n)$.

We feed the distignuisher with transcripts generated by the various experiments. For $n \in \mathcal{I}$, define $PD(n, h, m, q)$ as the probability that D outputs 1 on input an element outputted from EXP_n on input the formula Φ_n, the satisfying assignment t_n, and threshold values h, m, q. Formally,

$$PD(n, h, m, q) = Pr(s \stackrel{R}{\leftarrow} EXP_n(\Phi_n, t_n, h, m, q) : D_n(s) = 1).$$

When *label-th* $= 3n$, *pf-lab-th* $= 3n$, and *pf-clause-th* $= 0$, EXP_n performs the very same computations of the simulator M. Thus,

$$PD(n, 3n, 3n, 0) = P_S(n).$$

When *label-th* $= 0$, *pf-lab-th* $= 0$, and *pf-clause-th* $= 0$, EXP_n performs the very same computations of the prover A. Thus,

$$PD(n, 0, 0, 0) = P_V(n).$$

Using these properties we shall prove the zero-knowledgness of (A, B). Roughly speaking, if the algorithm D distinguishes between M's output and B's view, then the very same algorithm either violates the zero knowledgness of the underlying (*Prover, Verifier*), or it can be used to "break" the encryption scheme E.
Since for all $n \in \mathcal{I}$, $|P_S(n) - P_V(n)| \geq n^{-d}$, then one of the 4 differences

1. $|PD(n, 3n, 3n, 0) - PD(n, 3n, 3n, n)|$, or
2. $|PD(n, 3n, 3n, n) - PD(n, 0, 3n, n)|$, or
3. $|PD(n, 0, 3n, n)) - PD(n, 0, 0, n)|$, or
4. $|PD(n, 0, 0, n) - PD(n, 0, 0, 0)|$,

is $\geq 1/(4n^d)$ for infinitely many $n \in \mathcal{I}$. From claims 1-4 (see below) this violates either the zero-knowledgness of (*Prover, Verifier*) or the security of the encryption E.

Claim 1. For all sufficiently large $n \in \mathcal{I}$,

$$|PD(n, 3n, 3n, 0) - PD(n, 3n, 3n, n)| \leq 1/(4n^d).$$

Proof of Claim 1. By contradiction. Suppose that there is an infinite set $\mathcal{I}' \subseteq \mathcal{I}$ such that for $n \in \mathcal{I}'$, $|PD(n, 3n, 3n, 0) - PD(n, 3n, 3n, n)| \geq 1/(4n^d)$. Then there is a function $j(\cdot)$ with $j(n) \in [0, n-1]$, such that $\forall n \in \mathcal{I}'$,

$$|PD(n, 3n, 3n, j(n)) - PD(n, 3n, 3n, j(n)+1)| \geq 1/(4n^{d+1}).$$

The only difference between $EXP_n(\Phi, t, 3n, 3n, j(n))$ and $EXP_n(\Phi, t, 3n, 3n, j(n)+1)$, is in the $j(n)$-th step of the procedure *Prove-clauses*. In the first, the pair $\tau_{j(n)}$, *Proof* $\Psi_{j(n)}$ is the output of *Simulator*, while in the second the pair consists of a random string and the output of *Prover* on input the formula $\Psi_{j(n)}$ and its satisfying assignment $t_{\Psi_{j(n)}}$. If D is such that $|PD(n, 3n, 3n, j(n)) - PD(n, 3n, 3n, j(n)+1)| \geq 1/(4n^{d+1})$ then, because of Lemma 2.2, it violates the zero-knowledgness of (*Prover, Verifier*).

Claim 2. For all sufficiently large $n \in \mathcal{I}$,

$$|PD(n, 3n, 3n, n) - PD(n, 0, 3n, n)| \leq 1/(4n^d).$$

Proof of Claim 2. By contradiction. Suppose that there is an infinite set $\mathcal{I}' \subseteq \mathcal{I}$ such that for $n \in \mathcal{I}'$, $|PD(n, 3n, 3n, n) - PD(n, 0, 3n, n)| \geq 1/(4n^d)$. Then there is a function $j(\cdot)$ with $j(n) \in [0, 3n-1]$, such that $\forall n \in \mathcal{I}'$,

$$|PD(n, j(n), 3n, n) - PD(n, j(n)+1, 3n, n)| \geq 1/(12n^{d+1}).$$

This implies the existence of an efficient non-uniform algorithm $BREAK$ that, on input a random encryption w, outputs a correct guess of the bit b encrypted by w, with a "substantial advantage". Next is the formal description of the algorithm $BREAK$.

Description of $BREAK_n(w)$.

