# Zaps and Their Applications

Cynthia Dwork* Moni Naor†

**Abstract**

A zap is a two-round, witness-indistinguishable protocol in which the first round, consisting of a message from the verifier to the prover, can be fixed "once-and-for-all" and applied to any instance, and where the verifier does not use any private coins. We present a zap for every language in NP, based on the existence of non-interactive zero-knowledge proofs in the shared random string model. The zap is in the *standard* model, and hence requires no common guaranteed random string.

We present several applications for zaps, including 3-round concurrent zero knowledge and 2-round concurrent deniable authentication, in the timing model of Dwork, Naor and Sahai [16], using moderately hard functions [14].

## 1 Introduction

The concept of zero-knowledge, introduced in the ground-breaking paper of Goldwasser, Micali, and Rackoff [26], has proved to be an invaluable tool in the design of cryptographic primitives and protocols. For example, consider an identification protocol based on pseudo-random function evaluation: I am identified by my ability to evaluate a function $f_s$, where only I know the seed $s$. Given a challenge $x$, I produce $y$ and prove that $y = f_s(x)$ critically without revealing any information about $s$.

An appealing and frequently useful relaxation of zero-knowledge, called *witness-indistinguishability*, was proposed by Feige and Shamir [19]. Roughly speaking, in the context of NP, the difference is as follows: An interactive proof system is zero-knowledge if a prover, knowing a witness for membership of a string $x$ in an NP langauge $L$, can correctly "convince" a verifier to accept $x$ while revealing no information whatsoever about the witness. If there are two "independent" witnesses for $x \in L$, a proof system is witness-indistinguishable if the verifier cannot tell which of the two witnesses is being used by the prover to carry out the proof, even if the verifier knows both witnesses. We restrict our attention to NP because we are interested in the realistic setting in which parties are restricted to probabilistic polynomial time computations[1].

In this work we obtain surprising results on the numbers of rounds needed in order to achieve zero-knowledge and witness-indistinguishability. For this purpose we introduce and investigate. zaps. A zap is a two-round witness indistinguishable protocol in which

(i) the first round, consisting of a message from the verifier to the prover, can be fixed "once-and-for-all" and applied to any instance and

*Microsoft Corporation, 1065 L'Avenida, Mountain View, CA 94043, USA. E-mail: dwork@microsoft.com. Work performed while the author was at the IBM Almaden Research Center.

†Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. Some of this work performed while at the IBM Almaden Research Center and Stanford University. Supported in part by a grant from the Israel Science Foundation and ONR grant N00014-97-1-0505, Multidisciplinary University Research Initiative "Semantic Consistency in Information Exchange". E-mail: naor@wisdom.weizmann.ac.il.

[1]The literature on these subjects is extensive. See in particular the papers [12, 20, 24], the lecture notes [13], and the textbooks [21, 32].

(ii) the verifier uses only *public coins*.

That is, the system remains sound and witness-indistinguishable even if the statements to be proven are chosen after the first-round message is fixed. Thus, if we think of the participating parties as families of *non-uniform*, rather than uniform, probabilistic polynomial time-bounded Turing machines, the existence of a zap for a language $L$ implies the existence of 1-message witness-indistinguishable proof system for $L$.

Throughout the paper we will distinguish between the *shared random string model*, in which the parties have access to a common *guaranteed random* string, and what we call the *standard* model, in which no such assumption is made. Whenever we refer to noninteractive zero-knowledge proofs (NIZKs), we mean in the shared random string model (the definition of NIZK forces this). We present zaps for every language $L \in \mathrm{NP}$ based on the existence of a noninteractive zero-knowledge proof system for $L$ in the shared random string model. The zap is in the standard model, and hence requires no common guaranteed-random string. Using current NIZK technology this means that zaps can be based on any family of trapdoor permutations.

Not only can zaps be constructed from NIZKs, but the converse holds as well: if every langage in NP has a zap and one-way functions exist, then every language in NP has a NIZK. This result (and its proof) gives a somewhat formal view of zaps, but yields little intuition for why zaps and NIZKs exist at all. Indeed, our first constructions of zaps were not based on NIZKs, but relied on the new notion of a *verifiable pseudo-random bit generator*, or VPRG. Roughly speaking, a pseudo-random sequence is *verifiable* if a party knowing the pseudo-random seed can construct verifiable "proofs" of the bits of the pseudo-random sequence. Moreover, a VPRG with some number $k$ of output bits passes what we call the "$i$th bit test" for all $1 \leq i \leq k$: given proofs of the values of all but the $i$th bit in the sequence, it is computationally infeasible to guess the $i$th bit with a non-negligible advantage over $1/2$. Thus, VPRGs can be viewed as a special case of the verifiable pseudo-random functions (VPRF's) of Micali, Rabin, and Vadhan [33], in which the domain is very small. We give constructions for VPRGs and a relaxation, *approximate VPRGs*. The importance of VPRGs is this: Zaps (and NIZKs) exist if and only if approximate VPRGs exist in the standard model.

We present applications of zaps in several models. Specifically, we construct faster implementations of important cryptographic primitives in each of the standard, timing-based, and resettable models. Although in some cases the absolute improvement in rounds may be modest, the number of rounds that we achieve in each case is within 1 of the best possible. For example, all previous witness-indistinguishable proof systems require at least *three* rounds of communication. Zaps achieve witness indistinguishability in two rounds. The fact that zaps also yield non-uniform one-round witness-indistinguishability suggests that *proving* a lower bound of two rounds is unlikely.

The most interesting application in this paper is in the *timing* model of Dwork, Naor, and Sahai [16], where, using moderately hard functions [14] and timed commitments [7], we obtain 3-round *concurrent* black-box[2] zero knowledge proofs of knowledge for all of NP. A 3-round black-box zero-knowledge protocol with timing (even without concurrency) is interesting in its own right: it is known that in the standard model (no timing) this is impossible to achieve (with negligible soundness error assuming NP $\not\subset$ BPP) [23], while the possibility of concurrency implies that at least $\Omega(\log n/\log\log n)$ rounds are required [10]; thus, adding timing allows us to go well below the lower bounds in the standard model. Recently, using zaps, Dwork and Stockmeyer have obtained 2-round timing-based black-box (concurrent) zero-knowledge interactive proofs under the assumption that certain functions have no fast *auditors*; they also provide a prover-advice based variant for which soundness is absolute (in this variant the prover can have arbitrary computation time) [17]. We note that even in the timing-based model, zero-knowledge proof systems for languages outside of BPP require two rounds of interaction. No such result is known for the bounded-advice model.

We also use zaps to construct 2-round deniable authentication protocols [12, 15, 16, 17]. Intuitively,

[2] A protocol is 'black-box zero-knowledge' if there is a universal simulator, which when given "black box" access to any verifier strategy, is able to produce a simulation of that verifier with the prover. Virtually all zero-knowledge proofs until very recently where black-box (but see [1] for the first example of a protocol which does not fit this category).

deniable authentication is like a signature scheme in that it permits one party to authenticate messages to another party, based on a public key; however, unlike in the case of digital signatures, the authenticating conversation "leaves no trace," for example, it may be simulable, and hence can be *effectively* repudiated.

The relative ease with which we are able to reduce the amount of interaction provides further motivation for the timing model of [16] – in our opinion a more realistic one than the shared guaranteed random string model (see e.g. [11])– and a complexity theory of moderately hard functions [14].

Using zaps and timed commitments we also obtain a different type of improvement on the results in [16, 17]. The timing model requires a mild "$(\alpha, \beta)$" assumption about the relative rates of the clocks of non-faulty processors, and the protocols in [16] require processors (typically, the prover), to wait until an interval of at least $\beta \geq \alpha$ time has elapsed (as measured on the processor's own clock). $\alpha$ and $\beta$ are chosen so as to tolerate actual system and communication delays. The proofs in [16, 17] require the parameters to be set according to the slowest non-faulty processors. Our new techniques permit flexibility in this respect: fast verifiers with good communication links to the prover are not forced to suffer delays due to slower concurrent verifiers.

In the standard model, without timing assumptions, we give a 2-round oblivious transfer protocol based on the quadratic residuouosity assumption and using public keys; without previously established public keys the protocol requires three rounds.

Finally, we consider a model of computation in which the prover's use of randomness is severely restricted, as, for example, in the case of a smart card, in which the prover may have a short embedded truly random seed and read-only memory. Canetti, Goldreich, Goldwasser, and Micali [9] give one formalization, termed *resettable* zero-knowledge (rZK). Informally, a protocol protects a witness (either in the zero-knowledge sense or in the indistinguishability sense) in the resettable model if the protection holds even if the prover may be re-started (reset) many times and forced to repeatedly use the same random tape (the prover may also be re-started using a different, but still random, tape).

Using zaps and timed commitments, we construct a 3-round timing-based rZK proof system for any language in NP. As noted in [9], rZK proofs *cannot* be proofs of knowledge, so, despite the connections between smart-card, resettable, and concurrent zero-knowledge [9, 28], this result is incomparable with our 3-round concurrent-ZK proofs of knowledge.

We also observe that 2-round (and even non-constructive 1-round) resettable witness-indistinguishability is easily obtained from a zap, simply by having the prover's "random" bits in the zap be a pseudo-random function of the verifier's initial message and the input. This improves (both in conceptual and round complexity) upon the 5-round resettable witness-indistinguishability results in [9].

In all our protocols that employ timing, only the verifier needs access to a (local) clock. This is particularly appealing in the resettable case, in which the prover may be a smart card, since the card may not be equipped with a clock.

## 1.1 Outline

In Section 2 we review the definitions of known cryptographic primitives. A formal definition of a zap is given in Section 3. In Section 4 we prove the existential equivalence of zaps (in the standard model) and NIZKs (in the shared random string model). Section 5 defines and constructs verifiable pseudo-random bit generators (VPRGs) and approximate VPRGs, together with a proof that zaps (and hence, by the above-mentioned result, NIZKs) exist if and only if approximate VPRGs exist in the standard model. In Section 7 we discuss the timing-based applications (3-round concurrent zero knowledge and 2-round deniable authentication). Section 6 contains our zap-based oblivious transfer protocol. In Section 8 we discuss uses of zaps in the resettable model of [9]. Finally, open questions are discussed in Section 9.

## 2 Brief Review of Cryptographic Primitives

Throughout this paper, unless otherwise noted, we assume that all "good" parties (the non-faulty prover and verifier) are modeled by uniform probabilistic polynomial time Turing machines. However, our protocols remain sound regardless of the computational power of the prover, and we achieve zero-knowledge against non-uniform probabilistic polynomial time cheating verifiers.

Let $\nu(n)$ denote a function that grows more slowly than the inverse of any polynomial, *i.e.*, for all $c > 0$ there is an $n_0$ such that $\nu(n) < 1/n^c$ for all $n \geq n_0$.

### 2.1 Interactive Proof Systems and Zero Knowledge

The concepts described here are due to Goldwasser, Micali, and Rackoff [26].

An *interactive protocol* is a pair $(P, V)$ of interactive probabilistic polynomial time machines, the *prover* $P$ and the *verifier* $V$, where $P$ has two inputs $x$ and $y$, and $V$ has inputs $x$ and $z$. The protocol $(P, V)$ is an *interactive proof for the language* $L$ if the following two conditions hold (we let $n$ denote the size of $x$):

1. *Completeness*: For all $x \in L$, there exists a $y$ such that

$$\text{Pr}[(P, V)(x, y) \text{ accepts}] > 1 - \nu(n)$$

2. *Soundness*: For all probabilistic polynomial time machines $P^*$, for all $x \notin L$, for all $y$

$$\text{Pr}[(P, V)(x, y) \text{ accepts}] < \nu(n)$$

where the probability spaces are over the coin flips of both machines. The parameter $n$ plays the role of a *security parameter*.

