2-Round Zero Knowledge and Proof Auditors*

Cynthia Dwork[†] Larry Stockmeyer[‡]

Abstract

We construct 2-round (*i.e.*, 2-message), public-coin, black-box (concurrent) zero-knowledge proof systems and arguments for any language in NP under the assumption that the prover is resource-bounded during the execution of the protocol.

1 Introduction

Determining the minimum amount of interaction needed for zero-knowledge proofs has been the subject of intensive study since the concept was introduced in the ground-breaking paper of Goldwasser, Micali, and Rackoff [24]. We show that two rounds are sufficient (but perhaps not necessary!) if the prover is *resource-bounded* during execution of the protocol: that is, there exist two-round black-box zero-knowledge proof systems for any language in NP under the assumption that the prover is resource-bounded during the execution of the protocol.

The concept of resource-boundedness during an execution is not new. For example, the assumption that a prover is polynomial-time bounded "during its lifetime" implies that it is so bounded during any single protocol execution. Specific polynomial time bounds are used explicitly in the constant-round *concurrent* zero-knowledge protocols in [12, 13]. We use them here, together with certain assumptions, to obtain 2-round black-box zero-knowledge proof systems and arguments (proof systems with polynomial-time bounded provers) for any language in NP. As we will discuss, the assumption of resource-boundedness is independent of whether or not the prover is deterministic.

In this work we also consider bounds on prover *advice*. These, too, are implicit in the literature. For example, in the Feige-Shamir 4-round zero-knowledge argument [15], in which the verifier chooses a trapdoor commitment scheme, it is implicitly assumed that the prover does not have access to a huge sorted table of commitment scheme descriptions, together with their corresponding trapdoors, that can be indexed into in polynomial time: otherwise, seeing the verifier's choice of scheme, the prover could look up the trapdoor information in the table.

Colloquially, imagine a setting in which the prover enters a special "proving chamber" from which access to the outside world is impossible and into which only a certain polynomial amount

---

*An extended abstract of this paper appears in the Proceedings of the 34th ACM Symposium on Theory of Computing, May 2002.

[†]Microsoft Research - Silicon Valley Campus, 1065 L'Avenida, Mountain View, CA 94043,
email: dwork@microsoft.com. Work performed while this author was at the Compaq Systems Research Center.

[‡]IBM Almaden Research Center, 650 Harry Rd., San Jose, CA 95120, USA, email: stock@almaden.ibm.com.

of information, or "advice," may be carried. More specifically, there is a parameter n where the good prover needs only $O(n)$ bits of advice, and no cheating prover with $O(n^{2-\varepsilon})$ bits of advice can convince the verifier to accept a false statement with non-negligible probability. Such a setting might be realized by means of a special smart-card reader. In this model there exist 2-round black-box zero-knowledge proof systems for any language in NP. Soundness in this setting does not rely on any unproven assumptions; the prover may have unbounded time and scratch space – only the advice needs to be bounded.

Our protocols start from the *zaps* of Dwork and Naor [11]. Informally, a zap is a witness-indistinguishable[1], two-round, public-coin, proof system, in which the first-round message (denoted ρ below) can be fixed once and for all. The new ingredient is what we call a function *proof auditor*. Very roughly, a proof auditor for a function f takes as input x, y, and a supposed "proof" z that $y = f(x)$, and either accepts or rejects. The requirement is: $\forall x, y$ there exists z causing the auditor to accept (x, y, z) if and only if $y = f(x)$.

To understand the role of the proof auditor, consider the following flawed approach to constructing a 2-round timed zero-knowledge proof system for a language L in NP. Let f be an "icky-poly" time function that on input x cannot be computed in time less than $q(|x|)$. Assume the existence of a probabilistic public-key cryptosystem scheme generator $\mathcal{G}$, in which an encryption of a string is an absolute commitment to the string. For E produced by $\mathcal{G}$, and for all y, let $E(y)$ denote the set of encryptions of y under E. Let the common inputs be the security parameter 1^k and a string t of size polynomial in k. The prover P is trying to prove to the verifier V that $t \in L$.

Protocol Flawed Prototype

1. $V \longrightarrow P : x \in_R \{0,1\}^k$, ρ (for the zap)
2. $P \longrightarrow V$:

   Choose: $E \in_R \mathcal{G}(1^k)$, $x^* \in_R \{0,1\}^k$, $\beta \in_R E(x^*)$;

   Send: E, zap("$t \in L \vee \beta \in E(f(x))$"), β

Timing Constraints: The verifier checks that the prover responds within time $T \ll q(k)$.

The *intuition* is that, since the prover cannot find $f(x)$ in time less than $q(|x|)$, it also cannot find a public encryption key E and an encryption β of $f(x)$ in that time; thus, the second clause of the zap must be false and hence, if the zap is accepted, it must be the case that $t \in L$.

There are two serious difficulties with this approach. First, since, using current techniques, the zap must go through Cook's theorem, the length of the zap (because of the second clause) must be at least $q(|x|)$, and hence cannot be even written down in time $\ll q(|x|)$. We simply ignore this problem for the current discussion: just pretend for now that the length of the zap can miraculously be made small. The second problem is that the intuition for soundness is flawed. We do not know how to show that finding $E \in \mathcal{G}(1^k)$ and $\beta \in E(f(x))$ requires as much time as does computing $f(x)$ from x. This (and the corresponding question for commitment) is a long-standing open problem

[1] A proof system of a statement having two witnesses is witness-indistinguishable if the verifier, even knowing both witnesses, cannot determine which witness is used in constructing the proof.

in cryptography. The concept of a proof auditor (for f) is inspired by this second problem, as we now explain.

Suppose that $t \notin L$ but some cheating prover is able to convince V to accept t with probability 1 over the coin flips of both machines. That is, for all Step 1 challenges x, the cheating prover always manages to quickly send the required Step 2 message of the Flawed Prototype, and the verifier always accepts the zap. Then, intuitively, the *computation* of the cheating prover, together with the decommitment information, is itself a "proof" of the statement $y = f(x)$, where y is obtained by decommitting β using the decommitment information. Moreover, under the assumption of the miraculously small-length zap, the proof is short, assuming also that the decryption key and decryption computation are both short as well (much less than $q(|x|)$).

Still speaking intuitively, suppose statements of the form "$y = f(x)$" have no short proofs. That is, suppose checking the statement "$y = f(x)$" is no easier than finding $f(x)$ given x. Then such perfect cheating is impossible since, as just argued, it would yield a short "proof" of "$y = f(x)$": feed x, ρ to the prover and let it respond with E, β, and the zap. The proof is the decommitment information (it needn't be easy to find, just short), the decommitment computation, and the computation of the prover. (Since we assume the prover always succeeds, the computational time of V is irrelevant: we know what it is going to say.) Since the issue is the difficulty of *checking* the value of the function, rather than of *computing* this value, nondeterminism of the prover is irrelevant.

The concept of a proof auditor for the function f formalizes the informal use of the term "proof" in the above discussion. We describe an interactive proof system that uses a function f (Protocol ZEBRA), and tie the existence of a cheating prover using certain specified resources (and succeeding with any given probability), to the existence of a proof auditor for f using essentially the same resources. That is, for NP language L, we show that if there exists a cheating prover and a $t \notin L$ such that the prover, using such-and-such resource bounds, erroneously convinces the verifier to accept t with probability so-and-so, then there is a proof auditor for the function f using (approximately) such-and-such resource bounds, and having correctness probability (approximately) so-and-so. We then either *assume* (in the case of time) or *prove* (in the case of advice) that no such proof auditor – and hence cheating prover – exists. This addresses the erroneous reasoning problem of the Flawed Prototype. The problem of the long zap (with the clause $\beta \in E(f(x))$) is addressed through a combination of prover preprocessing and (possibly limited) malleability of encryptions.