$\{\Phi_n = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$ is a formula in $3SAT$. t_n is a satisfying assignment for Φ_n. w is either in $E_n(0)$ or in $E_n(1).\}$

BREAK.1 {Choose labels.}

For $i = 1, ..., 3n$ do:

Randomly choose two n-bit strings $r_i, \overline{r}_i$ and compute the pairs (u_i, w_i) and $(\overline{u}_i, \overline{w}_i)$, where

$$\begin{array}{ll} w_i = E(1, r_i) \text{ and } \overline{w}_i = E(1, \overline{r}_i) & \text{if } i \leq j(n) - 1 \\ w_{j(n)} = E(1, r_{j(n)}) \text{ and } \overline{w}_{j(n)} = w & \text{if } i = j(n) \text{ and } t_n(u_{j(n)}) = 1 \\ w_{j(n)} = w \text{ and } \overline{w}_{j(n)} = E(1, \overline{r}_{j(n)}) & \text{if } i = j(n) \text{ and } t(u_{j(n)}) = 0 \\ w_i = E(t_n(u_i), r_i) \text{ and } \overline{w}_i = E(t_n(\overline{u}_i), \overline{r}_i) & \text{if } i \geq j(n) + 1 \end{array}$$

Set $\mathit{labeling} = ((u_1, w_1), (\overline{u}_1, \overline{w}_1), ..., (u_k, w_k), (\overline{u}_k, \overline{w}_k))$ and $\mathit{coins} = (r_1, \overline{r}_1, ..., r_{3n}, \overline{r}_{3n})$.

BREAK.2 {Choose ρ_i's and proofs of correctness for the labeling of Φ.}

Run $\mathit{Prove\text{-}labeling}(\Phi, t, \mathit{labeling}, \mathit{coins}, 3n)$ obtaining $(\rho_1, ..., \rho_{3n}, \mathit{Proof}\Pi_1, ..., \mathit{Proof}\Pi_k)$.

BREAK.3 {Prove that each clause is satisfiable.}

Run $\mathit{Prove\text{-}clauses}(\Phi, t, \mathit{labeling}, \mathit{coins}, n)$ obtaining $(\tau_1, ..., \tau_n, \mathit{Proof}\Psi_1, ..., \mathit{Proof}\Psi_n)$.

BREAK.4 Set $\sigma = \rho_1 \cdots \rho_{3n} \tau_1 \cdots \tau_n$.

Set $\mathit{Proof}\Phi = (w_1, \overline{w}_1, ..., w_k, \overline{w}_k, \mathit{Proof}\Pi_1, ..., \mathit{Proof}\Pi_k, \mathit{Proof}\Psi_1, ..., \mathit{Proof}\Psi_n)$.

Set $SAMPLE = (\sigma, \mathit{Proof}\Phi)$.

If $D_n(SAMPLE) = 1$ then set $b = 1$ else $b = 0$.

If $PD(n, j(n), 3n, n) > PD(n, j(n) + 1, 3n, n)$ then **Output**(b) else **Output**$(1 - b)$.

It is clear that $BREAK$ is an efficient non-uniform algorithm. Moreover, the probability space generated by $SAMPLE$, when $w \stackrel{R}{\leftarrow} E_n(0)$ is equal to the space generated by $EXP_n(\Phi_n, t_n, j(n), 3n, n)$, and when $w \stackrel{R}{\leftarrow} E_n(1)$ it is equal to the space generated by $EXP_n(\Phi_n, t_n, j(n) + 1, 3n, n)$. Hence, $\forall n \in \mathcal{I}'$

$$|Pr(w \stackrel{R}{\leftarrow} E_n(0) : BREAK_n(w) = 1) - Pr(w \stackrel{R}{\leftarrow} E_n(1) : BREAK_n(w) = 1| \geq 1/(12n^{d+1}).$$

This violates the security of E.

Claim 3. For all sufficiently large $n \in \mathcal{I}$,

$$|PD(n, 0, 3n, n) - PD(n, 0, 0, n)| \leq 1/(4n^d).$$

Proof of Claim 3. By contradiction. Suppose that there is an infinite set $\mathcal{I}' \subseteq \mathcal{I}$ such that for $n \in \mathcal{I}'$, $|PD(n,0,3n,n) - PD(n,0,0,n)| \geq 1/(4n^d)$. Then there is a function $j(\cdot)$ with $j(n) \in [0, 3n-1]$, such that $\forall n \in \mathcal{I}'$,

$$|PD(n,0,j(n),n) - PD(n,0,j(n)+1,n)| \geq 1/(12n^{d+1}).$$

The only difference between $EXP_n(\Phi,t,0,j(n),n)$ and $EXP_n(\Phi,t,0,j(n)+1,n)$, is in the $j(n)$-th step of the procedure *Prove-labeling*. In the first, the pair $\rho_{j(n)}$, $Proof\,\Pi_{j(n)}$ (or only $\rho_{j(n)}$ if $j(n) > k$) is the output of *Simulator*, while in the second the pair consists of a random string and the output of *Prover* on input the formula $\Pi_{j(n)}$ and its satisfying assignment $t_{\Pi_{j(n)}}$. If D is such that, for all $n \in \mathcal{I}'$, $|PD(n,0,3n,n) - PD(n,0,0,n)| \geq 1/(4n^d)$, then because of Lemma 2.2, it violates the zero-knowledgness of (*Prover, Verifier*).