To define zero-knowledge we allow $V$ to have an auxiliary input $z$. Let $(P, V)(x, y, z)$ denote the transcript of the conversation between $P$ and $V$ when $P$'s input is $(x, y)$ and $V$'s input is $(x, z)$. Then $(P, V)$ is zero-knowledge if for every (possibly cheating) verifier $V'$, there exists a (probabilistic polynomial time machine) simulator $S$, such that for all deterministic polynomial size circuit family of tests $T$ and all $x, y$ such that $(P, V)(x, y)$ accepts with all but negligible probability, and for all $z$:

$$|\text{Pr}[T((P, V')(x, y, z))] - \text{Pr}[T(S(x, z))]\,| < \nu(n) \tag{1}$$

where the probability in the first (resp., second) term is over the random choices of $P$ and $V'$ (resp., $S$).

### 2.2 Proofs of Knowledge [19]

Let $L$ be an NP language accepted by a nondeterministic polynomial time Turing machine $M_L$. A computation path is a sequence of nondeterministic choices made by $M_L$. The set of accepting computation paths on input $x \in L$ is the *witness set* of $x$, denoted $w(x)$.

An *interactive proof of knowledge* system for NP language $L$ is a pair of algorithms $(P, V)$ satisfying:

1. Completeness: $\forall x \in L$ , $\forall w \in w(x)$

$$\text{Pr}[(P, V)(x, w) \text{ accepts}] > 1 - \nu(n)$$

where the probability is over the coin flips of $P$ and $V$;

2. Soundness: there exists an expected probabilistic polynomial time knowledge extractor $M$ such that for all (possibly cheating) probabilistic polynomial time prover $P'$ and $\forall x \forall w'$

$$|\Pr[(P', V)(x, w') \text{ accepts}] - \Pr[M(x, P'(x, w')) \in w(x)]| < \nu(n)$$

where the probability is taken over the coin flips of $V$ and $M$.

The soundness condition is very strong: since it holds for all $w'$ we have in particular that $P'$ need not flip coins, since its favorable coin tosses can be incorporated into $w'$. Note that if $x \notin L$, then $w(x) = \emptyset$, so the probability that $V$ accepts is negligible.

An interactive proof of knowledge is zero knowledge if it satisfies the definition of zero knowledge (simulability).

## 2.3 Witness Indistinguishability

The concept of *witness indistinguishability* was proposed by Feige and Shamir [19] as a relaxation of zero-knowledge. Unlike the case with zero-knowledge, witness indistinguishability is closed under parallel and concurrent composition.

**Definition 2.1 (Witness Indistinguishability)** *A proof system $(P, V)$ is witness indistinguishable if for any polynomial time $V'$, for all $x \in L$, for all $w_1, w_2 \in w(x)$, and for all auxiliary inputs $z$ to $V'$, the distribution on the views of $V'$ following an execution $(P, V')(x, w_1, z)$ is non-uniform probabilistic polynomial time indistinguishable from the distribution on the views of $V'$ following an execution $(P, V')(x, w_2, z)$.*

Note that the auxiliary input $z$ can even be the two witnesses $w_1, w_2$. Thus, *even knowing both witnesses*, $V'$ should not be able to distinguish which witness is being used by $P$.

**Theorem 2.1 ([19])** *Every zero-knowledge protocol is witness indistinguishable.*

**Theorem 2.2 ([19])** *Witness indistinguishability is preserved under parallel and concurrent composition of protocols.*

## 2.4 Noninteractive Zero-Knowledge Proof Systems

The following discussion is based on [12, 20, 37]: A (single theorem) non-interactive proof system for a language $L$ allows one party $P$ to prove membership in $L$ to another party $V$ for any $x \in L$. $P$ and $V$ initially share a string $\sigma$ of length polynomial in the security parameter $n$ which is trusted to have been chosen at random. To prove membership of a string $x$ in $L_n = L \cap \{0, 1\}^n$, $P$ sends a message $\pi$ as a proof of membership. $V$ decides whether to accept or to reject the proof $\pi$ as function of $x$ and $\sigma$. Non-interactive zero knowledge proof systems were introduced in [5, 4]. Non-interactive zero-knowledge schemes for proving membership in any language in NP may be based on any trapdoor permutation (see [20, 29]). Assuming a trapdoor permutation on $k$ bits, the length of a proof of a satisfiable circuit of size $M$ (and the size of the shared random string) is $O(Mk^2)$.

We assume that the shared string $\sigma$ is generated according to the uniform distribution on strings of length polynomial in $n$ and $k$, where the polynomial depends on the particular protocol.

Let $L$ be in NP and for any $x \in L$ let $w(x)$ be the set of strings that witness the membership of $x$ in $L$, as described above. For the proof system to be of any use, $P$ must be able to operate in polynomial time if it is given a witness $w \in w(x)$. We call this the *tractability* assumption for $P$. In general $w$ is not available to $V$.

Let $PN(x, w, \sigma)$ be the distribution of the proofs generated by $P$ on input $x$, witness $w$, and shared string $\sigma$. Suppose that $P$ sends $V$ a proof $\pi$ when the shared random string is $\sigma$. Then the pair $(\sigma, \pi)$ is called the "conversation". Any $x \in L$ and $w \in w(x)$ induces a probability distribution $CONV(x, w)$ on conversations $(\sigma, \pi)$ where $\sigma$ is a shared string and $\pi \in PN(x, w, \sigma)$ is a proof.

For the system to be zero-knowledge, there must exist a simulator $Sim$ which, on input $x$, generates a conversation $(\sigma, p)$. Let $Sim(x)$ be the distribution on the conversations that $Sim$ generates on input $x$, let $Sim_U(x) = Sim_U$ be the distribution on the $\sigma$ part of the conversation, and let $Sim_P(x)$ be the distribution on the proof component. In the definitions of [4, 20] the simulator has two steps: it first outputs $Sim_U$ without knowing $x$, and then, given $x$, it outputs $Sim_P(x)$.

**Definition 2.2** *A pair of probabilistic polynomial time machines $(P, V)$ with shared random string $\sigma$ is a* non-interactive zero-knowledge proof system *for the language $L \in NP$ if:*

*Completeness: For all $x \in L_n$, for all $w \in w(x)$ and for random $\sigma$, with overwhelming probability for $\pi \in_R PN(x, w, \sigma)$, we have that $V$ accepts on input $(\sigma, x, \pi)$. The probability is over the choice of the shared string $\sigma$ and the internal coin flips of $P$.*

*Soundness: For all $y \notin L_n$ with overwhelming probability over $\sigma$ and for all $\pi' \in \{0,1\}^*$, we have that $V$ rejects on input $(\sigma, y, \pi')$. The probability is over the choices of the shared string $\sigma$.*

*Zero-knowledge: There is a probabilistic polynomial time machine $Sim$ which is a simulator for the system: For all non-uniform polynomial time distinguishers $T$, for all $x \in L$, and $w \in w(x)$,*

$$|\Pr[T(s) = 1 | s \in_R Sim(x)] - \Pr[\mathcal{T}(s) = 1 | s \in_R \mathcal{CONV}(x, w)]| \quad < \quad \nu(n)$$

*where $\nu(n)$ is a negligible function and where the probability space is taken over the random choices of $\sigma$ and over the random choices of the $Sim$ and $\mathcal{P}$.*

Note that this definition of NIZK does not require that the system be sound if the instance $x$ is chosen after the public random string is known. Nevertheless, it is sufficiently strong for our purposes.

As shown in [20], any NIZK satisfying Definition 2.2 is also *general witness indistinguishable*:

**Definition 2.3** *([20]) A NIZK for language $L$ in NP is general witness indistinguishable if for all polynomial distinguishers $T$ for a random string $\sigma$, for any sequence $\{(x_i, w_i^1, w_i^2)\}_{i=1}^m$ chosen by $T$ where $x_i \in L_n$ and $w_i^1, w_i^2 \in w(x_i)$ for all $1 \leq i \leq m$ we have*

$$|\Pr[T(\pi_1^1, \pi_2^1, \dots \pi_m^1) = 1] - \Pr[T(\pi_1^2, \pi_2^2, \dots \pi_m^2) = 1]| < \nu(n)$$

*where for all $1 \leq i \leq m$ and $b \in \{0,1\}$ we let $\pi_i^b \in_R PN(x_i, w_i^b, \sigma)$. The probability space is over $\mathcal{P}$'s and $T$'s random coins and the choice of $\sigma$.*

Note that general witness indistinguishability implies witness indistinguishability even if $x_1 = \ldots = x_m$, which will be the case of interest here.

## 2.5 Deniable Authentication

A *public key authentication* scheme permits an authenticator $A\mathcal{P}$ to convince a second party $V$, only having access to $A\mathcal{P}$'s public-key, that $A\mathcal{P}$ is willing to authenticate a message $m$. However, unlike in the case of digital signatures, deniable authentication does not permit $V$ to convince a third party that $A\mathcal{P}$ has authenticated $m$ – there is no "paper trail" of the conversation (say, other than what could be produced by $V$ alone). Thus, deniable authentication is incomparable with digital signatures. Deniable authentication first

appeared in [12, 15]; and was formalized in [16] (see also [17]). Several 4-round timed concurrent deniable authentication protocols are given in [16, 17]
The authentication protocol should satisfy:

Completeness: For any message $m$, if the prover and verifier follow the protocol for authenticating $m$, then the verifier accepts.

Soundness – Existential Unforgeability Against Chosen Message Attack: Suppose that the copies of $A\mathcal{P}$ are willing to authenticate any polynomial number of messages $m_1, m_2, \ldots$, which may be chosen adaptively by an adversary $\mathcal{A}$ who also controls the verifier $V'$. We say that $\mathcal{A}$ successfully attacks the scheme if a forger $C$, under control of $\mathcal{A}$ and pretending to be $A\mathcal{P}$, succeeds in authenticating to a third party $D$ (running the protocol of the original verifier $V$) a message $m \neq m_i$, $i = 1, 2, \ldots$. The soundness requirement is that all probabilistic polynomial time $\mathcal{A}$ will succeed with at most negligible probability.

Zero-Knowledge - Deniability: Consider an adversary $\mathcal{A}$ as above and suppose that the copies of $A\mathcal{P}$ are willing to authenticate any polynomial number of messages. Then for each $\mathcal{A}$ there exists a polynomial-time simulator that outputs an indistinguishable transcript.

### 2.6 Security of Encryption

We will need public-key cryptosystems for two of our applications: Resettable Zero-Knowledge (Section 8.2) and Deniable Authentication (Section 7.2). The security requirements of these two applications are different. To specify the security of an encryption scheme one must describe the power of the attacker in terms of access to the system (chosen plaintext, chosen ciphertext) and what it means to break the system (semantic-security, non-malleability). See [12] or [2] for a discussion of notions of security. The deniable authentication application requires a system that is non-malleable against chosen-ciphertext attacks in the post-processing mode (called CCA-2 in [2]). The resettable zero-knowledge application requires semantic security against chosen plaintext attacks (there are some other requirements from the encryption scheme which transcend security).

### 2.7 Using Time in the Design of Protocols

Dwork, Naor and Sahai [16] have shown the power of time in the design of zero-knowledge protocols through the use of an $(\alpha, \beta)$ assumption. This says that all good parties are assumed to have clocks that satisfy the $(\alpha, \beta)$-*constraint* (where $\alpha \leq \beta$): for any two (possibly the same) non-faulty parties $P_1$ and $P_2$, if $P_1$ measures $\alpha$ elapsed time on its local clock and $P_2$ measures $\beta$ elapsed time on its local clock, and $P_2$ begins its measurement in real time after $P_1$ begins, then $P_2$ will finish after $P_1$ does.

The protocols in [16, 17] use time in two explicit ways: (i) Delays: one party must delay the sending of some message until at least some specified time $\beta$ has elapsed on its local clock; (ii) Time-outs: one party requires that the other deliver its next message before some specified time $\alpha$ has elapsed on its (first party's) local clock. In this work we are able to eliminate the use of delays; the protocols only use time-outs. Furthermore we do not require a *global* $(\alpha, \beta)$-constraint, rather each instantiation of the protocol can fix its own values based on the local characteristics of the network. An essential ingredient of our protocols is the *implicit* use of time via *moderately hard functions* [14]. In particular, we use *timed commitments with verifiable recovery,* described next.