The connection between cheating provers and proof auditors (Theorem 6.1) is crucial to our results. To make the connection completely rigorous, we must carefully define the class of cheating provers (Section 5), as well as proof auditors for functions (Section 4). Our protocol requires the prover to do some preprocessing, including the computation of $f(x^*)$ for a randomly chosen x^*. Speaking intuitively, the preprocessing cannot be done with little advice (we prove) or very quickly (we believe). Thus, intuitively, the results of the supposed preprocessing cannot be obtained by a cheating prover *during the course of its interaction with the verifier* if the cheating prover is advice-bounded or time-bounded during the time of the actual interaction. The (rather joyless) definition of the classes of provers formally captures the preprocessing and interacting portions of the provers, as well as their potential nondeterminism. The (even less joyful) definition of a function proof auditor must be structured accordingly to maintain the connection between cheating provers and proof auditors. As a result, proof auditors are structured to have two components, the *advice* portion, tied to prover preprocessing, and the *auditing* portion, tied to the interactive part of the prover. We will refer to the auditing portion in our informal summary of results below.

Note that in the Flawed Prototype, arguing black-box zero-knowledge is straightforward, and hinges on the witness-indistinguishability of the zap and the semantic security of the commitment scheme. This will be the case in all our protocols. The interesting portions of our work will always be the proofs of soundness.

1.1 Informal Summary of Results

Despite not yet having given formal definitions of the new terms, we informally summarize our results in order to give the reader the flavor of our work. We list our results in decreasing order of the strength of assumptions used. We have one basic protocol, and each result is a specific instantiation of this basic protocol. The protocol is described in Section 3. Proof auditors are formally defined in Section 4. The connections between proof auditors and cheating provers is described in Section 6, and three results described here are derived as applications of this connection.

All our results use zaps, which can be constructed from NIZKs or from approximate verifiable bit generators [11]. All known constructions of these primitives require trapdoor permutations for efficient provers for completeness, but not for soundness. (For powerful provers, one-way permutations suffice.) The lengths of the zaps affect our choice of parameters. The shortest zaps we know of are constructed via the NIZKs of Kilian and Petrank [26].

A *completely malleable* cryptosystem has the property that, given only the encryptions of two bits (and the public key, if it is a public-key system), it is easy to compute encryptions of both the AND and the XOR of the corresponding plaintext bits (with no increase in ciphertext length).

(1) Assume (for soundness) that there exists a function f with no *fast* proof auditor, and assume the existence of a completely malleable semantically secure cryptosystem. Then, assuming the existence of zaps, NP $\subseteq$ TZKIP(2), where TZKIP(2) denotes the set of languages with 2-round timed zero-knowledge proofs. The assumption of a completely malleable cryptosystem is enormous and quite possibly unjustified. Certainly the problem has been open for many years [31].

Results (2) and (3) require an XOR-malleable (for completeness) semantically secure (for zero knowledge) public-key cryptosystem (given encryptions of two bits it is easy to compute an encryption of their XOR *without access to the private key*). The current candidates are the Goldwasser-Micali [23] and the Goldreich-Goldwasser-Halevi modification of the Ajtai-Dwork scheme [18, 1]. The former requires the quadratic residuousity assumption, the latter requires hardness of the worst-case unique shortest vector problem. The punishing size of the Ajtai-Dwork public key has guided our choice. Again, the specific cryptosystem (lengths of keys and ciphertexts as a function of security parameter) affects our parameters.

(2) Assume (for soundness) there exists a function f computable with a circuit consisting only of XOR gates, and having no fast auditor. Then, assuming zaps and an XOR-malleable public-key cryptosystem, NP $\subseteq$ TZKIP(2).

(3) This result involves a class $\mathcal{LIN}$ of linear functions (defined in Section 7.2). We prove the following fact: Almost all functions $f \in \mathcal{LIN}$ have no *small-advice* (for now, think "small circuit") auditor. Assuming zaps and an XOR-malleable public-key cryptosystem, NP $\subseteq$ AZKIP(2), where AZKIP(2) denotes the class of languages enjoying 2-round advice-bounded zero-knowledge interactive proof systems. The assumptions are used only for completeness and zero-knowledge. Soundness is unconditional.

In all cases, soundness holds even against a *nondeterministic* cheating prover (see Section 5). As noted above, proving black-box zero-knowledge is straightforward, and the same proof works in all cases (Section 8). It is also straightforward that our protocol achieves concurrent zero-knowledge, although the desired degree of concurrency affects the choice of parameters. We will make no further mention of concurrent zero-knowledge until the end of Section 8.

1.2 Relation to Previous Work: Counting Rounds

Recall that, informally, an interactive proof system is zero-knowledge if and only if there exists a probabilistic polynomial-time simulator that can generate conversations that are polynomial-time indistinguishable from real conversations (see Section 2 for a precise definition).

Goldreich and Oren defined *auxiliary-input* zero-knowledge, the variant of zero-knowledge now used in virtually all cryptographic applications, and showed that no language outside of BPP has a 2-round auxiliary-input zero-knowledge protocol [21].

Goldreich and Krawczyk defined the so-called *black-box* variant of zero-knowledge (BBZK), in which the simulator requires only oracle access to the verifier, and showed that no language outside of BPP has a 3-round *private-coin* BBZK proof or an $O(1)$-round *public-coin* BBZK proof. The lower bounds apply to arguments as well [20]. Feige and Shamir showed the existence of 4-round BBZK arguments for any language in NP [15]. Goldreich and Kahan obtained 6-round BBZK proofs [19].

The study of concurrent zero-knowledge – in which a single prover may be concurrently engaged in multiple asynchronous interactions – was initiated by Dwork, Naor, and Sahai, who showed that, using a mild timing assumption, it is possible to achieve black-box concurrent zero-knowledge in 5 rounds [12]. Dwork and Sahai showed how to move all use of time to a preprocessing phase, engaged in by each prover and verifier pair asynchronously, whenever the two parties should choose for the first time to engage in the interactive proof protocol [13].

Kilian, Petrank, and Rackoff showed a 5-round lower bound for private-coin black-box *concurrent* zero-knowledge for languages outside of BPP [27]; this was strengthened to 8 rounds by Rosen [32] and to $O(\sqrt{\log n})$ by Canetti, Kilian, Petrank, and Rosen [8]. The best known upper bound, due to Kilian, Petrank, and Richardson [28] (extending techniques introduced by Richardson and Kilian [30]), is $\omega(\log^2 n)$, meaning that the running time may be bounded by $h(n)\log^2 n$ for any function $h : \mathbb{N} \to \mathbb{N}$ that cannot be bounded by any constant.

Dwork and Naor introduced the notion of *moderately hard* (roughly: tractable but very expensive) functions, and proposed a method for combatting junk e-mail and junk faxes by means of these functions [10]. Moderately hard functions can be used to implement the above-mentioned timing assumptions; they are used in this work in the design of the 2-round resource-bounded zero-knowledge proof systems.

Dwork and Naor used zaps and the timing-based *timed commitments* of Boneh and Naor [6] (which in turn required some moderately hard functions) to construct a 3-round timing-based concurrent zero-knowledge argument for any language in NP [11].

Also in the spirit of "zero knowledge in the real world," Canetti, Goldreich, Goldwasser, and Micali considered the problem of *resettable* zero-knowledge: the prover's access to random bits is restricted to a single, fixed string, for example, in the case of a pseudo-random seed that is built into a smart card, and the prover may be re-started over and over, each time using exactly the same random bits [7]. The techniques developed in [7] apply to the protocol of Kilian, Petrank, and Richardson, to yield a polylogarithmic-round resettable black-box proof system for any language in NP. The smart-card setting was also considered by Impagliazzo, Naor, Reingold, and Shamir, who noted the connection to concurrent zero-knowledge [25].