Claim 4. For all sufficiently large $n \in \mathcal{I}$,

$$|PD(n,0,0,0) - PD(n,0,0,n)| \leq 1/(4n^d).$$

Proof of Claim 4. By contradiction. Suppose that there is an infinite set $\mathcal{I}' \subseteq \mathcal{I}$ such that for $n \in \mathcal{I}'$, $|PD(n,0,0,0) - PD(n,0,0,n)| \geq 1/(4n^d)$. Then there is a function $j(\cdot)$ with $j(n) \in [0, n-1]$, such that $\forall n \in \mathcal{I}'$,

$$|PD(n,0,0,j(n)) - PD(n,0,0,j(n)+1)| \geq 1/(4n^{d+1}).$$

The only difference between $EXP_n(\Phi,t,0,0,j(n))$ and $EXP_n(\Phi,t,0,0,j(n)+1)$, is in the $j(n)$-th step of the procedure *Prove-clauses*. In the first, the pair $\tau_{j(n)}$, $Proof\,\Psi_{j(n)}$ is the output of *Simulator*, while in the second the pair consists of a random string and the output of *Prover* on input the formula $\Psi_{j(n)}$ and its satisfying assignment $t_{\Psi_{j(n)}}$. If D is such that $|PD(n,0,0,j(n)) - PD(n,0,0,j(n)+1)| \geq 1/(4n^{d+1})$ then, because of Lemma 2.2, it violates the zero-knowledgness of (*Prover, Verifier*).

∎

If a particular implementation of Bounded NIZK proof systems for $3SAT$ with efficient prover is known, then it is possible to construct a more efficient on-line simulatable bounded NIZK proof system for $3SAT$. For example, the simulator in the proof system of [BlDeMiPe] is already an on-line simulator and can be thus used without any modification.

5 Many-provers Non-Interactive Zero-knowledge Proof Systems

Consider a scenario in which we have many independent provers, using the same random string σ to prove different theorems. For instance, a scientific community in which all libraries possess copies of the same tables of random numbers prepared by RAND corporation, the RAND tables. This is essentially a short string *shared* by the scientific community. Can they use the RAND tables to give one another Non-Interactive Zero-Knowledge Proofs? (see [DeMiPe1])

A many-provers NIZK proof system is a solution to this problem. In this section we first formally define the model. Then, we describe how to implement such a system using metaproofs and on-line simulators. Finally, we prove the correctness of our solution.

Definition 5.1 Let (*Prover, Verifier*) be a sender–receiver pair, where *Prover* is history-insensitive and *Verifier* is polynomial time. We say that (*Prover, Verifier*) is a Many-provers Non-Interactive Zero-Knowledge Proof System (MP-NIZK proof system) if the following 3 conditions hold.

1. *Completeness.* $\forall \Phi \in 3SAT$, and for all satisfying assignments t for Φ,

$$Pr\Big(\sigma \stackrel{R}{\leftarrow} \{0,1\}^n; Proof \stackrel{R}{\leftarrow} Prover(\sigma, \Phi, t) : Verifier(\sigma, \Phi, Proof) = 1\Big) = 1.$$

2. *Soundness.* For all probabilistic algorithms *Adversary* outputting pairs $(\Phi', Proof')$, where $\Phi' \notin 3SAT$, $\forall d > 0$, and all sufficiently large n,

$$Pr\Big(\sigma \stackrel{R}{\leftarrow} \{0,1\}^n; (\Phi', Proof') \stackrel{R}{\leftarrow} Adversary(\sigma) : Verifier(\sigma, \Phi', Proof') = 1\Big) < n^{-d}.$$

3. *Zero-Knowledge.* There exists an efficient algorithm S such that $\forall \Phi_1, \Phi_2, ... \in 3SAT$, for all satisfying assignments $t_1, t_2, ...$, for all efficient non-uniform algorithms D, $\forall d > 0$, and all sufficiently large n,

$$\Big|Pr(s \stackrel{R}{\leftarrow} View(n, \Phi_1, t_1, \Phi_2, t_2, ...) : D_n(s) = 1) - Pr(s \stackrel{R}{\leftarrow} S(1^n, \Phi_1, \Phi_2, ...) : D_n(s) = 1)\Big| < n^{-d}$$

where

$$View(n, \Phi_1, t_1, \Phi_2, t_2, ...) = \Big\{\sigma \stackrel{R}{\leftarrow} \{0,1\}^n; \quad \begin{array}{l} Proof_1 \stackrel{R}{\leftarrow} Prover(\sigma, \Phi_1, t_1); \\ Proof_2 \stackrel{R}{\leftarrow} Prover(\sigma, \Phi_2, t_2); \\ \quad \vdots \\ : (\sigma, Proof_1, Proof_2, ...)\Big\}. \end{array}$$

We call *Simulator* the algorithm S.