**Timed Commitment.** A string commitment protocol allows a sender to commit, to a receiver, to some value. The protocol has two phases. At the end of the *commit* phase the receiver has gained no information about the committed value, while after the *reveal* phase the receiver is assured that the revealed value is indeed the one to which the sender originally committed. Timed commitments, defined and constructed by Boneh and Naor [7], are an extension of the standard notion of commitments in which there is a potential forced opening phase permitting the receiver, by computation of some moderately hard function, to recover the committed value without the help of the committer. The price paid in terms of security is that the committed value is hidden for only a limited amount of time. The important requirements of timed commitments are (i) The future recoverability of the committed value is verifiable: if the commit phase ends successfully, then the receiver is correctly convinced that forced opening will yield the value. (ii) Forcibly recovered values and decommitments are verifiable: the receiver not only obtains the value, but also a proof of its validity, so that anyone who has the commitment (or the transcript of the commit phase) can verify the value *without going through a recovery process*, i.e. in fixed amount of time. (iii) The commitment is immune to parallel attacks, i.e. even if the receiver has much more computing power than assumed, it cannot recover the value substantially more quickly than a single-processor receiver. We denote by $T$ the bound on the time below which it is safe to assume that the timed commitment cannot be broken with non-negligible probability, even by a PRAM.

Specifically, we are interested in timed commitment schemes with the following structure. The committer sends to the receiver a string $\zeta$, which constitutes the commitment. For every "valid" commitment $\zeta$, it is possible, through moderately hard computation, to recover a pair $(y, \pi)$ such that $\pi$ is an easily checked witness to the fact that $\zeta$ is a commitment to $y$. The set of valid commitments is in NP: For every valid commitment $\zeta$ there is a witness to the statement "$\zeta$ is a valid commitment to a string that can be recovered through the forced recovery process." Finally, the forced recovery time is relatively large compared to the time of all other operations in the protocol (such as, constructing $\zeta$, verifying a correctly decommitted value, verifying future recoverability, etc.) Thus, we think of all other operations as "easy" while recovery is "moderately hard." The scheme in [7] has this structure and properties.

### 2.8 Oblivious Transfer

In a 1-out-of-2 Oblivious Transfer protocol one party, the *sender*, has two strings $(M_1, M_2)$ as its input, and the second party, the *chooser*, has a bit $b$. The chooser should learn $M_b$ and nothing regarding $M_{1-b}$ while the sender should gain no information about $b$. 1-out-2 OT was suggested by Even, Goldreich and Lempel [18], as a generalization of Rabin's "oblivious transfer" [38].

## 3 Formal Definition of a Zap

A zap is a 2-round (2-message) protocol for proving membership of $x \in L$, where $L$ is a language in NP, satisfying the following conditions Letting the first-round (verifier to prover) message be denoted $\rho$ and the second-round (prover to verifier) response be denoted $\pi$:

Completeness: Given $x$, a witness $w \in w(x)$, and a first-round $\rho$, the prover, running in time polynomial in $|x|$, can generate a proof $\pi$ that will be accepted by the verifier. The probability is over the choices made by the prover and the verifier. Moreover, the verifier's decision whether to accept or reject is a polynomial time function of $x$, $\rho$, and $\pi$. In other words, a zap is a *public-coin* protocol.

Soundness: With overwhelming probability over choice of $\rho$, there exists no $x' \notin L$ and round-2 message $\pi$ such that the verifier accepts $(x', \rho, \pi)$.

Witness-Indistinguishability: Let $w, w' \in w(x)$ for $x \in L$. Then $\forall \rho$, the distribution on $\pi$ when the prover has input $(x, w)$ and the distribution on $\pi$ when the prover has input $(x, w')$ are nonuniform probabilistic polynomial time indistinguishable.

It follows immediately from the definitions that a zap yields a *non-constructive* non-uniform single round witness-indistinguishable protocol; informally, the first-round message, $\rho$, can be fixed once and for all, without hurting any of the completeness, soundness, or witness-indistinguishable properties. More rigorously, for each $n$, there exists a string $\hat{\rho}_n$ such that, letting $L_n = L \cap \{0,1\}^n$,

1. Given $x \in L_n$ and a witness $w \in w(x)$, the prover can generate a proof $\pi$ that will be accepted by the verifier. Moreover, the verifier's decision whether to accept or reject is a polynomial time function of $x, \hat{\rho}_n$, and $\pi$.
2. There exists no $x' \notin L_n$ and message $\pi$ such that the verifier accepts $(x', \hat{\rho}_n, \pi)$.
3. For all $x \in L_n$ and all $w, w' \in w(x)$, the distributions $\mathcal{P}(x, w, \hat{\rho}_n)$ and $\mathcal{P}(x, w', \hat{\rho}_n)$ are indistinguishable by any non-uniform probabilistic polynomial time distinguisher.

**Comparison with NIZKs.** Zaps differ from non-interactive zero-knowledge proof systems (NIZKs) in two respects, making the two concepts incomparable. On the one hand, zaps do not require that the prover and verifier have access to a common guaranteed random string. On the other hand, NIZKs provide more provable protection of the witness than do zaps, since NIZKs can be simulated without access to the witness while zaps provide no such guarantee.

# 4 The NIZK-Based Construction

Assume we have a NIZK system (in the shared string model) satisfying Definition 2.2 for a language $L$. We will construct a zap for $L$ (in the standard model). We will first provide some intuition for the construction. Consider a NIZK in the shared string model; we try to convert it into a zap by somehow generating the shared string $\sigma$. Suppose we let the verifier choose a string $B$ and fix $\sigma = B$. The danger with this approach is that there may be "bad" choices for $\sigma$ that leak information about the witness, and the verifier might choose $B$ to be one of them, thus harming the witness protection. If, to compensate, we have the prover choose its own random string $C$ and we set $\sigma = B \oplus C$ (that is, $\sigma$ is the bit-wise exclusive-or of $B$ with $C$), then the danger is that the prover will use the simulator to come up with a $\sigma'$ that "proves" that $x \in L$ (that is, causing $V$ to accept $x$), even for $x \notin L$. The prover could then set $C = \sigma' \oplus B$, violating soundness.

The actual protocol strikes a balance between these two ideas: a NIZK is repeated many times in parallel, but not quite independently, as follows. The $j$th instance has common string $\sigma_j$, defined to be the bitwise exclusive-or of two strings, one chosen by the prover and the other chosen by the verifier. The verifier's choice for the $j$th instance may be any string $B_j$; however the prover may only choose a single string $C$ that is used in all instances.

**Choice of Parameters (General Construction).** Assume that we have a NIZK for $L$ which, for proving membership of strings of length $|x|$, with security parameter $n$, uses a common shared string of length $\ell = \ell(n, |x|)$. Assume further that on any input $y \notin L$ of length $|x|$ the NIZK errs with probability at most $q = q(n)$. In Equation 2, $k = k(n, |x|) = |\rho|$, the number of random bits sent by the verifier in the first-round message. The number of copies $m = m(n)$ of the NIZK will be $k/\ell$. To achieve soundness guarantee $\delta$ for the zap (that is, a cheating prover should succeed with probability at most $\delta$), we choose $k$ satisfying

$$q^{k/\ell} < \frac{\delta}{2^{|x|+\ell}}. \tag{2}$$

## 4.1 Protocol Z: A Zap

In order to achieve soundness against an arbitrarily powerful prover and yet to require only feasible computation from the "good" prover, we must assume the existence of a NIZK with these properties, such as the systems in [20, 29].

Let $x \in L$ be an NP-statement to be proved to the verifier. We do not need $x$ to be fixed before execution of the protocol begins. Let $w$ be the witness to $x \in L$ known by the prover, let $n$ be the security parameter, and let $\mathrm{PN}(x, w, \sigma)$ be the distribution on messages sent in the NIZK by a (non-cheating) prover when the common random string is $\sigma$ of length $\ell(n, |x|)$. For simplicity, in the remainder of this discussion we assume $x$ and $|x|$ are related by some fixed polynomial so that it suffices to think of $\ell(n, |x|)$ as a function solely of $n$. Let $k = k(n)$ and $m = m(n)$ satisfy Equation 2.

**First Round:** $V \longrightarrow P$ : The verifier sends to the prover a random $k$-bit string $\rho = b_1 \dots b_k$, which is interpreted as $B_1 \dots B_m$, where $B_j$ denotes the $j$th block of $\ell$ consecutive bits and $\ell$ is the length of the common random string used by the NIZK.

**Second Round:** $P \longrightarrow V$ : The prover responds as follows. First, it chooses a random $\ell$-bit string $C = c_1 \dots c_\ell$. For $j = 1 \dots m$ define $\sigma_j$ to be the bitwise exclusive-OR of $B_j$ and $C$:

$$\sigma_j = B_j \oplus C.$$

Then the prover sends to the verifier $x$, $C$, and the $m$ noninteractive proofs

$$\{\pi_j \in_R \mathrm{PN}(x, w, \sigma_j)\}_{j=1\dots m}.$$

**Final Check:** The verifier checks that each of the $m$ NIZKs $\pi_1, \dots, \pi_m$ would result in acceptance; if so, then the verifier accepts the zap; otherwise the verifier rejects. This completes the description of Protocol Z.

**Lemma 4.1** *Protocol Z is sound; moreover, for all $n$, there exists a choice $\hat{\rho_n} = \hat{b}_1 \dots \hat{b}_{k(n)}$ for the first round message that yields perfect soundness: $\forall x \notin L_n \, \forall \pi \; V(x, \hat{\rho_n}, \pi)$ rejects.*

*Proof.* Let $\ell = \ell(n)$ and $k = k(n)$. Fix an $x \notin L$ and random bit string $C = c_1 \dots c_\ell$. We will show that with overwhelming probability, over the choice of $b_1, \dots, b_k$, the prover will fail to convince the verifier to accept $x$. The key point is that once everything but the $b$'s has been fixed, the $\sigma_j$'s are truly random – because the $B_j$'s are. Therefore each copy of the NIZK proof has probability at most $q$ of failing to cause rejection. Since each proof is independent (because the random $b_i$'s used in each copy of the NIZK proof are independent), the overall probability that all $m = k/\ell$ copies fail is at most $q^m$.

The number of possible assignments to the $c$'s, and $x \notin L$ is at most $2^{\ell+|x|}$. Hence, as long as

$$2^{\ell+|x|} q^m = 2^{\ell+|x|} q^{k/\ell} \leq \delta$$

(which is guaranteed by our choice of $k$ in (2)) the probability over $b_1, \dots, b_k$, that there even exists a "bad" choice of $c_1 \dots c_\ell$, an $x \notin L$, and a zap $\pi$ that erroneously causes the verifier to accept $x$, is at most $\delta$ (cf. the soundness requirement in Definition 2.2). Since $\delta < 1$, there must exist some $\hat{\rho_n} = \hat{b}_1 \dots \hat{b}_k$ that provides soundness against all $x \notin L_n$: $\forall x \notin L_n \, \forall \pi \; V(x, \hat{\rho_n}, \pi)$ rejects.

$\diamondsuit$

**Lemma 4.2** *Protocol Z is witness indistinguishable.*

*Proof.*

We prove witness indistinguishability *for every* $\rho$. Thus, fix an arbitrary $\rho$ for the entire proof. We will carry out a standard hybrid argument with the following steps along the chain. Let $w$ and $w'$ be two witnesses that $x \in L$, and let $n = |x|$. At one extreme of the chain the witness $w$ is used in each of the $m$ NIZKs; at the other extreme the witness $w'$ is used in every copy. At each step along the chain we increase by one then number of copies of the NIZK in which $w'$ is used (and decrease the number in which $w$ is used).

Let $\langle w, w', i\rangle$ denote the distribution on transcripts in which the first $i$ copies of the NIZK are consructed using $w$ and the remaining $m - i$ copies are constructed using $w'$. The distribution is over random random choices made by the prover (since $\rho$ is fixed. Let $T$ be a non-uniform polynomial-time test that takes as input a transcript and outputs a single bit. We write $T(\langle w, w', i\rangle)$ to denote $T$'s behavior on a transcript chosen uniformly from $\langle, w, w', i\rangle$.