Changing perspective, Barak, Goldreich, Goldwasser, and Lindell examined the question of *resettably-sound* zero-knowledge, in which the verifier, rather than the prover, has limited access to random bits and can be re-started using the same random bits every time. Here the difficulty is to ensure soundness [3]. They showed that constant-round resettably-sound zero-knowledge arguments exist if collision-free hash functions exist, while resettably-sound zero-knowledge proof systems exist only for languages in P/poly. Their constructions make heavy use of a result of Barak, who showed for the first time how a simulator can explicitly make use of knowledge of a (cheating) verifier's *program*. Barak obtained a constant-round, public-coin, *non-black-box* zero-knowledge proof system for any language in NP [2]; soundness is based on the existence of collision-free hash functions.

2 Review of Cryptographic Primitives

In this section we briefly review the definitions of semantic security, non-interactive zero-knowledge, and zaps, and state the properties of the particular constructions from the literature that we rely on in the current work. We let poly(n) denote a function that is bounded above by an^d for some constants a and d, and let $\nu(n)$ denote a function that grows more slowly than the reciprocal of any polynomial in n.

2.1 Semantic Security and Probabilistic Encryption

In their seminal paper [23] Goldwasser and Micali give the following definition of a probabilistic public-key cryptosystem.

Definition 2.1 (Probabilistic Public-Key Cryptosystem (PPKC)) *A* probabilistic public-key cryptosystem $\mathcal{E}$ *consists of*

1. *a probabilistic polynomial-time key generator* $\mathcal{G}$*:* $\mathcal{G}(1^k)$ *outputs strings* (e, d)*; we let* $\mathcal{G}_1(1^k)$ *denote the first output,* e*, and* $\mathcal{G}_2(1^k)$ *denote the second output,* d*; in our case, we relax this to expected polynomial time;*

2. *a polynomial-time encryption function* $E(e, b, r) = E_e(b, r)$ *where* $b \in \{0, 1\}$*,* $|r| = p(n)$*, for some polynomial* p*;* $E_e(b)$ *denotes the probabilistic function where* r *is chosen at random;*

3. *a polynomial-time decryption function* $D(d, e, c) = D_d(e, c)$*;*

4. *if* $\mathcal{G}(1^k)$ *outputs* (e, d)*, then* $\forall b \in \{0, 1\}$ $\forall r \in \{0, 1\}^{p(n)}$
   $D_d(e, E_e(b, r)) = b$*.*

The restriction to 1-bit messages is arbitrary; longer messages may be encrypted bit-by-bit.

The fundamental notion for security against a chosen plaintext attack[2] established in [23] is *semantic security*. This is known to be equivalent to *indistinguishability of encryptions*, for which in turn there are several equivalent definitions. The following, appearing in [17], works best for our proof of zero-knowledge.

Definition 2.2 (Indistinguishability of Encryptions) *A PPKC* $(\mathcal{G}, E, D)$ *has* indistinguishability of encryptions *if for every polynomial-size circuit family* $\{T_k\}$*, every polynomial* p*, all sufficiently large* k*, and every* $x, y \in \{0,1\}^{\mathrm{poly}(k)}$*, where* $|x| = |y|$*,*

$$|\Pr[T_k(G_1(1^k), E_{G_1(1^k)}(x)) = 1] - \Pr[T_k(G_1(1^k), E_{G_1(1^k)}(y)) = 1]| < \frac{1}{p(k)}$$

where the probability space in each term is over the random coins of G_1 *and* E*.*[3]

Remark 2.1 We will abuse notation in the following common fashion. We will use E to denote E_e and D to denote D_d, and we will sometimes say that E (or E/D) is the output of $\mathcal{G}(1^k)$. We write $c \in_R E(x)$ to mean that c is chosen randomly from the encryptions of x by E.

Goldwasser and Micali suggested a scheme based on the (assumed) hardness of distinguishing quadratic residues modulo a composite N from non-quadratic residues modulo N. Since two of our results use this cryptosystem, we describe it in detail.

Define $Q_n(z) = 1$ if and only if $\exists r\, r^2 \equiv z \bmod n$. If p prime then $Q_p(z)$ is easy to compute (via Fermat's Little Theorem). For p prime, the Jacobi symbol, $\left(\frac{z}{p}\right)$, is defined as

$$\left(\frac{z}{p}\right) = \begin{cases} 1 & \text{if } Q_p(z) = 1 \\ -1 & \text{if } Q_p(z) = 0 \end{cases}$$

For $N = pq$, where p, q are prime, define

$$\left(\frac{z}{N}\right) = \left(\frac{z}{p}\right)\left(\frac{z}{q}\right)$$

Without the factorization p, q computing $\left(\frac{z}{N}\right)$ is easy; but computing $Q_N(z)$ appears to be hard. Goldwasser and Micali's scheme is now easily described:

1. *Public Key:* N where $N = pq$ is a product of two primes, both congruent to 3 mod 4 (*i.e.*, N is a *Blum integer*). For technical reasons we insist that the key generator only produces $N = pq$ for which the factors p and q have been *certified* via the Goldwasser-Kilian primality test [22]. Note that -1 is a quadratic non-residue modulo N with Jacobi symbol $\left(\frac{-1}{N}\right) = 1$.

2. *Private Key:* Primes p and q, together with their certificates.

[2] A chosen plaintext attack is the weakest notion of an attack that makes sense in the public-key model: since the encryption key is public, the adversary can compute encryptions of any plaintext message at any time.

[3] Any auxiliary information to the distinguisher, such as the result of a polynomial-length chosen plaintext attack, can be built into the distinguisher circuit; so a separate auxiliary input to the distinguisher is not neeeded.

3. *Encryption:* Bit-by-bit.
   Encryptions of 0: $r^2 \bmod N$ where $r \in_R \mathbb{Z}_N^*$
   Encryptions of 1: $-r^2 \bmod N$ where $r \in_R \mathbb{Z}_N^*$.

4. *Decryption of Ciphertext c:* $1 - Q_N(c) = 1 - Q_p(c) \cdot Q_q(c)$.

Indistinguishability of encryptions is achieved under the following so-called Quadratic Residuousity Assumption (taken from [23]):

> Let P_1 be a fixed polynomial. For each integer k, let C be a circuit with $2k$-bit inputs and one Boolean output. Let C_k be the minimum size of circuits C such that for a fraction $\frac{1}{P_1(k)}$ of the products N of two k-bit primes, $C(N,x) = 1$ iff x is a quadratic residue modulo N, for all x such that $(\frac{x}{N}) = 1$. Then for all polynomials Q, for all sufficiently large k: $C_k > Q(k)$.

Property 1 *The Goldwasser-Micali cryptosystem is "XOR-malleable": given two 1-bit ciphertexts (that is, ciphertexts of single bits) α and β, the decryption of $\alpha\beta \bmod N$ is the exclusive-or of the decryptions of α and of β.*

Property 2 *The set of all valid public keys of the Goldwasser-Micali cryptosystem is in NP; moreover, every execution of the public-key cryptosystem generator produces a certified Blum integer.*

2.2 Interactive Proof Systems and Zero Knowledge

The concepts described here are due to Goldwasser, Micali, and Rackoff [24]. The presentation is adapted from [11]. We will be interested in the case in which the prover is polynomial-time bounded and the language L belongs to NP. For $t \in L$, let $W_L(t)$ denote the set of strings that witness the membership of t in L. We assume that the prover has a witness $w \in W_L(t)$ as an auxiliary input; generally, w is not available to the verifier. However, the verifier has an auxiliary input v which may be thought of as extra "advice" to the verifier. 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 t and w, and V has inputs t and v. 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 t):

1. *Completeness*: For all $t \in L$, all $w \in W_L(t)$, and all v,
$$\Pr[(P,V)(t,w,v) \text{ accepts}] > 1 - \nu(n)$$

2. *Soundness*: For all probabilistic polynomial-time machines P^*, for all $t \notin L$, for all w, v,
$$\Pr[(P^*,V)(t,w,v) \text{ accepts}] < \nu(n)$$

where the probability spaces are over the coin flips of both machines. The parameter n plays the role of a *security parameter*. In our formal treatment, we use k to denote the security parameter and allow it to be different than $n = |t|$, although this has no real effect on the definitions because we assume that k and $|t|$ are polynomially related.