A sender–receiver pair (*Prover, Verifier*) is a Non-Interactive Proof System for $3SAT$ if Completeness and Soundness hold.

Remark 5.1 An alternative definition of the zero-knowledge property is that there are several independent provers, each using the same algorithm and the same reference string, but its own private random string. Since the prover is history-insensitive these two definitions are equivalent and capture the same notion of many (independent) provers. Furthermore, the *Prover* can be viewed as a publicly available algorithm that can be run with appropriate inputs.

Remark 5.2 The many-provers notion extends that of unbounded theorems of [BlDeMiPe] since it does not require memory. Indeed the prover is modeled as an history-insensitive algorithm and does not have a *random selector* [BlDeMiPe] which gives the same answer (even though random) to the same question (á la oracle), and thus it is history-sensitive.

Remark 5.3 Notice the on-line nature of the proof-system. The definition is with respect to a sequence of theorems which are produced on-line.

5.1 The many-provers protocol (P,V)

This is the first application of metaproofs. When they are combined with an on-line simulatable bounded NIZK proof system, they give a protocol for many-provers NIZK proof systems.

We describe a sender-receiver pair (P, V). P can prove in zero-knowledge the 3-satisfiability of any number of 3-satisfiable formulae with n clauses each. We then employ a technique of [BlDeMiPe] to extend this by showing how to use the same protocol to prove any number of formulae, each of arbitrary size.

Let (*Prover, Verifier*) be a Bounded NIZK proof system for $3SAT$ with efficient prover. Let $Prover(\sigma, \Phi, t)$ be the efficient prover's program that uses σ as its reference string and the satisfying assignment t to prove the $\Phi \in 3SAT_n$.

Recall that there is a constant $c > 0$ such that on input a n^c-bit string r, $\Phi \in 3SAT_n$, and a satisfying assignment t, *Prover* computes a string pf, $|pf| \leq n^c$, such that $Verifier(r, \Phi, pf) = 1$. Let $L = \bigcup_n L(n)$ be the language where

$$L(n) = \{(r, \Phi) : |r| = n^c,\ \Phi \in 3SAT_n,\ \text{and } \exists pf, |pf| \leq n^c \text{ such that } Verifier(r, \Phi, pf) = 1\}.$$

Then recall that $\Phi \in 3SAT_n$ iff $(r, \Phi) \in L(n)$ for all strings r. Moreover $L \in \mathcal{NP}$ and thus there is a fixed polynomial-time computable reduction $REDUCE$ such that

$$(r, \Phi) \in L(n) \iff \Psi = REDUCE(r, \Phi) \in 3SAT_{n^b}$$

where $b > 0$ is a fixed constant depending only on the reduction $REDUCE$. More precisely, the formula Ψ is obtained by encoding the computation of *Verifier* on input r, Φ, pf. With each "witness" pf one can associate in polynomial-time a satisfying assignment α for Ψ. Recall that $Witness(r, \Phi, pf)$ is the poly-time procedure that returns the satisfying assignment α for $\Psi = REDUCE(r, \Phi)$.

5.2 Description of (P,V)

Input to P and V:

- A random string $\sigma_1 \circ \sigma_2$, where $|\sigma_1| = n^c$ and $|\sigma_2| = n^{bc}$.
- $\Phi \in 3SAT_n$.

Instructions for P

P.0 Let t be a satisfying assignment for Φ.

P.1 Run $Prover(\sigma_1, \Phi, t)$. (Call pf the output.)

P.2 Run $\mu P(\sigma_1 \circ \sigma_2, \Phi, pf)$. Call *Proof* the output and send it to V.

Instructions for V

Input from P: a string *Proof*.

V.1 If $\mu V(\sigma_1 \circ \sigma_2, \Phi, Proof)$ =ACCEPT then ACCEPT. Else, REJECT.

5.3 (P,V) is a non-interactive proof system

The completeness is immediate. For the soundness observe that if $\Phi \notin 3SAT$ then there is only a negligible probability that there is a *pf* such that $\mathit{Verifier}(\sigma_1, \Phi, pf) = 1$. Moreover, there is only a negligible probability that there exists a string Proof' such that $\mathit{Verifier}(\sigma_2, \Psi, \mathit{Proof}') = 1$.

5.4 The simulator S

We first describe the program of the simulator S and then prove that it satisfies the zero-knowledge requirement. Let (*Prover, Verifier*) be an on-line simulatable bounded NIZK proof system for $3SAT$ with efficient prover, and let $M = (M_1, M_2)$ be an on-line simulator for (*Prover, Verifier*).

S's program

Input: 1^n, where $n > 0$. A sequence $\Phi_1, \Phi_2, ...$ of 3-satisfiable formulae with at most n clauses each.

S.1 {Choose σ'_1 and σ_2.}

Run $M_1(1^n)$ obtaining σ'_1 and *state*.

Randomly choose a n^{bc}-bit string σ_2.