Assume for the sake of contradiction that there exists a probabilistic polynomial time test $T$ and $1 \le j \le m$ such that for some a fixed $a$ and infinitely many $n$:

$$\Pr[T(\langle w, w', j-1\rangle) = 1] - \Pr[T(\langle w, w', j\rangle) = 1] \ge \frac{1}{n^a}$$

The probability space is over the choices made by the prover and the randomness of $T$. We will show that this contradicts the witness-indistinguishability of the underlying NIZK.

Let $(P^*, V^*)$ be the underlying NIZK protocol (running in the shared random string model). Let $\tau$ be a truly random string of $\ell$ bits. Choose $u \in_R \{w, w'\}$ and give $u$ to $P^*$. Let $P^*$ generate a proof $\pi \in_R \mathrm{PN}(x, u, \tau)$. By the witness-indistinguishability of the NIZK, with overwhelming probability over choice of $\tau$, no non-uniform probabilistic polynomial time machine, even given $w$ and $w'$, has non-negligible advantage of guessing the value of $u$ from $\pi$. We will show how to use $T$ to violate this indistinguishability.

Using $w$ and $w'$, construct a simulated transcript of Protocol Z as follows. Break $\rho = b_1, \ldots, b_k$ into $m = k/\ell$ blocks $B_1, \ldots, B_m$. Set $C = \tau \oplus B_j$, so that $\sigma_j = B_j \oplus C = \tau$, which is truly random by assumption. For all $i < j$, construct $\pi_i \in_R PN(x, w, \sigma_i)$. For all $i > j$, construct $\pi_i \in_R PN(x, w', \sigma_i)$. Set $\pi_j = \pi$ which, by assumption, is a uniformly chosen element of $PN(x, u, \tau)$. Let $\Pi$ denote the resulting transcript.

Run $T$ on $\Pi$. Since $\tau$ is truly random and uniformly distributed, $C$ is uniformly distributed as well, so the resulting transcript of $m$ NIZKs is a uniformly chosen element of either $\langle w, w', j-1\rangle$ (if $u = w'$) or $\langle w, w', j\rangle$ (if $u = w$). We can therefore use $T$'s assumed ability to distinguish these two cases to obtain a non-negligible advantage in guessing whether $u = w$ or $u = w'$. $\diamondsuit$

**Theorem 4.3** *Protocol Z is a zap.*

*Proof.* Completeness is immediate from the completeness of the underlying NIZK system. Completeness for the 1-round non-uniform case follows from the fact that, for any $\hat{\rho}$, since $C$ is random, the probability that there is some block $\hat{B}_i$ such that $\pi \in_R \mathrm{PN}(x, w, \hat{B}_i \oplus C)$ but $V^*(\hat{B}_i \oplus C, x, \pi)$ does not accept, is negligible. (Here, as earlier, $\hat{B}_i$ is the $i$th consecutive block of $\ell$ bits in $\hat{\rho}$.) Soundness, non-uniform soundness, and witness-indistinguishability have been argued. Finally, the proof of witness-indistinguishability in Lemma 4.2 holds for *every* $\rho$, assuming the underlying NIZK enjoys general witness indistinguishability (Definition 2.3), in particular, it holds for the strings $\hat{\rho}_n$ of Lemma 4.1 (the strings providing perfect soundness). $\diamondsuit$

Our main conclusion is therefore:

**Corollary 4.4** *Let $L \in NP$ be arbitrary.*

1. *If there exists a NIZK for $L$ in the common guaranteed-random string model, then there exists a zap for $L$ in the standard model.*
2. *If there exist zaps in the standard model for* every *language in NP, and if there exist non-uniform one-way functions, then there is a NIZK for $L$ in the common guaranteed-random string model.*

*Proof.* The first claim is immediate from the construction and correctness of Protocol Z.

For the second claim we directly apply the idea of Feige, Lapidot, and Shamir [20] of transforming the proof of the statement $x \in L$ into a witness-indistinguishable proof for the statement "the common shared random string $\sigma$ is pseudo-random OR $x \in L$". As we will explain, to carry out this approach it is sufficient to have

- a pseudo-random generator $G$ that, say, doubles the length of the seed (in this case a random string is unlikely to be the output of the generator for any seed) and
- a zap for the language $L' = \{(x, \sigma) | x \in L \text{ or } \exists s \ \sigma = G(s)\}$.

The desired pseudorandom generators exist iff non-uniform one-way functions exists [27]; moreover, since $L'$ is clearly in NP, it has a zap by assumption. We assume for simplicity (and without loss of generality) that the verifier's message in the zap is chosen uniformly at random.

Recall we are trying to show that if one-way functions and zaps exist, then there exists a NIZK in the shared random string model. Given a shared random string, treat it as $(\sigma, \rho)$ where $\rho$ is the verifier's first-round message in the zap for the language $L'$. The prover simply transforms its witness for $x \in L$ to a witness for $(x, \sigma) \in L'$. Soundness follows from the fact that most $\sigma$'s are not equal to $G(s)$ for any $s$ (this follows from the assumption that $G$ is length-doubling and $\sigma$ is truly random).

The system is zero-knowledge since, critically, the simulator for a NIZK is permitted to choose the common string and may in particular choose it to be $G(s)$ for some random $s$. Then for a random $\rho$ it uses $s$ as the witness for $(x, G(s)) \in L'$. The non-uniform probabilistic polynomial time indistinguishability of outputs of $G$ from truly random strings, and the witness indistinguishability of the zaps for $L'$, imply that the output of the simulator is indistinguishable from a real transcript. $\diamondsuit$

# 5 Zaps and Verifiable Pseudo-Random Bit Generators

In this section we characterize aaps in terms of a new cryptographic primitive: *verifiable pseudo-random sequence generator* (VPRG). A VPRG is a pseudo-random generator where the holder of the seed can generate "proofs" of consistency of some parts of the sequence without hurting the unpredictability of the remaining bits. In the standard model we will exhibit a construction of zaps from VPRGs (Protocol VZ below). As we will see, the construction works even if the VPRG is *approximate*, in that the "proofs" of the bit values are occasionally incorrectly accepted, so it is possible to "cheat" a little. We will also show that if zaps exist then so do approximate VPRGs. Very roughly, *approximate* VPRGs can be designed to have multiple witnesses, so zaps, with their witness-indistinguishability, are sufficiently strong to yield the necessary proofs $\pi$ of consistency with some member in $S(VK)$. In contrast, we do not know how to design *strict* VPRGs to have multiple witnesses.

The following summarizes the relationships between zaps, VPRGs, and NIZKs, both in the standard model and in the common guaranteed random string model.

**Summary 5.1** *1. NIZKs exist in the common guaranteed random string model if and only if VPRGs exist in the common guaranteed random string model (Theorem 5.3).*

2. *NIZKs exist in the common guaranteed random string model if and only if zaps exist in the standard model (Theorem 4.3 and Corollary 4.4).*

3. *Zaps exists in the standard model if and only if approximate VPRGs (with certain parameters) exist in the standard model (Corollary 5.7 and Theorem 5.8).*

**Definition:** An $(s, k)$ *verifiable pseudo-random generator (VPRG)* is a pseudo-random sequence generator which, for security parameter $1^n$, maps a random seed $v$ of length $s(n)$ to an output sequence $a_1, \ldots, a_{k(n,|x|)}$ of length $k(n, |x|)$ and a verification key $VK$ where $s(n)$ and $k(n, |x|)$ and the length of $VK$ are fixed polynomials. The mapping should satisfy the following requirements:

**Binding:** The public verification key $VK$ binds the sequence: there is only one sequence $a_1, a_2, \ldots a_k$ consistent with $VK$ (there can be two different seeds $v$ and $v'$ that yield that same $VK$, but then they should also yield the same $a_1, a_2, \ldots a_k$).

**Verifiability:** For any subset $I \subseteq \{1, \ldots, k\}$ of indices, given the seed $v \in \{0,1\}^{s(n)}$ it is possible to construct a polynomial length proof $\pi$ of the consistency of the values of $\{a_i\}_{i \in I}$. Given $VK$ the verifier can check the proof $\pi$ of the consistency of $\{a_i\}_{i \in I}$ with $VK$. For any $VK$ and $I \subset \{1, \ldots, k\}$ there is (at most) one assignment to $\{a_i\}_{i \in I}$ for which there is a proof $\pi$ that the verifier accepts.

**Passing the $i$th Bit Test:** For all $1 \leq i \leq k$ and polynomial time adversaries $T$ the following should be true. Suppose that $T$ receives for a random $v \in \{0,1\}^{s(n)}$ the verification key $VK$ and

$$a_1, a_2, \ldots a_{i-1}, a_{i+1}, \ldots a_k.$$

The adversary $T$ selects $I \subset \{1, \ldots, k\}$ such that $i \notin I$ and receives a random proof $\pi$ for the consistency of $\{a_j\}_{j \in I}$. It then attempts to guess $a_i$. The probability, over the choice of the seed, the random choices in the construction of the proof $\pi$, and the random choices by $T$ that $T$ guesses $a_i$ correctly is negligibly close to $1/2$.

**Remark 5.2** *: Consider a* subset *test, i.e. instead of a single $1 \leq i \leq k$ there is a missing subset of indices and the distinguisher gets the values of $a_{i'}$ at all other locations plus candidate values for the missing locations. It can then ask to see a proof of consistency for any subset $I$ not including any of the missing indices and then has to guess if the candidate values are correct or just random. This test is equivalent to the $i$th bit test, just as the distinguishing test and the next bit test are equivalent for regular pseudo-random generators. Note that in the case of verifiable pseudo-random* functions *(VPRF) such an equivalence is not clear. The relation between VPRGs and VPRFs is further discussed in Sections 5.2 and 9.*

We also use a relaxation of VPRGs, which we call $d(n)$*-approximate* VPRGs. The differences are

**Relaxed Binding:** For any $VK$, there are at most $d(n)$ values for the revealed string that are accepted as consistent with $VK$. More precisely, there exists a set of strings $S(VK)$ of size at most $d(n)$ such that for any set $I$ of indices, if the prover can convince the verifier to accept the values $\{a_i\}_{i \in I}$, then there exists a string in $S(VK)$ consistent with these values.

**Two-Round Communication:** The proof of consistency may be "zap-like", that is, the choice of $VK$ and the proof may depend on a first-round message chosen by the verifier.

**Theorem 5.3** *VPRGs in the* shared random string *model exist if and only if NIZKs exist in the shared random string model. Moreover, in the* shared random string *model NIZKs imply VPRGs of arbitrary expansion.*

*Proof.*(Sketch) To construct VPRGs from NIZKs in the shared random string model, commit (say, using the protocol of [34], taking the first several bits of the common random string to be the "first-round" message of the receiver) to the seed of a pseudo-random sequence and use a NIZK to prove that the revealed value is the correct one. For the converse, given a VPRG in the common random string model, construct essentially the NIZK of Feige, Lapidot, and Shamir [20], in which the bits of what they call the intermediate random string (IRS) (see more about them below) are the bits of the VPRG. $\diamondsuit$

## 5.1 Zaps based on VPRG

**Proofs Based on Hidden Random Strings:** We find the following "physical" intuition helpful for describing certain types of proofs of membership. The prover is dealt a sequence of $\ell$ binary cards, where each card has value 1 with probability 1/2. The prover knows the values of the cards and can choose any subset to reveal to the verifier. The verifier learns absolutely nothing about the values of cards that are not explicitly revealed. The prover has no control over the values of the cards. The sequence of cards is a *hidden random string* (HRS).

To prove that $x \in L$, the prover, holding witness $w \in w(x)$, can choose any subset of the hidden bits to reveal to the verifier (cards to turn over). Let $\alpha$ be the locations and values of the revealed bits in the HRS. In addition to $\alpha$, the prover may send extra information, $\beta$, to the verifier. The verifier decides whether to accept or reject $x$ as a function of $\alpha$, $\beta$, and $x$.

The *soundness* requirement is that for some $q < 1$ such that $1 - q$ is non-negligible (that is, $q$ is non-negligibly far from 1), the probability (over the values of the hidden random bits) that the prover can cause the verifier to accept an $x \notin L$ is at most $q$, even if the prover is arbitrarily powerful. That is, with non-negligible probability $1 - q$ there is no triple $(x, \alpha, \beta)$ such that $x \notin L$ and the verifier accepts $(x, \alpha, \beta)$.