Let $(P, V)(t, w, v)$ denote the transcript of the conversation between P and V when P's input is (t, w) and V's input is (t, v). 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 tests T and all t, w, v such that $w \in W_L(t)$,

$$|\Pr[T((P, V')(t, w, v))] - \Pr[T(S(t, v))]| < \nu(n) \tag{1}$$

where the probability in the first (resp., second) term is over the random choices of P and V' (resp., S). The system is *black-box* zero-knowledge if the simulator requires only oracle access to V' in order to generate the simulated conversations.

We distinguish between the terms *proof system* and *argument system* (or, simply, *arguments*). In a proof system, soundness holds independent of the computational power of the prover; in an argument system soundness is only guaranteed if the prover is restricted to probabilistic polynomial time. Our constructs are all of proof systems, although in the timing-based proofs the provers are explicitly restricted to an amount of computation determined by the time bound. In all our constructions the good prover can always carry out its protocol in probabilistic polynomial time.

2.3 Noninteractive Zero-Knowledge Proof Systems

The following discussion is based on [9, 14, 29, 11]: 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 $t \in L$. P and V initially share a string σ which is trusted to have been chosen at random; the length of σ is polynomial in the security parameter k and the length of t. To prove membership of a string t in $L_n = L \cap \{0,1\}^n$, P sends a message π as a proof of membership. V decides whether to accept or to reject the proof π as function of t and σ. 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 [14, 26]). We assume that the shared string σ is generated according to the uniform distribution on strings of length polynomial in k and n, where the polynomial depends on the particular protocol.

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

For the system to be zero-knowledge, there must exist a simulator *Sim* which, on input t, generates a conversation (σ, π). Let $Sim(t)$ be the distribution on the conversations that *Sim* generates on input t, let $Sim_U(t) = Sim_U$ be the distribution on the σ part of the conversation, and let $Sim_P(t)$ be the distribution on the proof component. In the definitions of [4, 14] the simulator has two steps: it first outputs Sim_U without knowing t, and then, given t, it outputs $Sim_P(t)$.

Because we do not explicitly use NIZKs in this paper, the formal definition is deferred to the Appendix. 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 $O(Mk^2)$ [26]; zero-knowledge depends on the existence of nonuniform one-way permutations (or trapdoor permutations, in the case of efficient provers).

2.4 Witness-Indistinguishability

The concept of *witness-indistinguishability* was proposed by Feige and Shamir [16] as a relaxation of zero-knowledge. Unlike the case with zero-knowledge, witness indistinguishability is closed under parallel and concurrent composition.

Definition 2.3 (Witness-Indistinguishability) *A proof system (P, V) is* witness indistinguishable *if for any polynomial-time V', for all $t \in L$, for all $w_1, w_2 \in W_L(t)$, and for all auxiliary inputs v to V' of length polynomial in $|t|$, the distribution on the views of V' following an execution $(P, V')(t, w_1, v)$ is non-uniform polynomial-time indistinguishable from the distribution on the views of V' following an execution $(P, V')(t, w_2, v)$.*

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

Theorem 2.1 ([16]) *Every zero-knowledge protocol is witness-indistinguishable.*

Theorem 2.2 ([16]) *Witness-indistinguishability is preserved under parallel and concurrent composition of protocols.*

As shown in [14], any NIZK satisfying Definition A.1 is also *general witness-indistinguishable*:

Definition 2.4 ([14]) *A NIZK for language L in NP is* general witness-indistinguishable *if for all non-uniform polynomial-time distinguishers $T = \{T_\ell\}_{\ell \geq 1}$, $\forall c\, \exists n_0\, \forall n > n_0\, \forall m < n^c$, for any sequence $\{(t_i, w_i^1, w_i^2)\}_{i=1}^m$ chosen by T where $t_i \in L_n$ and $w_i^1, w_i^2 \in W_L(t_i)$ for all $1 \leq i \leq m$:*

$$|\Pr[T(P(t_1, w_1^1, \sigma), \ldots, P(t_m, w_m^1, \sigma)) = 1] - \Pr[T(P(t_1, w_1^2, \sigma), \ldots, P(t_m, w_m^2, \sigma)) = 1]| < \nu(n)$$

where the probability space is that of P's random coins and the choice of σ.

General witness-indistinguishability implies that (1) the same common string σ can be used repeatedly without compromising witness-indistinguishability, and (2) we have witness-indistinguishability even if $t_1 = \ldots = t_m$, which is the case of interest in constructing zaps from NIZKs.

2.5 Zaps

Zaps are a class of 2-round public-coin witness-indistinguishable proof systems. Zaps were defined and constructed by Dwork and Naor, and the material in this section borrows heavily from their presentation [11].

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

Completeness: Given t and a witness $w \in W_L(x)$, and a first-round ρ, the prover, running in time polynomial in $|t|$, can with overwhelming probability generate a proof π 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 t, ρ, and π (i.e., a zap is a public-coin protocol).

Soundness: With overwhelming probability over choice of ρ, there exists no $t' \notin L$ and round-2 message π such that the verifier accepts (t', ρ, π).

Witness-Indistinguishability: Let $w, w' \in W_L(t)$ for $t \in L$. Then $\forall \rho$, the distribution on π when the prover has input (t, w) and the distribution on π when the prover has input (t, w') are non-uniform polynomial-time indistinguishable.

It follows immediately from the definitions that a zap yields a *non-constructive* single round witness-indistinguishable protocol; informally, the first-round message, ρ, can be fixed once and for all, without hurting any of the completeness, soundness, or witness-indistinguishable properties (see [11] for details).

Zaps and NIZKs are incomparable: zaps do not require a common guaranteed random string, but NIZKs provide more provable protection of the witness than do zaps: since NIZKs are zero-knowledge, they can be simulated without access to the witness. Zaps provide no such guarantee. Dwork and Naor showed that zaps exist in the real world if and only if NIZKs exist in the world in which the prover and verifier have access to a common guaranteed-random string[4]. In this paper the NIZK system of choice will be that of Kilian and Petrank [26], which relies on the existence of non-uniform one-way permutations (or trapdoor permutations, if the prover is to be efficient).

As noted in [11], by having the prover's random bits in a zap be a pseudo-random function of the verifier's initial message and the input, zaps yield resettable witness-indistinguishability.

3 Description of the Protocol

In this section we give a 2-round, resource-bounded, zero-knowledge protocol for any language in NP. We call this protocol ZEBRA, for Zero-Knowledge Bounded-Resource. The protocol depends on a particular language L, function f, committing probabilistic public-key cryptosystem $\mathcal{E}$, and zap scheme $\mathcal{Z}$. Below we discuss f, $\mathcal{E}$, and L, then give the protocol, and then discuss $\mathcal{Z}$.

3.1 The Computational Model

Computational "devices" (*e.g.*, proof auditors and (cheating) provers) can be either Turing machines or random-access machines, provided that the same choice is made throughout. Each device is deterministic and uniform. Probabilistic devices are permitted by allowing one input to be a string of random bits. Nonuniform devices are permitted by allowing one input to be an advice string which can be chosen differently for each value of the security parameter k. Two measures of computational resource are used: the amount of time used by a device and the amount of advice given to a device (see Remark 3.1).

[4][11] also showed that zaps can be constructed if and only if there exists an *approximate verifiable pseudo-random bit generator*. Roughly speaking, these are pseudo-random strings in which a party knowing the pseudo-random seed can construct verifiable proofs of the bits of the string, and the revealed values and their proofs do not leak any usable information about the unrevealed bits. Approximate VPRGs for efficient provers can be constructed based on any family of certified trapdoor permutations with a common domain, together with a hard-core predicate.