S.2 {Compute proofs.}

For each input formula Φ do:

Run $M_2(state, \Phi)$ obtaining pf'_Φ.

Compute $\Psi = REDUCE(\sigma'_1, \Phi)$ and $\beta_\Psi = \mathit{Witness}(\sigma'_1, \Phi, pf'_\Phi)$.

Run $\mathit{Prover}(\sigma_2, \Psi, \beta_\Psi)$. (Call Proof_Φ the output.)

Output: $(\sigma'_1 \circ \sigma_2, \mathit{Proof}_{\Phi_1}, \mathit{Proof}_{\Phi_2}, ...)$

Notice that the simulation process is *on line*, that is once the reference string $\sigma'_1 \circ \sigma_2$ has been chosen, then S can produce a proof for each input formula regardless of previous and future formulae. As we shall see, this is an important property for applications.

5.5 (P,V) is zero-knowledge

Theorem 5.1 *If (Prover, Verifier) is an on-line simulatable bounded NIZK proof system for* $3SAT$ *with efficient prover, then* (P, V) *is a many-provers NIZK proof system for* $3SAT$.

Proof. It remains to prove the zero-knowledgeness. To this aim, we show that the output of algorithm S of the previous section cannot be distinguished from the view of the verifier V by any efficient non-uniform algorithm.

We proceed by contradiction. Assume that there exist a constant $d > 0$, an infinite subset $\mathcal{I} \subseteq \mathcal{N}$, a set $\{(\Phi_1, ..., \Phi_{R(n)})\}_{n \epsilon \mathcal{I}}$ of sequences of 3-satisfiable formulae, where Φ_i has n clauses, and an efficient non-uniform algorithm $D = \{D_n\}_{n \epsilon \mathcal{I}}$ such that for all $n \in \mathcal{I}$

$$|P_V(n) - P_S(n)| \geq n^{-d},$$

where $P_V(n) = Pr(s \stackrel{R}{\leftarrow} View(n, \Phi_1, t_1, \Phi_2, t_2, ..., \Phi_{R(n)}, t_{R(n)}) : D_n(s) = 1)$ and $P_S(n) = Pr(s \stackrel{R}{\leftarrow} S(1^n, \Phi_1, ..., \Phi_{R(n)}) : D_n(s) = 1)$.

Let $R(n)$ be a polynomial such that the running time and the size of the program of each algorithm D_n is bounded by $R(n)$. Without loss of generality we can consider $R(n)$-tuples of 3-satisfiable formulae $\Phi_1, ..., \Phi_{R(n)}$, instead of arbitrary sequences of 3-satisfiable formulae $\Phi_1, \Phi_2, ...$

As we have seen in the last section, a main difference between S's output and the view of the verifier is in the first proofs: they are all computed by the algorithm M_2 in S's output, while they are all true proofs in $View$. We will now describe an efficient non-uniform algorithm $EXPR = \{EXPR_n\}_{n \in \mathcal{I}}$. Each $EXPR_n$ takes k, $0 \leq k \leq R(n)$ as input and has "wired-in" the formulae $\Phi_1, ..., \Phi_{R(n)}$ along with their satisfying assignments. Roughly speaking, $EXPR_n$ produces as output a "random" string and "proofs" for all formulae Φ_i's. For the first k formulae it behaves like the simulator, from the $(k+1)$-th and on like the prover. All prior proofs are selected as simulator S does and all subsequent proofs as prover P does. We now formally describe the algorithm:

Description of $EXPR_n(k)$

$\{\Phi_1, ..., \Phi_{R(n)}$ is a sequence of 3-satisfiable formulae with at most n clauses each, and $t_1, ..., t_{R(n)}$ are their satisfying assignments, respectively.$\}$

Input: an integer k such that $0 \leq k \leq R(n)$.

EXPR.1 {Choose σ_1 and σ_2.}

If $k = 0$ randomly choose a n^c-bit string σ_1. Else, run $M_1(1^n)$ obtaining σ_1 and *state*.

Randomly choose a n^{bc}-bit string σ_2.

EXPR.2 {Choose proofs.}

For $i = 1, ..., k$ do:

Run $M_2(state, \Phi_i)$ obtaining pf'_i.

Compute $\Psi_i = REDUCE(\sigma_1, \Phi_i)$ and $\beta_i = Witness(\sigma_1, \Phi_i, pf'_i)$.

Run $Prover(\sigma_2, \Psi_i, \beta_i)$ obtaining $Proof_{\Phi_i}$.

For $i = k+1, ..., R(n)$ do:

Run $Prover(\sigma_1, \Phi_i, t_i)$ obtaining pf_i.

Compute $\Psi_i = REDUCE(\sigma_1, \Phi_i)$ and $\alpha_i = Witness(\sigma_1, \Phi_i, pf_i)$.

Run $Prover(\sigma_2, \Psi_i, \alpha_i)$ obtaining $Proof_{\Phi_i}$.