The *witness indistinguishability* requirement is that there exist a simulator that on input $x \in L$ (but without a witnesses to $x \in L$),

1. can create $(\alpha, \beta)$ identically distributed to the $(\alpha, \beta)$ pairs created in real executions of the proof;
2. given $\alpha$, $\beta$, and *any* witness $w'$ to $x \in L$, can generate an assignment to the remaining cards so that the distribution on *extended transcripts*, that is, the hidden cards, the revealed cards $\alpha$, and $\beta$, is *identical* to the distribution on extended transcripts in real executions by a prover holding witness $w'$. We call this "completing the simulation with $w'$."

Again: $\alpha$ and $\beta$ are chosen without access to a witness; then, given any witness $w' \in w(x)$, the simulator can create an assignment to all the cards, hidden and exposed, so that the distribution on triples containing $\alpha$, $\beta$, and the values for all the cards is exactly the distribution on these values in real executions with witness $w'$.

The concept of an HRS-based proof is exemplified by the noninteractive zero-knowledge proof systems of Feige, Lapidot and Shamir [20] and of Kilian and Petrank [29]. The idea is to implement the hidden random string using the output of the VPRG and to the opening using the proof capabilities of the VPRG.

### Protocol VZ: A VPRG-based Zap

The choice of parameters for VPRG-based zaps differs slightly from the choice in the case of NIZK-based zaps. This is because in the case of the VPRG we have less freedom: $k = k(n)$ (the length of $\rho$) is tied to the VPRG.

**Choice of Parameters:** Assume we have an HRS-based proof that for string $x$ and security parameter $n$ polynomially related to $|x|$ uses $\ell(n)$ cards, and on any input $x$ errs with probability at most $q$. Let $s = s(n)$ be the length of a seed permitting the VPRG to output $k = k(n, |x|)$ bits. To achieve soundness guarantee $\delta$ (that is, a cheating prover should succeed with probability at most $\delta$), we require that $k(n, |x|)$ will sufficiently expand the input: it should satisfy

$$q^{k(n,|x|)/\ell(n)} < \frac{\delta}{2^{n+s(n)+\ell(n)}}.$$

**The Protocol:** Let $m = k/\ell$. The HRS proof will be repeated $m$ times. The verifier sends to the prover random bits $b_1, \dots, b_k$.

The prover chooses $\ell$ random bits $c_1, \dots, c_\ell$ and a random seed $v \in \{0,1\}^s$ for the VPRG. Let $VK$ and $a_1, a_2, \dots, a_k$ be the output of the VPRG on $v$. The $i$th bit of the HRS is defined to be

$$a_i \oplus b_i \oplus c_{i \bmod \ell+1}.$$

The prover sends to the verifier: $VK$, $c_1, \dots, c_\ell$, and $m$ HRS-based proofs that $x \in L$, where the $j$th proof uses the $j$th block of $\ell$ bits of the HRS. For all revealed cards $1 \le i \le k$ the prover provides $a_i$ and a proof $\pi$ for the consistency of the revealed values.

Let $(\alpha_j, \beta_j)$ be the values of the revealed cards and additional information in the $j$th copy of the HRS-based proof, for $j = 1, \dots, m$. For the revealed cards the verifier, using $VK$, checks that the value revealed is the correct one. If not, the verifier rejects; otherwise the verifier accepts iff for all $m$ instances of the HRS-based proof, the HRS-based verifier accepts.

**Lemma 5.4** *Protocol VZ is witness-indistinguishable.*

*Proof.* Let $w$ and $w'$ be two witnesses that $x \in L$. Assume there exists a sequence $b_1^*, \dots, b_k^*$ and a distinguisher $T$ that, given $(x, w, w')$ and a transcript consisting of $b_1^*, \dots, b_k^*$ followed by the responses of the $m$ HRS-based proofs of $x \in L$, succeeds with non-negligible advantage $\varepsilon$ to guess which witness, $w$ or $w'$, was used by the prover in generating the response.

By a pigeonhole argument, for some $1 \le j \le m$ there exists a distinguisher for the following two types of transcripts, that distinguishes between them with advantage at least $\varepsilon/m$:

1. The prover uses witness $w$ for the first $j-1$ copies of the HRS-based proof and $w'$ for copies $j \dots m$.

2. The prover uses witness $w$ for the first $j$ copies of the HRS-based proof and $w'$ for copies $j+1 \dots m$.

Let us fix such a $j$ for the remainder of the proof.

We first use the simulator, whose existence is guaranteed by the definition of an HRS-based proof, to choose $\alpha$ and $\beta$ for the $j$th copy of the HRS-based proof. For a given seed $v$ to the VPRG, for the positions $i$ indicated by $\alpha$, we choose $c_i$ so that the value $a_i \oplus b_{(j-1)\ell+(i \bmod \ell+1)} \oplus c_{i \bmod \ell+1}$ opened is the value indicated by $\alpha$.

By the definition of witness-indistinguishability for an HRS-based proof, the simulator, now given $w$ and $w'$, can efficiently find a completion (choices for the unopened values) $\gamma$ corresponding to the case in which the witness used is $w$, as well as a completion $\gamma'$, corresponding to the case in which the witness used is $w'$. Together with the seed $v$, the completions $\gamma$ and $\gamma'$ determine, respectively, the values of the bits $c_i$ for each position $i$ that is *not* indicated by $\alpha$ (the values for the positions indicated by $\alpha$ were fixed above and will remain unchanged throughout the rest of the proof). Let $r = |\gamma| = |\gamma'|$. For $0 \le d \le r$, we denote by $c(\gamma, \gamma')_{[d]}$ values for the $c$'s not indicated by $\alpha$ that agree with $\gamma$ in positions $1, \dots, d$ and agree with $\gamma'$ in positions $d+1, \ldots, r$. Thus, when $d = 0$ the values all agree with $\gamma'$, while when $d = r$ the values all agree with $\gamma$.

We will now form a hybrid chain on proof strings. In every element in the chain, the seed $v$ remains unchanged, as do the $b^*$'s and the values for the $c$'s in the positions indicated by $\alpha$. Only the $c$'s not indicated by $\alpha$ will change as we move from one element in the chain to the next. The first element in the chain has values $c(\gamma, \gamma')_{[r]}$ for the $c$'s not indicated by $\alpha$. Thus, these values all agree with $\gamma$, where the witness is $w$. Having fixed all the $c$'s for this element of the chain, we can complete the description of the first element of the chain. The first $j-1$ blocks are HRS-based proofs constructed with witness $w$, and blocks $j+1$ through $m$ are constructed with witness $w'$. Moreover, by choice of the $c$'s, the $j$th block has been completed with $w$.

The next element in the chain has values $c(\gamma, \gamma')_{[r-1]}$ for the $c$'s not indicated by $\alpha$. Everything else remains the same: the values for the remaining $c$'s that were fixed in the description of the first element in the chain are again used here. Then, having again fixed all the $c$'s, the first $j-1$ blocks are HRS-based proofs constructed with witness $w$, and blocks $j+1$ through $m$ are constructed with witness $w'$. Note that the $j$th block might not really be something that could have been generated by the prover, since it is not completely consistent with a proof constructed using either $w$ or $w'$.

In general, for $0 \leq d \leq r$, the $d+1$th element in the chain has values $c(\gamma, \gamma')_{[r-d]}$ for the $c$'s not indicated by $\alpha$, for $0 \leq d \leq r$. The last element in the chain has values $c(\gamma, \gamma')_{[0]}$, that is, it agrees completely with $\gamma'$.

We note that the chain is non-empty, since otherwise the behavior of the prover on witnesses $w$ and $w'$ is identical and therefore yields no possibility of distinguishing between the two witnesses. Thus, the number of steps in the $\gamma \to \gamma'$ hybrid chain is $1 \leq r \leq \ell$ (including the endpoints, the chain has $r+1$ elements). We assumed an $\varepsilon/m$ advantage in distinguishing the two endpoints of the chain, hence there is an $i \leq \ell+1$ where the adversary has advantage at least a $\varepsilon/(m\ell)$ to distinguish between the $i-1$ and $i$th elements in the chain. The pseudo-randomness of the VPRG can be broken at this location. The subset $I$ is the one determined by $\alpha$ and the HRS proofs used in the other $m-1$ blocks. $\diamondsuit$

**Lemma 5.5** *Protocol VZ is sound; moreover, the first round can be fixed non-uniformly.*

*Proof.* Let $x \notin L$, $c_1, \ldots, c_\ell$, and the VPRG verification key $VK$ be fixed. We will show that with overwhelming probability, over the choice of $b_1, \ldots, b_k$, the prover will fail to convince the verifier to accept $x$. The key point is that once everything but the $b$'s has been fixed, the hidden random string is truly random – because $b_1, \ldots, b_k$ have not been chosen yet. Therefore each copy of the HRS-based proof has probability at most $q$ of failing to cause rejection. Since each proof is independent (because the $b_i$'s used in each copy of the HRS-based proof are independent), the overall probability that all $m = k/\ell$ copies fail is at most $q^m$.

The number of possible assignments to the $c$'s, $VK$'s, and $x \notin L$ is at most $2^{\ell+|s|+|x|}$. Hence, as long as

$$2^{\ell+|s|+|x|} q^m \leq \delta$$

for a *random* $b_1, \ldots, b_k$, the probability that there even exists a "bad" choice of $c_1, \ldots, c_\ell$, $VK$, and $x$ that erroneously causes the verifier to accept, is at most $\delta$. Thus, not only is the protocol sound, but the first message (the $b$'s) can be fixed non-uniformly. $\diamondsuit$

**Theorem 5.6** *Given an HRS proof system for a language $L$ using $\ell$ cards and with probability of error at most $q$ and given a VPRG mapping a seed $s$ to $k$ bits, if*

$$q^{k/\ell} < \frac{\delta}{2^{|x|+|s|+\ell}}$$

*then protocol VZ is a zap for $L$.*

Note that if instead of a VPRG we use a $d(n)$-approximate VPRG, then we can obtain a similar result by adjusting the counting argument to accomodate the $d(n)$ possible openings consistent with $VK$:

**Corollary 5.7** *Given an HRS proof system for $L$ using $\ell$ cards and with probability of error at most $q$ and given a $d(n)$-approximate VPRG mapping a seed $s$ to $k$ bits, if*

$$q^{k/\ell} < \frac{1}{d(|x|)} \frac{\delta}{2^{|x|+|s|+\ell}}$$

*then protocol VZ is a zap for $L$.*

As we show next, the converse holds as well and we can use zaps in order to obtain *approximate* VPRGs.

**Theorem 5.8** *Let $\ell(n)$ be any polynomial. Fix $m \geq 2$. Let $G$ be any pseudo-random generator taking a seed of length $s(n)$ and producing an output of length $\ell(n)$. Then, assuming every language $L \in NP$ has a zap, one can construct a $d(n)$-approximate VPRG expanding a seed of length $m \cdot s(n)$ to a string of length $m \cdot \ell(n)$, where $d(n) = m2^{\ell(n)}$.*

Note that the expansion is arbitrary, since $\ell(n)$ is an arbitrary polynomial and pseudo-random generators exist for any polynomial expansion, based on any one-way function.

*Proof.* We use the commitment scheme of [34] (in this scheme, the receiver sends an initial message, which can be fixed non-uniformly). The prover commits to $m$ seeds of length $s(n)$; $VK$ is the concatenation of the $m$ commitments. Using the pseudo-random generator, each seed yields a block of length $\ell(n)$, for a total output length of $m \cdot \ell(n)$. For any set $I$ of indices, the prover can reveal the values of the pseudo-random bits $\{a_i\}_{i \in I}$, and can prove using a zap that the revealed bits in at least $m-1$ of the blocks are consistent with $VK$ (this is certainly in NP, so it has a zap by assumption).

Verifiability is immediate from the zap. Relaxed binding is also simple, since given $VK$, the number of possible strings the prover can convince the verifier to accept is $m2^{\ell(n)} = d(n)$ (the prover has freedom to choose one of $m$ blocks on which he can cheat and which of $2^{\ell(n)}$ values to plug in there).