3.2 The Function f

Let $f : \{0,1\}^* \to \{0,1\}^*$ such that $|f(x)| = |x|$ for all x.[5] For $\ell \geq 1$, let f_ℓ denote f restricted to $\{0,1\}^\ell$. We assume that f is in FP/poly, *i.e.*, it is computable by a family of polynomial-size circuits. If this polynomial is $O(\ell^c)$, then we say that f *has* $O(\ell^c)$*-size circuits*. The definition of f having $O(\ell^c)$*-size XOR-circuits* is similar, except that circuits may contain only XOR-gates. We say that f has $O(\ell^c)$*-size circuits (resp., XOR-circuits) constructible in time* $O(\ell^{c'})$ if there is a deterministic $O(\ell^{c'})$-time device that, when given 1^ℓ, constructs a circuit (resp., an XOR-circuit) of size $O(\ell^c)$ that computes f_ℓ.

3.3 The Probabilistic Public-Key Cryptosystem

Fix some bit-by-bit PPKC $\mathcal{E}$ with the property that, when given a security parameter 1^k, $\mathcal{G}$ probabilistically outputs an encryption/decryption pair (E, D) and a witness $w(E, D)$ that (E, D) is a valid key pair. In the Quadratic-Residuosity-based scheme of Goldwasser and Micali [23] (see Section 2.1), for example, if one insists that the primes be certified, then E is a Blum integer N, D is its factorization $N = pq$, and $w(E, D)$ is an NP-proof [22] that p and q are prime (the check that $N = pq$ and $p, q \equiv 3 \bmod 4$ can be done directly). Note that, given only E, the combination of D and $w(E, D)$ is a witness that D is an algorithm that correctly decrypts E. Let $\mathcal{G}(k)$ be the set of "legal" E's for security parameter k, *i.e.*, the E's that can be produced by $\mathcal{G}_1$ on input 1^k. Assume for simplicity that $|E(0)| = |E(1)| = k$ for all $E \in \mathcal{G}(k)$.[6] Extend E to encrypt strings of bits in the usual way (bit by bit). Thus, if $|x| = \ell$ then $|E(x)| = \ell k$. Let $E(x)$ be the set of legal encryptions of x using E. Note that, given E, x, and α, the combination $(D, w(E, D))$ is a witness that $\alpha \in E(x)$; to check this witness, check the witness $w(E, D)$ and check that $x = D(\alpha)$.

Let $\mathcal{E}$ be an encryption scheme, and let $\odot$ denote a binary Boolean operation. The function $\mu(E, \gamma_1, \gamma_2)$ is a $\odot$*-mallet for* $\mathcal{E}$ if (i) for all $k \geq 1$, all $E \in \mathcal{G}(k)$, all $b_1, b_2 \in \{0,1\}$, all $\gamma_1 \in E(b_1)$ and all $\gamma_2 \in E(b_2)$, $\mu(E, \gamma_1, \gamma_2) \in E(b_1 \odot b_2)$, and (ii) μ is computable in poly(k) time for $E \in \mathcal{G}(k)$ and $|\gamma_1| = |\gamma_2| = k$. A cryptosystem $\mathcal{E}$ is *XOR-malleable* if it has an XOR-mallet. A cryptosystem $\mathcal{E}$ is *completely-malleable* if it has an XOR-mallet and an AND-mallet.

Given the function f, let g be a function such that: (i) if $\alpha \in E(x)$ then $g(E, \alpha) \in E(f(x))$, and (ii) g is computable in time $O(s(\ell) \cdot k^b)$ where b is a constant depending only on $\mathcal{E}$ and where f has $s(\ell)$-size circuits. These are the only properties of g used in the proofs. Intuitively, for implementation, g on those inputs α of length ℓk is defined by taking a fixed circuit C that computes f on inputs with $|x| = \ell$ and "pushing α through C". The details will depend on the particular version of the result, *e.g.*: (i) the encryption scheme is a completely malleable commitment scheme and the circuit C contains ANDs and XORs; (ii) the encryption scheme is the Goldwasser-Micali scheme, which is XOR-malleable, and C contains only XOR gates.

---

[5] Only the case $|f(x)| = |x|$ is considered because it is sufficient for our purposes and avoids introducing another parameter. However, it will be obvious that the entire development can be done without this restriction.

[6] Our results hold for $|E(0)| = |E(1)| = \text{poly}(k)$, and we later note a minor modification needed in this case (Remark 7.4).

3.4 The Protocol for the Language L

Fix a language $L \in \mathrm{NP}$. Recall that, for each $t \in L$, $W_L(t)$ is the set of witnesses proving that $t \in L$, and $|w|$ is polynomial in $|t|$ for all $w \in W_L(t)$. The protocol depends on a particular function f, cryptosystem $\mathcal{E}$, and zap scheme $\mathcal{Z}$. It also depends on three constants a, d, and e (to be chosen later). $\ell = \lceil k^d \rceil$ and ak^e is the time bound that the prover must meet. The degrees of polynomials in polynomial time bounds, *e.g.*, d and e, do not have to be integers. When a noninteger must be rounded to an integer, we assume that it is rounded up, unless noted otherwise. For example, the definition of ℓ above will be written $\ell = k^d$. In this paper e is used as a mnemonic for "easy"; it does not denote the base of the natural logarithm. The common input to P and V includes a security parameter 1^k. An input to P is a "theorem" t; if $t \in L$ then P has a witness $w \in W_L(t)$. We assume that there is a polynomial $\mathrm{IL}(k)$ (Instance Length) such that $|t| \leq \mathrm{IL}(k)$.[7] As part of its second-round message to the verifier, the prover sends t. The "theorem" t is given to V by P, rather than as an input to V, because we want to allow a "cheating prover" to choose $t \notin L$ after seeing the first-round message from V and attempt to make V accept this false theorem[8].

In order to better present the new ideas, we do not state all of our results in full generality. In particular: (i) we assume that d is large enough that, for all $E \in \mathcal{G}(k)$, there is a certificate of length $O(k^{d+1})$ that E is a valid encryption key[9]; (ii) we design ZEBRA to have soundness error $\leq 2^{-k}$. It will be obvious how to give the definitions and results in greater generality (for example, the soundness error of ZEBRA is a variable) at a cost of a more complicated presentation. To achieve (ii), we use a zap protocol that has soundness error $o(2^{-k})$ (for definiteness, say 4^{-k}) when it is used in ZEBRA with security parameter k. The description of ZEBRA is in Figure 1.

Presently, we will describe the use of zaps in more detail. First, let us briefly give some intuition for the protocol. The use of *two* keys is so that we can give a *witness-hiding* proof of the fact that at least one is a valid key (witness-indistinguishability is useless if there is only one witness). If the verifier had another way of checking that a key E chosen by the prover were valid, then we could get by with just a single E. This would allow us to drop the first zap ("$E_1 \in \mathcal{G}(k) \vee E_2 \in \mathcal{G}(k)$") and eliminate one of the second two zaps. So the prover would be giving a zap of the statement "$t \in L \vee \alpha \in E(x)$". In contrast, in the Flawed Prototype, the prover sends a zap of "$t \in L \vee \beta \in E(f(x))$". In Protocol ZEBRA the verifier, by means of g, can check that *if* $\alpha \in E(x)$ *then* $\beta \in E(f(x))$, so logically, if the zap is accepted, the verifier can deduce that either $t \in L$ or $\beta \in E(f(x))$: exactly as in the Flawed Prototype. Thus by means of g (and a sufficiently malleable cryptosystem) we are able to obtain the "miraculously short zap" assumed in the discussion of the Flawed Prototype. As stated in the Introduction, we handle the remaining problem (faulty reasoning about soundness) through the concept of a proof auditor, described in the next section.

[7] For results where the cheating prover is advice-bounded, but not time-bounded, this assumption is not needed, provided that the running time of the good prover can depend polynomially on $|t|$ as well as k.

[8] The proof of zero-knowledge holds even if the theorem t is fixed in advance, giving the verifier more information before preparing its Round 1 message.

[9] This assumption is satisfied in the case of the Goldwasser-Micali PPKC, because we always choose $d \geq 1$ and k-bit primes have certificates of length $O(k^2)$ using the Goldwasser-Kilian test [22].