Output: $(\sigma_1 \circ \sigma_2, Proof_{\Phi_1}, ..., Proof_{\Phi_{R(n)}})$

For $k \in [0, R(n)]$, set

$$PRD(n, k) = Pr(s \stackrel{R}{\leftarrow} EXPR_n(k) : D_n(s) = 1).$$

Both $EXPR_n(0)$ and P perform the same computations, and thus

$$PRD(n, 0) = P_V(n).$$

Moreover, both $EXPR_n(n, R(n)$ and S perform the same computations. Hence,

$$PRD(n, R(n)) = P_S(n).$$

Since $|P_V(n) - P_S(n)| \geq n^{-d}$, $n \in \mathcal{I}$, there is a function $j(\cdot)$ with $j(n) \in [0, R(n) - 1]$ such that $|PRD(n, j(n)) - PRD(n, j(n) + 1)| \geq 1/(n^d R(n))$, $n \in \mathcal{I}$, Since $\mathcal{I}$ is an infinite set, at least one of the two sets $\{j(n) : j(n) \geq 1\}_{n\epsilon\mathcal{I}}$ and $\{j(n) : j(n) = 0\}_{n\epsilon\mathcal{I}}$ must be infinite. If $\{j(n) : j(n) \geq 1\}_{n\epsilon\mathcal{I}}$ is an infinite set then, by Lemma 2.1, this contradicts the zero-knowledgeness of (*Prover, Verifier*). Now, suppose that $\{j(n) : j(n) = 0\}_{n\epsilon\mathcal{I}}$ is an infinite set. This implies an efficient algorithm $DIST_n$ that distinguishes, for all $n \in \mathcal{I}$ such that $j(n) = 0$, between $View(n, \Phi_1, t_1)$ and $S(1^n, \Phi_1)$ in the proof system (*Prover, Verifier*):

Description of $DIST_n(r, proof)$

$\{\Phi_1, ..., \Phi_{R(n)}$ is a sequence of 3-satisfiable formulae with at most n clauses each, and $t_1, ..., t_{R(n)}$ are their satisfying assignments, respectively.$\}$

Input: $(r, proof)$, that is either the output of the simulator M or *Verifier*'s view in (*Prover, Verifier*) when *Prover* wants to prove Φ_1 using t_1.

DIST.1 {Choose σ_1 and σ_2.}

Set $\sigma_1 = r$. Randomly choose a n^{bc}-bit string σ_2.

DIST.2 {Choose proofs.}

Compute $\Psi_1 = REDUCE(\sigma_1, \Phi_1)$ and $\beta_1 = Witness(\sigma_1, \Phi_1, proof)$.

Run $Prover(\sigma_2, \Psi_1, \beta_1)$ obtaining $Proof_{\Phi_1}$.

For $i = 2, ..., R(n)$ do:

Run $Prover(\sigma_1, \Phi_i, t_i)$ obtaining pf_i.

Compute $\Psi_i = REDUCE(\sigma_1, \Phi_i)$ and $\alpha_i = Witness(\sigma_1, \Phi_i, pf_i)$.

Run $Prover(\sigma_2, \Psi_i, \alpha_i)$ obtaining $Proof_{\Phi_i}$.

DIST.3 {Decide whether $(r, proof)$ is an M's output or a *Verifier*'s view.}

Set $SAMPLE = (\sigma_1 \circ \sigma_2,\ Proof_{\Phi_1}, ..., Proof_{\Phi_{R(n)}})$.

Output: $D_n(SAMPLE)$.

It is immediately seen that if $(r, proof)$ is chosen accordingly to *Verifier*'s view or to M's output then $SAMPLE$ has the same distribution of the output of $EXPR_n(0)$ or $EXPR_n(1)$, respectively. Given our assumption on the distinguishing circuit D, for all $n \in \mathcal{I}$ such that $j(n) = 0$, we have

$$\left| Pr(s \stackrel{R}{\leftarrow} View(n, \Phi_1, t_1) : DIST_n(s) = 1) - Pr(s \stackrel{R}{\leftarrow} M(1^n, \Phi_1) : DIST_n(s) = 1) \right| \geq 1/(n^d R(n)).$$

This violates the zero-knowledgeness of (*Prover, Verifier*), which is a Bounded NIZK proof system. ∎

From theorems 4.1 and 5.1 and remark 2.3 we get the following.

Corollary 5.1 *If one-way functions and a Bounded NIZK proof system for* $3SAT$ *with efficient prover exist, then there is a many-provers NIZK proof system for* $3SAT$ *(with efficient prover).*

The above corollary implies that in order to reduce the complexity assumption upon which to implement an MP-NIZK and unbounded NIZK, one may as well reduce the complexity assumption for Bounded NIZK with efficient prover. If we employ the bounded NIZK proof system of [BlDeMiPe] as (*Prover, Verifier*) in the protocol (P, V) (recall that its simulator is already on-line) then from Theorem 5.1 we obtain the following.