It remains to show passing of the $i$th bit test. Suppose the construction fails this test with some bias $\delta$. We will use the block $B$ containing $i$, to distinguish pairs of the form $(C(v), \tau)$ from $(C(v), G(v))$, where $C(v)$ is a commitment to a seed $v$ of length $s(n)$ and $\tau$ is random of length $k(n)$. Given a pair $(C(v), \mu)$, construct a key $VK$ as follows. Choose $m-1$ seeds $v_1, \ldots, v_{m-1}$, and arrange commitments to these seeds and the commitment $C(v)$ so that $C(v)$ is the commitment to the supposed seed for block $B$. Open the values for all positions other than $i$, and provide a zap of approximate consistency with $VK$, using the chosen seeds $v_1, \ldots, v_{m-1}$ as the witnesses to the fact that the revealed bits in at least $m-1$ of the blocks are consistent with $VK$.

If $\mu$ is pseudo-random with seed $v$, then by the witness-indistinguishability of the zap, the advantage in guessing the $i$th bit is close to $\delta$ (the witness-indistinguishability may introduce a negligible error, so we don't get exact advantage $\delta$). On the other hand, if $\mu$ is truly random, then there can be no bias. Therefore we have a distinguisher for $(C(v), \tau)$ from $(C(v), G(v))$. $\diamondsuit$

**Remark 5.9** *In the case of ordinary pseudo-random generators, it is known that the ability to expand by even one bit can be used to obtain arbitrary expansion. Is the same true of (approximate) verifiable pseudo-random generators? From Corollary 4.4, Theorem 5.3, and Corollary 5.7 we have only a higher threshold: if any* polynomial *expansion is possible (from $n$ to $n^{1+\varepsilon}$ for fixed $\varepsilon$), then we can build zaps and hence arbitrary expansion. See more open problems in Section 9.*

### 5.2 Construction of VPRGs

A VPRG can trivially be constructed from any *verifiable pseudo-random function* (see, e.g., [33]), since if the domain of the VPRF is small one obtains a VPRG. However, such a construction is "overkill;" moreover, the only known constructions we have of VPRFs require the Strong-RSA assumption.

We provide an alternate construction, following along the lines of the trapdoor-based synthesizer construction of Naor and Reingold [36]. To obtain (non approximate) VPRGs we require that the trapdoor permutation be certified (see [3]).

We assume the existence of a family $\mathcal{F}_n$ of certified trapdoor permutations with common domain $\mathcal{D}_n$, together with a hard-core predicate ($n$ is a security parameter). The VPRG output is given as a binary matrix (say, in row-major order). The matrix has $r$ rows and $c$ columns, where $rc = k$. Choose $r$ functions $f_1, \ldots f_r$, from $\mathcal{F}$ (one for each row) and $c$ random $y$'s (one for each column) in the intersection of the ranges of all the trapdoor permutations $\mathcal{D}_n$. The $(i, j)$ entry of the matrix will be the hard-core predicate of $f_i^{-1}(y_j)$.

Let $VK = f_1, \ldots, f_r, y_1, \ldots, y_c$. To prove the value of the $(i, j)$ entry, reveal $f_i^{-1}(y_j)$. Verification is immediate using $VK$ and the fact that each $f_i$ is a permutation that is easy to compute in the forward direction.

The length of the seed $s$ is $r \log |\mathcal{F}_n| + c \log |\mathcal{D}_n|$. As $n$ is fixed and $k$ grows, the expansion is roughly quadratic. This completes the description of our VPRG construction. The proof that it satisfies the $i$th bit test closely follows the proof in [36].

The standard example of a certifiable trapdoor function is RSA with prime public exponent $e$ satisfying $e > N$. If we relax the *perfect* binding requirement and instead aim for an approximate VPRG, then we can use any trapdoor permutation. For instance, we can use RSA with small exponent: associate with the $j$th column the $j$th smallest prime. The $(i, j)$th output bit is the hard-core bit of $y_j^{1/p_j} \bmod N_i$. The possible problem is that $p_j$ may divide $\phi(N_i)$. In this case $y_j$ may have multiple $p_j$th roots, possibly with different hard-core bits, and the owner of the generator can "cheat." However, there can be at most $|N_i|/\log|N_i|$ such primes. We can take this into account in setting the parameters. This construction is very related to Shamir's pseudo-random generator [40].

## 6 Oblivious Transfer in the Standard Model

Although there are many protocols under various assumptions for oblivious transfer, to date no 3-round protocol has been shown secure, without resorting to a random oracle model. We provide a protocol for 1-out-of-2 OT for which we are able to prove that the chooser's privacy is protected by the quadratic residuousity assumption (QRA) [25], and the sender's privacy is protected statistically (that is, with overwhelming probability over choices made by the sender, at most one value is transmitted to the chooser)[3]. The protocol is not known to ensure correctness, that is, the sender may choose what to send as a function of the chooser's message.

For simplicity, we describe the protocol for the case in which the sender's two inputs are bits $b_0, b_1$ The first round of the protocol, described next, can be eliminated if the Sender has a public key. In this case, the public key is chosen to be a random first-round message $\rho$ for zaps.

1. If the Sender has no public key, then it chooses a first-round message $\rho$ for a zap and sends it to the Chooser. (If the Sender instead has a public key, then this round is not needed.)

2. Let $i \in \{0, 1\}$ be the Chooser's input. The Chooser chooses a random 2-prime modulus $N$ and two random strings $y_0, y_1$ in $\mathbb{Z}_N^*$ such that $y_{1-i}$ is a quadratic residue modulo $N$ and $y_i$ is a non-residue with Jacobi symbol 1. Using $\rho$, the Chooser gives a zap $\pi$ of the statement: "$y_0$ is a QR $\bmod N$ OR $y_1$ is a QR $\bmod N$."

[3] Previous applications of QRA to OT appear, for example, in [8, 31].

3. The Sender verifies the zap $(\rho, \pi)$ and, if verification fails, the Sender aborts. If verification succeeds, the Sender chooses $x_0, x_1 \in_R \mathbb{Z}_N^*$ and sends the following two values to the Chooser in any order: $\{y_{b_0}^{b_0} x_0^2 \bmod N, y_{b_1}^{b_1} x_1^2 \bmod N\}$.

We now give a proof sketch of correctness of the protocol. Assume first that both parties are following the protocol correctly. Let $y_i$ be the unique quadratic non-residue modulo $N$ among $y_0, y_1$. Then $y_i^{b_i} x_i^2$ is a quadratic residue modulo $N$ if and only if $b_i = 0$. On the other hand, since $y_{1-i}$ is a quadratic residue modulo $N$, so is $y_{1-i}^{b_{1-i}} x_{1-i}^2$, independent of the value of $b_{1-i}$. Thus, the ability of the Chooser to compute quadratic residuousity yields only and exactly the value of $b_i$.

Now assume the Sender follows the protocol correctly but the Chooser does not. The soundness of the zap ensures that at least one of $y_0, y_1$ is a quadratic residue modulo $N$. Assume then that $y_j$ is a quadratic residue modulo $N$. Then $y_j^{b_j} x_j^2 \bmod N$ is always a quadratic residue, independent of $b_j$, and independent of how $N$ is chosen. Thus, the Chooser can learn at most one of $b_0, b_1$. Finally, by the QRA and the way in which a good Chooser constructs $N, y_0, y_1$, the sender cannot distinguish which of $y_0, y_1$ is the quadratic residue. In particular, the (polynomial time bounded) sender cannot distinguish among the following four distributions $(N, y_0, y_1, (\rho, \pi))$ where $\rho$ is fixed in Step 1, $N$ is chosen according to the protocol, and the other elements are chosen as follows:

1. $y_0$ is a random quadratic residue modulo $N$, $y_1$ is a random non-residue with Jacobi symbol 1, and $y_0$ is the witness used in constructing $\pi$;
2. $y_0$ and $y_1$ are both quadratic residues modulo $N$ and $y_0$ is the witness used in constructing $\pi$;
3. $y_0$ and $y_1$ are both quadratic residues modulo $N$ and $y_1$ is the witness used in constructing $\pi$;
4. $y_1$ is a random quadratic residue modulo $N$, $y_0$ is a random non-residue with Jacobi symbol 1, and $y_1$ is the witness used in constructing $\pi$;

Distributions 2 and 3 are indistinguishable by the witness-indistinguishability of the zap. Distributions 1 and 2 (and, similarly, distributions 3 and 4) are indistinguishable by the QRA. Thus, distributions 1 and 4 are computationally indistinguishable, so the Sender does not learn which of $b_0, b_1$ has been transferred to the Chooser.

**Remark 6.1** *Naor and Pinkas [35] were able to modify this approach to produce a different protocol with similar security properties; their protocol is based on DDH and does not explicitly use zaps.*

# 7 Timing-Based Applications

In this section we describe two delay-free timing-based (see Section 2.7) applications for zaps:

- 3-round concurrent zero-knowledge proofs of knowledge for any language $L \in \mathrm{NP}$
- 2-round deniable authentication

## 7.1 3-round Concurrent Zero-Knowledge Proofs of Knowledge

At a high level, the protocol consists of two steps. Let $x \in L$ be the statement to be proved. (1) The verifier chooses a statement $S$ and proves, using a zap, that $S$ is true; (2) the prover gives a proof of knowledge of a witness to the statement "$x \in L \vee S$". Intuitively, soundness comes from the fact that the verifier's proof does not reveal a witness to $S$. This is achieved by constructing $S$ to be the logical-or of two independent statements – in such a case witness-indistinguishability is known to imply witness-hiding [19]. A single pre-processing step is needed for both the proof of knowledge and to provide the first-round $\rho$ for the verifier's zap of $S$.

In a little more detail, the statement $S$ is a claim that of two given timed commitments to two random strings, at least one is *valid* – forced recovery of the committed value is possible (see the discussion in Section 2.7). Verifiable recovery implies the existence of a knowledge extractor. The extractor is used in constructing the simulator for proving zero-knowledge.

Let $f$ be a one-way function. Let $f^{(k)}(s)$ be the $k$th iterate of $f$ applied to $s$. Associated with any randomly chosen $s$, there is a $k$-bit pseudo-random string $B$ consisting of the hard-core bits of

$$s, f(s), f^{(2)}(s), \ldots, f^{(k-1)}(s),$$

respectively [6]. The basic technique for proving knowledge of a witness $w \in w(x)$ is to commit to $B^0$ and $B^1$ by giving a pair $f^{(k)}(s^0), f^{(k)}(s^1)$. The verifier then chooses one of the two blocks, say, $B^i$, to be revealed. The prover releases $s^i$ and gives $w \oplus B^{1-i}$, together with a proof of consistency. Because this only gives a probability 1/2 of detecting cheating, the process is repeated many times in parallel. The pre-processing step (Step 1 in the protocol) is just the transmission of sufficiently many pairs of the form $f^{(k)}(s^0), f^{(k)}(s^1)$, together with a $\rho$ for the verifier's zap in Step 2.

**3-round Timed Concurrent ZK POK for $L \in$ NP.** Common input $x \in L$, input to prover $w \in w(x)$.

1. (a) Let $f$ be a fixed one-way permutation ($f$ is part of the protocol, known to both parties). The prover sends to the verifier $2p$ pairs $(f^{(k)}(s_1^0), f^{(k)}(s_1^1), \ldots, (f^{(k)}(s_{2p}^0), f^{(k)}(s_{2p}^1)))$ for randomly chosen $s_i^j$, $i = 1, \ldots, 2p$ and $j = 0, 1$.

   (b) The prover also sends to the verifier $\rho$, a round-one message for a zap.

2. (a) The verifier selects a random $2p$-bit string $c_1 \ldots c_{2p}$.

   (b) The verifier chooses two random values $y_0$ and $y_1$ of length $p$, and constructs from them two commitment strings $\zeta_0$ and $\zeta_1$ using the timed commitment protocol. Using $\rho$, the verifier sends $\pi$ proving that at least one of the $\zeta_i$ is valid ($(\rho, \pi)$ constitute a zap).

   (c) The verifier sends to the prover a new round-one message $\rho'$.