Protocol ZEBRA
For language L.
Based on: function f, committing encryption scheme $\mathcal{E}$ with generator $\mathcal{G}$, zap scheme $\mathcal{Z}$, and constants a, d, e.
Inputs to P: 1^k, t, and w.
Inputs to V: 1^k, history.

0. Before the protocol starts, P does the following precomputation:

   (a) Call $\mathcal{G}(k)$ twice to produce two encryption keys E_1 and E_2 and the certificates s_1 and s_2 proving that E_1 and E_2, respectively, are valid keys.

   (b) Let $\ell = k^d$ and $x^* \in_R \{0,1\}^\ell$, and for $i = 1, 2$, find $\alpha_i \in_R E_i(x^*)$ and $\beta_i = g(E_i, \alpha_i)$.

   (Note: the length of the precomputed information, $E_i, s_i, \alpha_i, \beta_i$ for $i = 1, 2$, is $O(\ell k)$.)

1. $V \longrightarrow P$: V chooses $x \in_R \{0,1\}^\ell$ and an additional string ρ of random bits that will used in zaps, and sends x and ρ to P.

2. $P \longrightarrow V$:

   (a) Send t to V, and for $i = 1, 2$, send E_i, α_i, and β_i.

   (b) Choose $i_0 \in_R \{1, 2\}$, and using the witness s_{i_0} proving that $E_{i_0} \in \mathcal{G}(k)$, send the second-round message of a zap that
   $$E_1 \in \mathcal{G}(k) \ \vee \ E_2 \in \mathcal{G}(k). \tag{2}$$

   (c) For $i = 1, 2$, using the witness w proving that $t \in L$, send the second-round message of a zap that
   $$t \in L \ \vee \ \alpha_i \in E_i(x). \tag{3}$$

3. V accepts iff:

   (a) P responds within time ak^e, and

   (b) $\beta_i = g(E_i, \alpha_i)$ for $i = 1, 2$, and

   (c) the verifiers for the three zaps in (2) and (3) all accept.

Figure 1: Protocol ZEBRA.

3.5 The Use of Zaps

We now describe the use of zaps in more detail. Let L_1 be the language containing the instances of the form $\langle k, E_1, E_2\rangle$ such that either $E_1 \in \mathcal{G}(k)$ or $E_2 \in \mathcal{G}(k)$. Note that L_1 belongs to NP, and a witness for $\langle k, E_1, E_2\rangle \in L_1$ is $w(E, D)$ where either $E = E_1$ or $E = E_2$. Let L_2 be the language containing the instances of the form $\langle t, \alpha, E, x\rangle$ such that either $t \in L$ or $\alpha \in E(x)$. Note that L_2 belongs to NP, and a witness for $\langle t, \alpha, E, x\rangle \in L_2$ is either $w \in W_L(t)$ (if $t \in L$) or the string of random bits used in generating α (if $\alpha \in E(x)$). For language L_i, we can take the length of an instance to be a polynomial function of k, say $\lambda_i(k)$. The prover P splits the received string ρ as $\rho = \rho_1\rho_2$ where, for $i = 1, 2$, $|\rho_i|$ is the number of random bits received by the zap's prover for L_i when the instance has length $\lambda_i(k)$. In step 2(b), P sends a proof π_1 such that the zap's verifier accepts $(\langle k, E_1, E_2\rangle, \rho_1, \pi_1)$. Step 2(c) is done similarly, using the same random string ρ_2 for both $i = 1, 2$.

Finally, for reasons of time-efficiency of the prover, zaps of statements of the form "$t \in L \vee \alpha_i \in E_i(x)$" are broken into $\ell = |x|$ zaps of statements "$t \in L \vee \alpha_{ij} \in E_i(x_j)$," where α_{ij} is the jth encryption in the bit-by-bit ciphertext α_i. Note that each of these "smaller" statements can be expressed by a circuit of size at most polynomial in k, the security parameter. Consequently, if the zaps are based on the NIZK of [26], all $2\ell + 1$ zaps can be constructed in time $O(\ell\,\mathrm{poly}(k))$, avoiding powers of ℓ (see Section 5 for precise calculations). This last point is important only when the prover is time-bounded during execution of the protocol; if the prover is not time-bounded during execution of the protocol, as in our results for advice-bounded provers, this breaking of "large" zaps into "smaller" zaps is not necessary.

Remark 3.1

1. The results for advice-bounded provers do not rely on time, so in the model with advice-bounded provers we may modify Protocol ZEBRA to omit the timeliness test 3(a) for P's response.
2. In the case of time-bounded provers, computation of the function f should ideally not be amenable to parallelism. If computation of f is parallelizable then the security parameter k should be chosen as a function of the number of processors j realistically available to the adversary. In particular, we need that $T \ll q(k)/(j^{1+\varepsilon})$ for some constant $\varepsilon > 0$, where T is the time bound in the timing constraint. Since T must be sufficiently large for the good prover to complete its task, it too will be a function of k. Thus, for example, if $T = k^{1.5}$ and $q(k) = k^2$, then we must choose $\sqrt{k} \gg j^{1+\varepsilon}$.

4 Proof Auditors for Function Evaluation

Informally, an "i.o. proof auditor" $\mathcal{A}$ for the function f consists of three parts (see Figure 2). One is a sequence of advice strings v_ℓ, one for each input (to f) length ℓ. The second part is a deterministic device $adv(\mathcal{A})$, called the *advisor*, that takes inputs 1^ℓ, a random string r_1, and the advice string v_ℓ. The lengths of r_1 and v and the running time of $adv(\mathcal{A})$ are polynomial in ℓ. It produces an output v'. The third part is a deterministic device $aud(\mathcal{A})$, the *auditor*, that takes inputs x, y, z, r_2, v' where $|x| = |y|$, z is a supposed "proof" that $y = f(x)$, r_2 is a random string, and v' is chosen by the advisor $adv(\mathcal{A})$ where $\ell = |x|$. The computational part of an i.o. proof auditor is split into

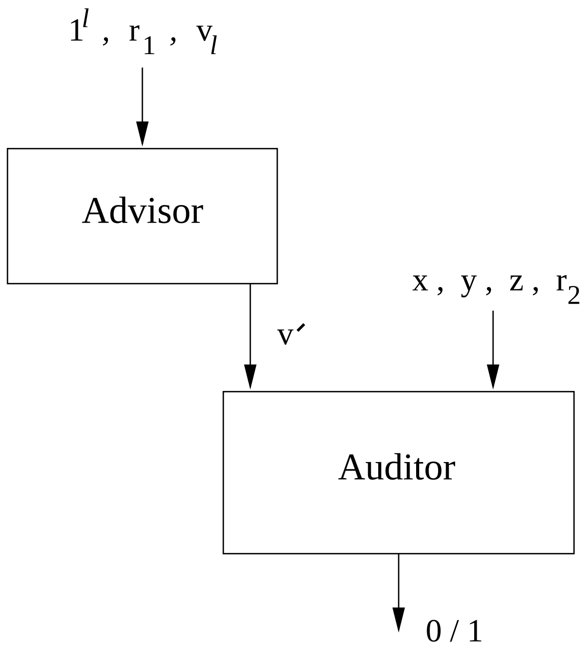

Figure 2: An i.o. proof auditor

two parts, an advisor and an auditor, because when making a connection between a prover and an i.o. proof auditor, the advisor corresponds to the precomputation of the prover and the auditor corresponds to the computation that the prover does during the protocol to finds its response to the verifier's message. One way in which this split is helpful is to allow $|v'| \ll |v|$; thus, the advisor (and the prover's precomputation) can perform a probabilistic polynomial-time computation on a "long" advice string v to produce a "short" advice string v' that is used by the auditor (and the prover's response computation). This is useful, for example, in a case where the description of f restricted to $\{0,1\}^\ell$ requires ℓ^2 bits (this description is contained in v_ℓ) but we want the advice string v' given to the auditor to have smaller length ℓ^c for some $c < 2$. A formal definition follows.