Corollary 5.2 *Under the QRA, there is a many-provers NIZK proof system for* $3SAT$.

5.6 Extensions: Proving theorems of unbounded size and Uniform Scenario

Using the same technique of [BlDeMiPe] we can show that we can handle proofs of theorems of unbounded size.

Furthermore, we actually achieve as a special case a multi-theorem NIZK which relies on the existence of a Bounded-NIZK and a secure probabilistic encryption schemes only.

Goldreich [Go] proposed a uniform-complexity treatment of encryption and zero-knowledge. The motivation for this comes from the fact that all parties in a cryptographic setting should be modeled by probabilistic polynomial-time machine that have access only to object generated by procedures with the same power. Our results can be adapted to the uniform setting. We can define uniform NIZK and then we show how our results of previous sections still hold in the new setting; details for such a setting and how the proofs need changed are given in [Go].

6 Further Applications

The primitives presented in this paper can be applied by (polynomial-time) users of a cryptographic system and thus can be applied in cryptographic settings. They give a variety of applications and new tools for secure systems.

6.1 Signatures without history

Based on secure probabilistic encryption schemes and NIZK, Bellare and Goldwasser [BeGo] has suggested a signature scheme which relies on the two tools. One is the NIZK proof systems which is publicly verifiable and the other is pseudo random functions collections. They construct a scheme which is secure against random message attack.

Even, Goldreich, and Micali [EvGoMi] proved that any scheme secure against random message attack (and memoryless) an be transformed into one secure against adaptive chosen message attack (and memoryless).

The scheme using NIZK is not memory-less since the original NIZK, a proof depends on previously given proofs. To prevent history-dependence Bellare and Goldwasser suggest a NIZK which involves initial preprocessing or reliance on a trusted center. Then Feige and Shamir [FeSh] by relaxing the security requirement to Witness-Hiding rather than Zero-Knowledge can apply the protocol without initial interaction. The methods uses an encryption of a seed of a pseudorandom function, and uses non-interactive proofs to show that certain activity related to the message and the encryption is performed correctly; only the signer is able to perform the task. (For more details see [BeGo]).

By using MP-NIZK rather than NIZK (which is history dependent) we finally achieve a signature system secure against adaptive chosen plaintext attack in the paradigm of [BeGo], which is

history-free, preprocessing-free (no trusted server as well), and without relaxation of the original security definition.

The signature schemes based on Universal One-way hash functions (UOWHF), an approach initiated in [NaYu] which yields a trapdoor-less signature possible, in contrast to the original Diffie-Hellman suggestion, are all history-dependent. (See also [ImNa, DeYu1], and finally [Ro] basing UOWHF on any one-way functions).

6.2 Enhance security by sequencing proofs

Another immediate application is cascading a constant number of metaproof systems to enhance security of zero-knowledge schemes. Using a metaproof one can hide the previous proof using a different encryption key. In each level one can use a key and even if only one of the keys is secure, the entire proof is secure (zero-knowledge).

The level of a meta-proof may be used to trace the proof back to its origin if a proof is passed among users.

6.3 Abuse-freeness: the oblivious warden

Desmedt [De] has introduced the notion of protecting the abuse of channels used for cryptographic tasks. This is a classical problem of prevention of abuse of resources and prevention of violations by users. His example is an authentication channel whose users do not care about the authentication, but try to convey extra information in the process of authenticating themselves. Desmedt suggests to protect channels by assigning wardens to monitor and modify information passing through the channel. He gives a nice set of techniques which enables abuse-freeness of protocols. He also suggests to use NIZK by which the sender proves to the warden that it follows the protocol correctly with respect to some initial interactive commitment.

Using the metaproof idea, a NIZK system used by the sender can be made abuse-free by having an oblivious warden (whose task is to prevent violations). The warden simply gets a proof of a theorem, verifies it and forward to the receiver the metaproof, rather then the original proof. He is able to convince the receiver just as well as the original sender, but by sending a proof which is based on his own random bits; its task is independent of the actual proof or NIZK system in use.

6.4 Hierarchical identification.

Another application mentioned in [BeGo] is identification schemes. We extend the notion (using the metaproof system) to enable hierarchical distribution of identification information. The original system enables a center to distribute unforgeable ID numbers. With the metaproof we can implement an hierarchical system in which a center can issue ID's to sub-centers (officers), later the local center can transfer the ID's on. Based on metaproofs level (metaproof, metametaproof, etc.), the user can verify the authenticity of ID's and the level of the officer is giving the ID. This hierarchical center structure is typical to large organizations.

We remark that we have recently investigated the theory of non-interactive protocols based on the new tools presented here [DeYu]. The results will be reported in a forthcoming paper, currently in preparation.

7 Conclusions

We have presented the new tool of "the Metaproof NIZK system". We applied it to solve the multi-prover NIZK system and to reducing complexity assumptions for non-interactive proof systems. We showed further cryptographic applications of the tools.