3. (a) For each $1 \le i \le p$, the prover sends to the verifier $s_i^{c_i}$. For each such $i$ the prover also computes $B_i$, the pseudo-random $k$-bit string consisting of the hard-core bits of

   $$s_i^{1-c_i}, f(s_i^{1-c_i}), f^{(2)}(s_i^{1-c_i}), \ldots, f^{(k-1)}(s_i^{1-c_i}).$$

   (b) The prover checks the zap $(\zeta_0, \zeta_1, \rho, \pi)$. If the proof is invalid, the prover terminates the protocol.

   (c) The prover chooses $z$ at random.

   (d) Using $B_1, \ldots, B_p$ the prover commits to $z$ and $w$. Specifically, it sends $z \oplus B_1, \ldots, z \oplus B_p$; similarly it commits to $w$, using blocks $B_{p+1} \ldots B_{2p}$. We call the commitments to $z$ the *first group*, and the commitments to $w$ the *second group*. Using $\rho'$, the prover constructs a proof $\pi'$ that at least one of the following two statements holds: (1) there exists $z$ consistent with all of the commitments in the first group and $z$ is the value committed to in one of the timed commitments $\zeta_0$ or $\zeta_1$; or (2) there exists $w$ consistent with all of the commitments in the second group and $w \in w(x)$. The witness used for constructing the zap is the set of strings $\{s_{p+1}^{1-c_{p+1}}, \ldots, s_{2p}^{1-c_{2p}}\}$.

*Timing constraints*: $V$ accepts $P$'s Round 3 message only if arrives within time $\alpha$ on $V$'s local clock from the time at which $V$ sent its Round 2 message. $\alpha$ and $\beta$ (for the timing assumption) should be chosen to satisfy $\alpha \leq \beta$ and $2\beta + \gamma < T$, where the value $T$ is the time below which it is safe to assume that the timed commitment cannot be broken, even by a PRAM, and $\gamma$ is an upper bound on the time it takes to create a zap by a program that is given a witness. For completeness, $\alpha$ must be sufficiently large to permit the necessary computations by $P$, and the round-trip message delay.

The protocol is concurrent zero knowledge because it is *straight-line simulable* via the forced openings: every interaction can be simulated without rewinding the prover [17]. To see this, consider a single interaction. The simulator generates a real round-one message, which is given to the verifier. The verifier constructs its timed commitments and their proof $\pi$. The simulator checks $\pi$ and, if it is correct, continues with the protocol. The clocks are frozen and the simulator computes the forced opening of the timed commitments, obtaining $y$, the de-commitment of one of $\zeta_0$ and $\zeta_1$. The clocks are started again, the simulator sets $z = y$, commits to $z$ and a random string (instead of $w$), and constructs $\pi'$ using the commitment to $z$ as the witness. When the adversarial scheduler schedules $P$'s next message, the simulator sends $\pi'$.

Now consider four classes of transcripts: they differ according to the value committed to in the first block (random or $z = y$), the value committed to in the second block ($w$ or random), and which witness is used in creating the zap $\pi'$ ($w$ or $z$). Only 4 of the eight possibilities are relevant.

1. First block: random; Second block: $w$; witness is $w$.
2. First block: $z = y$; Second block: random; witness is $y$.
3. First block: $z = y$; Second block: $w$; witness is $w$.
4. First block: $z = y$; Second block: $w$; witness is $y$.

The real transcripts are the first class. The simulator outputs the second class. Classes 1 and 3 are computationally indistinguishable by the one-wayness of $f$ and the properties of hard-core bits. Classes 2 and 4 are indistinguishable for the same reason. Classes 3 and 4 are indistinguishable by the witness-indistinguishability of zaps. Hence, classes 1 and 2 are computationally indistinguishable.

We now argue that the interaction is sound and a proof of knowledge. If the prover completes the proof with probability $\delta$, then standard extraction techniques can be used to obtain a witness (strings $s_i^{1-c_i}$ for the appropriate set of indices $i$) with probability negligibly close to $\delta^2$.

Suppose $x \notin L$, and that a cheating prover succeeds with non-negligible probability $\delta$ to cause the verifier to accept. Then the timed commitment scheme can be broken with probability negligibly close to $\delta^2/2$, as follows. Consider a (possibly fictitious) non-faulty process running a perfect clock. By the $(\alpha, \beta)$ assumption, if $V$ is non-faulty and measures time at most $\alpha$ on its own clock between the time at which it sent its round 2 message and the time at which it received $P$'s round 3 reply, at most $\beta$ real time has elapsed.

Assume we are given a timed commitment $\xi_1 \in_R TC(y)$. Run the cheating prover for one step. Choose $c_1 \ldots c_p$ at random. Choose $y'$ and give $\xi_1 \in_R TC(y')$; then, using the witness based on $y'$, act as the verifier and in Step 2 give a zap that at least one of $\xi_1$ and $\xi_2$ is valid. By definition, such a zap can be constructed within time $\gamma$. If the prover responds (which it will do with probability at least $\delta$), repeat Steps 2 and 3, using the same timed commitments and zap in Step 2, but with a new random string $c'_1, \ldots, c'_p$. If the prover responds again, use the revealed $s_{c'_i}$ to obtain at least one of $y, y', w \in w(x)$. Since $x \notin L$, the value obtained is either $y$ or $y'$. By the witness-indistinguishability of the verifier's zap, the value will be $y$ with probability 1/2. The total time required for extraction is at most $2\beta + \gamma < T$ contradicting the assumption that breaking the timed commitment requires time at least $T$. Thus, the system is sound. That the system is a proof of knowledge is immediate from the extraction procedure described above.

**Theorem 7.1** *The protocol described above is a 3-round timed concurrent zero-knowledge proof of knowledge system for any language $L$ in NP.*

**Remark 7.2** *The straight-line simulability also permits the prover to use differing $(\alpha, \beta)$ pairs for the different verifiers.*

## 7.2 Timed 2-round Deniable Authentication

We now describe a 2-round timed concurrent deniable authentication protocol (see Section 2.5 for definition and discussion), based on zaps and timed commitments.

The $A\mathcal{P}$ has a public key $E_1, E_2, \rho$, where $E_1$ and $E_2$ are public encryption keys chosen according to a public-key cryptosystem generator that is non-malleable against chosen-ciphertext attacks in the post-processing mode, and $\rho$ is a first-round message for a zap.

1. The verifier chooses random strings $y_0, y_1, r$ and sends to the prover $c \in_R E_1(m \circ r)$ and timed commitments $\zeta_0 \in_R TC(y_0)$ and $\zeta_1 \in_R TC(y_1)$. In addition, using $\rho$, the verifier gives a zap that at least one of the $\zeta_i$ is valid. Finally, the verifier also sends to the prover a first-round message $\rho'$ for a zap.

2. The prover checks the zap $(\rho, \pi)$ and aborts if verification fails. Otherwise, the prover sends to the verifier $\eta \in_R E_1(r)$, $\delta \in_R E_2(s)$ for a randomly chosen $s$. Using $\rho'$, the prover sends a zap $\pi'$ that at least one of the following holds: $\eta \in E_1(r)$ or $s \in \{y_0, y_1\}$ (more specifically, $\pi'$ is a proof that $\eta$ is an encryption under $E_1$ of the suffix of the message encrypted by ciphertext $c$ OR $\delta$ is an encryption under $E_2$ of one of the values committed to by $\xi_1, \xi_2$). The witness used in creating $\pi'$ is the set of random bits in creating $\alpha$.

$V$ accepts if and only if both (1) the zap $(\rho', \pi')$ is accepted and (2) $P$'s response is received in a *timely* fashion, as specified in the timing constraints.

*Timing constraints*: $P$'s Round 2 message must arrive within time $\alpha$ on $V$'s local clock from the time at which $V$ sent its Round 1 message. $\alpha$ and $\beta$ are chosen to satisfy $\alpha \leq \beta$ and $\beta + \gamma < T$, where the value $T$ is the time below which it is safe to assume that the timed commitment cannot be broken, even by a PRAM, and $\gamma$ is an upper bound on the time it takes to create a zap by a program that is given a witness. For completeness, $\alpha$ must be sufficiently large to permit the necessary computations by $P$, and the round-trip message delay.

This completes the description of the deniable authentication protocol.

**Theorem 7.3** *The 2-round protocol is sound and deniable.*

*Proof.* We first argue soundness. Suppose that the adversary is trying to forge message $m$ and is given by the verifier $E(m \circ r)$. Then by the non-malleability of $E_1$ it cannot produce $E_1(r)$, even if it has access to a decrypting oracle for $E_1$ on all messages with prefix different than $m$[4]. Given that the adversary provides a zap at Step 2, it must be the case that $s = y_i$ for some $i \in \{0, 1\}$. Then together, the real prover, who knows $D_2$, and the adversary can be used to break the timed commitment scheme with probability $1/2$: given $TC(y)$, choose $y'$ at random and give $TC(y')$; then, using the witness based on $y'$, give a zap that at least one of $TC(y)$ or $TC(y')$ (in random order) is recoverable. By definition, such a zap can be constructed within time $\gamma$. If the forger gives back $s = y$ within time $\alpha$, then $TC$ has been broken in time at most $\beta + \gamma < T$.

We now argue deniability. The simulator extracts from $TC(y_0)$ and $TC(y_1)$ either $y_0$ or $y_1$. It creates $E_1(r')$ for a random $r'$ and creates $E_2(y_i)$ and uses it as a witness to a zap that $\eta \in E_1(r)$ or $s = y_i$. The proof of indistinguishability of simulated and real transcripts is analogous to the proof of Theorem 7.1. $\diamondsuit$

[4]Actually it seems that we do not need $E_1$ to resist any chosen ciphertext attacks and it is enough that it is non-malleable against chosen plaintext attacks. The reason is that we can give the adversary an encryption of a random string instead of $E_1(r)$ and use the forced opening of the timed commitment in order to obtain a zap in the second step.

# 8 Witness Protection in the Resettable Model

## 8.1 Resettable Witness-Indistinguishability

For a formal definition of resettable witness indistinguishability, see [9]. We will motivate the definition informally by focusing on smart cards. Intuitively, a smart card is loaded with $x$, $w \in w(x)$, and a seed $s$ for a pseudo-random function, at the time it is created. This seed is the only source randomness the card has; furthermore, we assume that the card is stateless, i.e. does not change its internal memory between sessions. Our interest is in protecting the prover from a verifier $V^*$ that runs the prover many times on the same $x, w, s$. Let us use the notation $(P(x, w, s), V^*(x, z))$ denote the transcript of exactly this kind of attack where $z$ is auxiliary information known to $V^*$ (in particular, we may even have $z = w, w'$). Letting $w, w' \in w(x)$, a proof that $x \in L$ is resettable witness-indistinguishable if for all probabilistic polynomial time $T, V^*$ and $z$:

$$|\text{Pr}_{(V^*,s)}[T(P(x, w, s)V^*(x, z))] - \text{Pr}_{(V^*,s')}[T(P(x, w', s')V^*(x, z))]| \quad \leq \quad \nu(n).$$

Every zap for a language $L \in NP$ yields a 2-round resettable witness-indistinguishable proof system for $L$ as follows. On input $\rho$, the prover computes $R = f_s(x, \rho)$, where $f_s$ is a pseudo-random function with seed $s$. It then uses the bits $R$ as the "random" bits in computing the zap response $\pi$.

Soundness holds because the round-one message $\rho$ is not needed for *unpredictability* – indeed, soundness holds even if some $\hat{\rho}$ is fixed non-uniformly and before $x$ is chosen. As for witness-indistinguishability, from the WI of the zap it follows that an assumed distinguisher for the resettable system can be used to distinguish the output of the pseudo-random function from truly random, a contradiction.

## 8.2 Resettable Zero-Knowledge

We first present our 3-round timing-based rZK protocol for any $L \in NP$, and then compare it to previous results.

Let $(E, D)$ be the encryption and decryption algorithms of a semantically secure encryption method. The scheme need not be public-key, but there should be a public description $pd$ of the encryption key with the following two properties. (1) It is easy to verify that decryption is unique, that is, given ciphertext $c$ and a public description $pd$ there should be at most one $p$ satisfying $c \in E(p)$. (2) Given $pd$ it is easy to verify that there exists decryption key $dk$ such that given $c \in E(p)$ we have $D_{dk}(c) = p$.