With essentially no loss of generality, we assume that all advice strings are binary strings. We do this so that the length of v' bounds the number of bits of information in v', not just the amount of information to within a constant factor. We are not interested in separately bounding the number of random bits. For simplicity when defining a computational device, for example, an i.o. proof auditor, we let the function R be an upper bound on the number of random bits needed by the device; in a particular computation, some prefix of these random bits is used.

Definition 4.1 *An* i.o. proof auditor with proof length bound $Z(\ell)$, advisor advice bound $A_1(\ell)$, auditor advice bound $A_2(\ell)$, and advisor (resp., auditor) time bound $T_1(\ell)$ (resp., $T_2(\ell)$) *consists of three parts: a sequence $\{v_\ell\}_{\ell\geq 1}$ of binary advice strings where $|v_\ell| = A_1(\ell)$ for all ℓ, an* advisor *adv*($\mathcal{A}$) *and an* auditor *aud*($\mathcal{A}$) *(see Figure 2). The advisor adv($\mathcal{A}$) takes inputs 1^ℓ, r_1, v_ℓ with $|r_1| = R(\ell)$, and produces a binary output string v' with $|v'| = A_2(\ell)$, while running in time $T_1(\ell)$. The auditor aud($\mathcal{A}$) takes inputs x, y, z, r_2, v' with $\ell \stackrel{\text{def}}{=} |x| = |y|$, $|z| = Z(\ell)$, $|r_2| = R(\ell)$, and v', and produces a 0/1 output, while running in time $T_2(\ell)$.*

Let $f : \{0,1\}^* \to \{0,1\}^*$ with $|f(x)| = |x|$ for all x. Then $\mathcal{A}$ is an i.o. proof auditor for f with correctness bound $\chi(\ell)$ *if for some infinite set of ℓ's, proofs z that $y = f(x)$ are checked correctly with probability at least $\chi(\ell)$ over random x, r_1, r_2; i.e., there is an infinite set I of positive integers such that, for all $\ell \in I$,*

$$\Pr_{x,r_1,r_2}[\forall y\,(\,\exists z\,(aud(\mathcal{A})(x,y,z,r_2,v') = 1) \;\Leftrightarrow\; y = f(x)\,)\,] \geq \chi(\ell) \tag{4}$$

where $v' = adv(\mathcal{A})(1^\ell, r_1, v_\ell)$ and where the probability is over $x \in_R \{0,1\}^\ell$ and random r_1, r_2.

Here, "i.o." stands for both "infinitely often" (because it is enough that the proof auditor works for some infinite set I of $\ell = |x|$) and "input-output" (because the proof auditor is checking a proof z that $y = f(x)$).

Of course, proof auditing f is essentially no harder than evaluating f. If $\mathrm{P} \neq \mathrm{NP} \cap \text{co-NP}$, then there are (0/1-valued) functions that have polynomial-time proof auditors, but cannot be evaluated in polynomial time. We are interested in functions for which proof auditing is "almost as hard" as evaluating.

5 The Class of Provers

We now describe a general class of provers that includes the "good prover" P in ZEBRA. We then describe a closely related class of "cheating provers" that are permitted when showing the soundness of protocol ZEBRA. Recall that the definition of ZEBRA depends on a number d.

Definition 5.1

1. *A* good prover P with precomputation (resp., protocol) advice bound $A_1(k)$ (resp., $A_2(k)$) and precomputation (resp., protocol) time bound $T_1(k)$ (resp., $T_2(k)$) *consists of three parts: a sequence $\{v_k\}_{k \geq 1}$ of binary advice strings where $|v_k| = A_1(k)$ for all k, a precomputation device pre(P), and a protocol device pro(P) (see Figure 3).*

2. *The precomputation device pre(P) has inputs: a security parameter 1^k, a random string r_1 with $|r_1| = R(k)$, and the advice string v_k. The running time is at most $T_1(k)$ and the output is a binary string v' with $|v'| = A_2(k)$.*

3. *The protocol device pro(P) has inputs: a security parameter 1^k, a "theorem" t of length $\leq \mathrm{IL}(k)$, a "witness" w, a random string r_2 with $|r_2| = R(k)$, an advice string v' with $|v'| = A_2(k)$ obtained from the precomputation device, and strings x and ρ from the verifier where $|x| = \ell \stackrel{\text{def}}{=} k^d$. The running time is at most $T_2(k)$. The output consists of the information that is sent back to the verifier, namely, the string t, encryption functions E_1 and E_2, strings α_i and β_i of length ℓk for $i = 1, 2$, and strings π_i for $i = 1, 2, 3$ which are the second-round messages of the prover in the three zaps in (2) and (3) of protocol* ZEBRA.

4. *Let k be a number and let t and w be strings. The* acceptance probability *of (P, V) on $\langle t, w\rangle$ is the probability that the verifier V in the protocol accepts when the prover is P and the inputs to P are as above (again see Figure 3). The probability is over the choices of r_1, r_2, and the random choices made by V, that is, the random choice of x and ρ.*

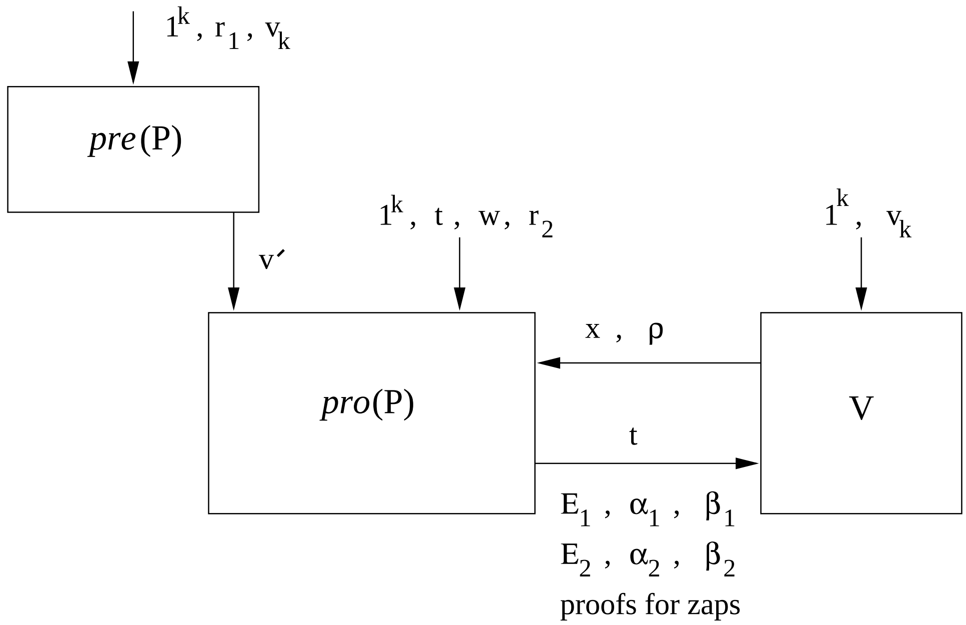

Figure 3: A good prover P.