References

[BaMo] L. Babai and S. Moran, *Arthur–Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes*, Journal of Computer and System Sciences, vol. 36, 1988, pp. 254–276.

[BeGo] M. Bellare and S. Goldwasser, *New Paradigms for Digital Signatures and Message Authentication based on Non-interactive Zero-knowledge Proofs*, Crypto 1989.

[BeMi] M. Bellare and S. Micali, *Non-interactive Oblivious Transfer and Applications*, Crypto 1989.

[BlDeMiPe] M. Blum, A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge Proof Systems*, preprint.

[BlFeMi] M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge Proof Systems and Applications*, Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, 1988.

[Bl] M. Blum, *How to Prove a Theorem So No One Else Can Claim It*, Proceedings of the International Congress of Mathematicians, Berkeley, California, 1986, pp. 1444–1451.

[Co] S. A. Cook, *The Complexity of Theorem-Proving Procedures,* Proc. 3rd Ann. ACM Symp. on Theory of Computing, New York, pp. 151–158.

[De] Y. Desmeth, *Abuse-free Cryptosystems: Particularly Subliminal-Free Authentication and Signature*, preprint.

[DiHe] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, vol. IT-22, no. 6, Nov. 1976, pp. 644–654.

[DePe] A. De Santis and G. Persiano, *Public-Randomness in Public-key Cryptosystems*, preprint.

[DeMiPe1] A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge Proof-Systems*, in "Advances in Cryptology – CRYPTO 87", vol. 293 of "Lecture Notes in Computer Science", Springer Verlag.

[DeMiPe2] A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge Proof-Systems with Preprocessing*, Crypto 1988.

[DeYu] A. De Santis and M. Yung, *General Secure Non-interactive Computations*, in preparation.

[DeYu1] A. De Santis and M. Yung, *On the Design of Provably-Secure Cryptographic Hash Functions*, preprint 1989.

[EvGoMi] S. Even, O. Goldreich, and S. Micali, *On-line/Off-line Digital Signatures*, Crypto 1989.

[FeSh] U. Feige, and A. Shamir, *Witness-Hiding Protocols*, Proceedings of the 22th Annual ACM Symposium on Theory of Computing, 1990 (to appear), announcement in Crypto-89.

[GaJo] M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of $\mathcal{NP}$-Completeness,* W. H. Freeman & Co., New York, 1979.

[Go] O. Goldreich, *A Uniform-Complexity Treatment of Encryption and Zero-Knowledge*, Technical Report no. 568, Technion, June 1989.

[GoGoMi] O. Goldreich, S. Goldwasser, and S. Micali, *How to Construct Random Functions*, Journal of the Association for Computing Machinery, vol. 33, no. 4, 1986, pp. 792–807.

[GoMi1] S. Goldwasser and S. Micali, *Probabilistic Encryption*, Journal of Computer and System Science, vol. 28, n. 2, 1984, pp. 270–299.

[GoMiRa] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof-Systems*, SIAM Journal on Computing, vol. 18, n. 1, February 1989.

[GoMiRi] S. Goldwasser, S. Micali, and R. Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attack*, SIAM Journal of Computing, vol. 17, n. 2, April 1988, pp. 281–308.

[GoMiWi1] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Design*, Proceedings of 27th Annual Symposium on Foundations of Computer Science, 1986, pp. 174–187.

[GoMiWi2] O. Goldreich, S. Micali, and A. Wigderson, *How to Play Any Mental Game*, Proceedings of the 19th Annual ACM Symposium on Theory of Computing, New York, pp. 218–229.

[Ha] J. Håstad, *Pseudorandom Generation under Uniform Assumptions*, Proceedings of the 22th Annual ACM Symposium on Theory of Computing, 1990 (to appear).

[ImLeLu] R. Impagliazzo, L. Levin, and M. Luby, *Pseudo-Random Generation from One-way Functions,* Proceedings of 21st STOC, May 1989.

[ImNa] R. Impagliazzo and M. Naor, *Efficient Cryptographicic Schemes Provably Secure as Subset Sum*, Proceedings of 30th FOCS, 1989.

[ImYu] R. Impagliazzo and M. Yung, *Direct Minimum Knowledge Computations*, in "Advances in Cryptology – CRYPTO 87", vol. 293 of "Lecture Notes in Computer Science", Springer Verlag pp. 40–51.

[Na] M. Naor, *Bit Commitment using Pseudo-randomness*, Crypto 1989.

[NaYu] M. Naor and M. Yung, *Public-key Cryptosystems Probably Secure Against Chosen Ciphertext Attacks*, Proceedings of the 22th Annual ACM Symposium on Theory of Computing, 1990 (to appear).

[Ro] J. Rompel, *One-way functions are Necessary and Sufficient for Secure Signatures*, STOC 90 (to appear).

[Ya] A. Yao, *Theory and Applications of Trapdoor Functions,* Proc. 23rd IEEE Symp. on Foundations of Computer Science, 1982, pp. 80–91.