An example of such an encryption scheme can be based on RSA with large public exponent, as in Section 5.2. That is, the public key is $(e, N)$, in which the exponent $e$ is prime and sufficiently large (so that $e$ cannot possibly divide $\phi(N)$); $pd = (e, N)$ in this case and the actual encryptin is doen using the hardcore predicate of the exponentiation with $e$ function. Alternatively, $E$ could be a pseudo-random permutation cipher (with random padding to avoid collisions) and where $pd$ is a (perfectly binding) commitment to the seed.

For this application, we require that the timed commitment scheme be secure non-uniformly, i.e. that there does not exist a PRAM with fixed advice tape that can break the commitment scheme with non-negligible probability in time less than $T$.

**3-round Timing-Based rZK for $L \in NP$**

1. The prover chooses $pd$ (the public description of the encryption key of $E$) and a random string $\rho$ and sends both to the verifier.

2. The verifier checks that encryptions under $E$ are uniquely decryptable (as discussed above) and if not, rejects. Assuming $E$ passes the test, the verifier chooses random strings $y_0, y_1$ and sends to the prover timed commitments $\zeta_0 \in_R TC(y_0), \zeta_1 \in_R TC(y_1)$ and, using $\rho$, a zap $\pi$ that at least one of the two timed-commitments is valid. The verifier also sends a string $\rho'$ to the prover.

3. The prover checks $(\pi, \rho)$. If it is accepted, then the prover uses the random bits defined by an application of its pseudo-random function on the message sent by the verifier to generate $a \in_R E(w)$ and $b \in_R E(z)$ where $w \in w(x)$ and $z$ is random. Using $\rho'$ and part of the output of the pseudo-random function the prover also generates a zap $\pi'$ that $w \in w(x)$ OR $z \in \{y_0, y_1\}$. The witness used consists of the random bits used in generating $a$. $a, b$ and $\pi'$ are sent.

The verifier checks that $(\rho', \pi')$ is accepted, that $b$ has unique decryption and that the prover's response was timely, as defined by the timing constraints, accepting if and only if all conditions are satisfied.
*Timing Constraints*: $P$'s Round 2 message must arrive within time $\alpha$ on $V$'s local clock from the time at which $V$ sent its Round 1 message. $\alpha$ and $\beta$ (from the timing assumption) are chosen to satisfy $\alpha \leq \beta$ and $\beta + \gamma < T$, where the value $T$ is the time below which it is safe to assume that the timed commitment cannot be broken, even by a PRAM, and $\gamma$ is an upper bound on the time it takes to create a zap by a program that is given a witness. For completeness, $\alpha$ must be sufficiently large to permit the necessary computations by $P$, and the round-trip message delay.

Note that the only party that has to measure time is $V$, which is considered more resourceful than the prover (who may be a smart-card with no independent clock) in the resettable setting

**Theorem 8.1** *For any $L \in NP$ the above protocol is rZK.*

*Proof.* A straight-line simulator can be constructed in a similar fashion to the construction in the proof of Theorem 7.1. For soundness we use the *existence* of a decryption algorithm $D$ with decryption key $dk$. If the protocol is not sound, then this key can be used to break the timed commitment in exactly the same way as the proof of knowledge was used in the proof of Theorem 7.1, violating the assumed non-uniform security of the timed commitment. We therefore have a non-constructive reduction: given an algorithm for providing false proofs for $L$ we know there exists an algorithm for breaking the timed-commitment; however, the reduction does not yield an effective method for the conversion.

Note that a proof of security which does not yield an effective procedure to break the underlying assumptions is rare. $\diamondsuit$

# 9 Open Questions

One vein of open problems induced by this work is with respect to the new primitive VPRG: Can VPRGs be composed "á la GGM", as can ordinary pseudo-random generators? This is related to the issue of constructing VPRGs with better expansion as well as to the question whether there is a general construction of VPRFs from VPRGs. A different issue is whether VPRGs can be based on an assumption weaker than trapdoor permutations? For example, is it possible to base VPRGs on the Diffie-Hellman assumption (either computational or the decisional version)?

A second vein of questions deals with efficiency and practicality. We have used general NIZKs; thus any proof must go through a reduction to an NP-complete problem. It would be useful to have more efficient, special-purpose zaps, for instance, a zap that one of $x$ and $y$ is a quadratic residue modulo $N$. Another concrete question regarding zaps is to construct one in conjunction with a timed-commitment, so that it will be simple to prove consistency.

A third vein of questions deals with round-efficiency: in which cases are our protocols round-optimal? It is not hard to argue that 2-round (non-black-box) zero-knowledge proofs *of knowledge* are impossible, even using timing. However, we do not know whether or not 2-round (non-black-box) zero-knowledge proofs are possible, with or without timing.

## References


[1] B. Barak, *How to Go Beyond The Black-Box Simulation Barrier*, Proc. of the 42nd IEEE Symposium on the Foundation of Computer Science, 2001.

[2] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations among notions of security for public-key encryption schemes*, Advances in Cryptology - Crypto'98, LNCS vol. 1462, Springer, 1998, pp. 26–45.

[3] M. Bellare and M. Yung, *Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation*, Journal of Cryptology 9(3), 1996, pp. 149–166.

[4] M. Blum, A. De Santis, S. Micali, and G. Persiano. *Noninteractive zero-knowledge*, SIAM Journal on Computing 20(6), 1991, pp. 1084-1118.

[5] Blum M., P. Feldman and S. Micali, *Non-Interactive Zero-Knowledge Proof Systems*, Proc. 20th ACM Symposium on the Theory of Computing, Chicago, 1988, pp 103-112.

[6] M. Blum and S. Micali, *How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits*, SIAM J. Comput. 13(4), 1984, pp. 850–864.

[7] D. Boneh and M. Naor, *Timed Commitments*, Advances in Cryptology - CRYPTO'2000 Proceedings, Lecture Notes in Computer Science No. 1880, Springer, 2000, pp. 236–254.

[8] G. Brassard, C. Crepeau,. and J. M. Roberts, *All-or-nothing disclosure of secrets*, Advances in Cryptology - CRYPTO '86, Lecture Notes in Computer Science No. 263, Springer, 1987 pp. 234–238.

[9] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali, *Resettable Zero-Knowledge*, Proc. 32nd ACM Symp. Theory of Computing, 2000, pp. 235–244.

[10] R. Canetti, J. Kilian, E. Petrank and A. Rosen. *Concurrent Zero-Knowledge Requires* $\tilde{\Omega}(\log n)$ *Rounds*, Proc. of the 33rd ACM Symposium on the Theory of Computing, 2001, pp. 570–579. Full version: Electronic Colloquium on Computational Complexity, Report TR01-050, Volume 8, 2001. Available: `www.eccc.uni-trier.de/eccc/`

[11] I. Damgård, *Efficient Concurrent Zero-Knowledge in the Auxiliary String Model*, Advances in Cryptology - EUROCRYPT 2000, Lecture Notes in Computer Science No. 1807, Springer, 2000, pp. 418–430.

[12] D. Dolev, C. Dwork and M. Naor, *Non-malleable Cryptography*, Siam J. on Computing, vol. 30(2), 2000, pp. 391–437.

[13] , C. Dwork, *The Non-Malleability Lectures*, Course notes for CS 359, Stanford University, Spring 1999, available at: `theory.stanford.edu/~gdurf/cs359-s99`.

[14] C. Dwork and M. Naor, *Pricing via Processing -or- Combatting Junk Mail*, Advances in Cryptology – CRYPTO'92, Lecture Notes in Computer Science No. 740, Springer, 1993, pp. 139–147.

[15] C. Dwork and M. Naor, *Method for message authentication from non-malleable crypto systems*, US Patent No. 05539826, issued Aug. 29th 1996.

[16] C. Dwork, M. Naor, and A. Sahai, *Concurrent Zero-Knowledge*. Proc. of the 30th ACM Symposium on the Theory of Computing, 1998, pp. 409–418.

[17] C. Dwork and A. Sahai, *Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints*, Lecture Notes in Computer Science No. 1462, Springer, 1998, 442–457.

[18] S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts,* Communications of the ACM 28, 1985, pp. 637–647.

[19] U. Feige and A. Shamir, *Witness Indistinguishable and Witness Hiding Protocols* Proc. 22nd ACM Symposium on the Theory of Computing, 1990, pp. 416–426.

[20] U. Feige, D. Lapidot and A. Shamir, *Multiple NonInteractive Zero Knowledge Proofs Under General Assumptions*, SIAM J. Comput. 29(1), 1999, pp. 1–28.

[21] O. Goldreich, **Foundations of Cryptography - Basic Tools**, Cambridge U. Press, 2001.

[22] O. Goldreich S. Goldwasser and S. Micali, *How to Construct Random Functions*, J. of the ACM 33 (1986), pp. 792–807.

[23] O. Goldreich and H. Krawczyk. *On the Composition of Zero Knowledge Proof Systems*. SIAM J. on Computing, Vol. 25, No. 1, pp. 169–192, 1996.

[24] O. Goldreich, S. Micali and A. Wigderson, *Proofs that Yield Nothing But their Validity, and a Methodology of Cryptographic Protocol Design*, J. of the ACM 38, 1991, pp. 691–729.

[25] S. Goldwasser and S. Micali. *Probabilistic Encryption*, Journal of Computer and System Sciences, Vol. 28, April 1984, pp. 270–299.

[26] S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof Systems,* SIAM Journal on Computing, Vol. 18, 1 (1989), pp. 186-208.

[27] J. Håstad, R. Impagliazzo, L. A. Levin and M. Luby, *A Pseudorandom Generator from any One-way Function*, SIAM J. Comput. 28(4), 1999, pp. 1364–1396.

[28] R. Impagliazzo, M. Naor, O. Reingold, and A. Shamir, *personal communication*, 1998.

[29] J. Kilian and E. Petrank, *An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions*, Journal of Cryptology, Vol. 11(1), 1998, 1-27.

[30] J. Kilian, E. Petrank and C. Rackoff, *Lower Bounds for Zero Knowledge on the Internet*, Proceedings of 39th IEEE Symposium on Foundations of Computer Science, 1998, pp. 484–492.

[31] E. Kushilevitz and R. Ostrovsky, *Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval*, Proceedings of 38th IEEE Symposium on Foundations of Computer Science, 1997, pp. 364-373, 1997.

[32] M. Luby, **Pseudorandomness and Cryptographic Applications**, Princeton University Press, 1996.

[33] S. Micali, M. Rabin, and Salil Vadhan, *Verifiable Random Functions*, Proceedings of 40th IEEE Symposium on Foundations of Computer Science, 1999, pp. 120–130.

[34] M. Naor, *Bit Commitment Using Pseudo-Randomness*, Journal of Cryptology, vol 4, 1991, pp. 151–158.

[35] M. Naor and B. Pinkas, *Efficinet Oblivious Transfer Protocols*, Proc. of the twelth Annual ACM-SIAM Symposium on Discrete Algorithms, 2001, pp. 448–457.

[36] M. Naor and O. Reingold, Synthesizers and their application to the parallel construction of pseudo-random functions, *J. of Computer and Systems Sciences*, vol. 58 (2), April 1999, pp. 336-375.

[37] M. Naor and M. Yung, *Public-key Cryptosystems provably secure against chosen ciphertext attacks* Proc. 22nd Annual ACM Symposium on the Theory of Computing, Baltimore, 1990, pp. 427–437.

[38] M. O. Rabin, "How to exchange secrets by oblivious transfer", Tech. Memo TR-81, Aiken Computation Laboratory, 1981.

[39] A. Rosen, *A Note on the Round-Complexity of Concurrent Zero-Knowledge*, Advances in Cryptology - CRYPTO'2000 Proceedings, Lecture Notes in Computer Science Vol. 1880, Springer, 2000, pp. 451–468.

[40] A. Shamir, *On the Generation of Cryptographically Strong Pseudorandom Sequences*, ACM Trans. on Computer Systems 1(1), 1983, pp. 38–44.