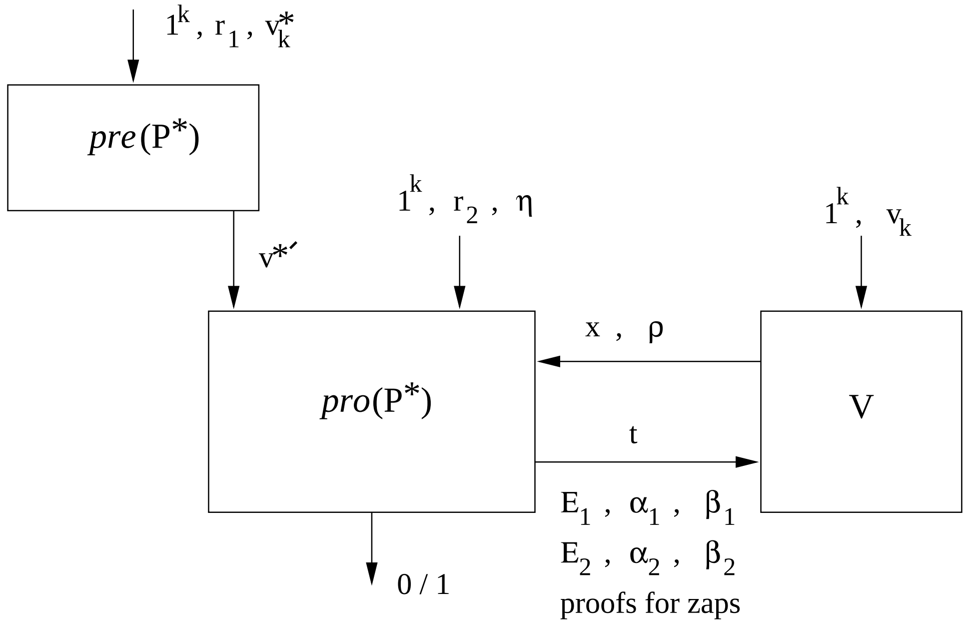

Figure 4: A cheating prover P^*.

We now define the class of "cheating provers" P^*. Two differences from good provers are:

1. *The sequence of binary advice strings $\{v_k^*\}_{k\geq 1}$ may be different than that used by P.*
2. *The protocol device pro(P^*) does not have inputs for t and w; as before, the second-round message to V includes a string t.*

To get stronger results: (i) we allow cheating provers to be nondeterministic – this is modeled by an additional "nondeterministic choice" input η to *pro*(P^*); and (ii) we let η depend on the other inputs to *pro*(P^*), including x and ρ, as we now explain. Associated with a cheating prover is a *nondeterminism bound* $N(k)$ such that $|\eta| = N(k)$. Because there can be a different η for each x and ρ, the definition requires some care. We cannot simply say that P^* succeeds if, for some $\eta = \eta(x, \rho)$, the verifier accepts some $t \notin L$. Because for each x there are E_i, α_i, β_i such that $E_i \in \mathcal{G}$, $\alpha_i \in E_i(x)$, and $\beta_i = g(E_i, \alpha_i)$ for $i = 1, 2$ (it might take more than time $O(k^e)$ to find β_i). Thus, if η contains a $t \notin L$ and these E_i, α_i, β_i, then P^* can make V accept a false theorem. What is missing from this (unuseable) definition is that a nondeterministic machine must *check* the validity of its nondeterministic choice. For example, a nondeterministic algorithm for satisfiability must check, before accepting, that its nondeterministic choice is a satisfying assignment. To make a cheating prover check η, we give *pro*(P^*) a 0/1-valued *check output* which intuitively means that *pro*(P^*) accepts (1) or rejects (0) η. When *pro*(P^*) accepts η it must be that this η can be used to cause V to accept some $t \notin L$. A formal definition follows. The structure of a cheating prover is shown in Figure 4. By giving *pro*(P^*) both a random input r_2 and a nondeterministic input η, both probabilistic and nondeterministic cheating provers are special cases.

Definition 5.2 *For $k \in \mathbb{N}$, let success(r_1, r_2, η, x, ρ) be the statement: "$t \notin L$ and V accepts when the inputs to P^* are $\langle 1^k, r_1, r_2, \eta, x, \rho\rangle$" ($t$ is an output of P^*). Let check(r_1, r_2, η, x, ρ) be the check output of pro(P^*). The* cheating probability *of (P^*, V) on k is the probability over r_1, r_2, x, ρ that*

$$\exists \eta\, (\mathit{check}(r_1, r_2, \eta, x, \rho) = 1) \;\wedge\; \forall \eta\, (\mathit{check}(r_1, r_2, \eta, x, \rho) = 1 \;\Rightarrow\; \mathit{success}(r_1, r_2, \eta, x, \rho)).$$

Note from Figures 3 and 4 that when the security parameter is k, the verifier V is given the same advice string v_k that the good prover P is given. However, when interacting with a cheating prover P^*, V is still given v_k even though P^* might be using a different advice string v_k^*. We use that v_k is shared between P and V; for example, v_k can contain information about f that both P and V need to compute g.

A cheating prover has three incomparable methods for choosing t. The first method is to choose t after seeing x and ρ. This has the advantage that t can depend on the particular x and ρ chosen by the verifier. It has the disadvantage that the computation of t must be done in time $O(k^e)$, for otherwise the verifier will reject. The second method is for t to be included in the advice string v_k^* and passed on to *pro*(P^*) as (part of) $v^{*\prime}$. An advantage here is that unlimited time can be spent to find t given k. A disadvantage is that both the precomputation and protocol advice bounds must be at least the length of t. The third method is to have *pre*(P^*) compute t. This has the advantage over the second method that the precomputation advice bound can be less than $|t|$ (even zero). A disadvantage is that the time to find t is bounded by the precomputation time of P^*; however, this bound can be larger than $O(k^e)$, so the third method is not subsumed by the first. For some of our specific results, we allow cheating provers to use unlimited precomputation time and advice. In this case the third method is subsumed by the second, although the first and second are still incomparable. If in addition the protocol time of the cheating prover is unlimited, as in our main advice-bounded result (Theorem 7.8), then the second and third methods are subsumed by the first.

Definition 5.3 *Let $\mathcal{P}$ be a subset of cheating provers.*[10] *The verifier V has* soundness error $\varepsilon(k)$ for L against the class $\mathcal{P}$, *if for all P^* in $\mathcal{P}$ and all $k \geq 1$, the cheating probability of (P^*, V) on k is at most $\varepsilon(k)$.*

The next lemma shows that the device P described in the protocol ZEBRA is a good prover, gives advice and time bounds for it, and shows that it has perfect completeness, *i.e.*, for all $t \in L$ and all $w \in W_L(t)$ the acceptance probability of (P, V) on $\langle t, w \rangle$ is 1, if the zap scheme $\mathcal{Z}$ has perfect completeness. The prover depends on the language L, the function f, the PPKC $\mathcal{E}$, the zap scheme $\mathcal{Z}$, and constants a, d, e. When we write, *e.g.*, $b = b(L, \mathcal{E}, \mathcal{Z})$ we mean that b is a constant that may depend on L, $\mathcal{E}$, and $\mathcal{Z}$, but not on the other objects a, d, e, or f.

Lemma 5.1 *Let c be such that f has $O(\ell^c)$-size circuits. There are constants $b = b(L, \mathcal{E}, \mathcal{Z})$ and $b' = b'(\mathcal{E})$ such that the following holds.*

1. *The device P described in protocol* ZEBRA *is a good prover.*

2. *The precomputation advice bound of P is $O(k^{cd} \log k)$. The precomputation time bound of P is $O(k^{cd+b'})$. If f has $O(\ell^c)$-size circuits constructible in time $O(\ell^{c'})$, then this advice bound can be decreased to zero if the time bound is increased to $O(k^{c'd} + k^{cd+b'})$.*

3. *The protocol advice bound of P is $O(k^{d+1})$. The protocol time bound of P is $O(k^{d+b})$.*

4. *If $e \geq d + b$ and a is large enough, then for all $t \in L$ and all $w \in W_L(t)$, the acceptance probability of (P, V) on $\langle t, w \rangle$ is $1 - \nu(k)$. If the zap scheme $\mathcal{Z}$ has perfect completeness, then this acceptance probability is 1.*

Proof. Let $pre(P)$ (resp., $pro(P)$) be the device that performs the computation in step 0 (resp., 2) of ZEBRA. Recall that $\ell = k^d$. The precomputation advice v_ℓ is a description of length $O(\ell^c \log \ell^c) = O(k^{cd} \log k)$ of a circuit C_ℓ of size $O(\ell^c)$ that computes f_ℓ. The precomputation device calls $\mathcal{G}$ twice (using r_1) to obtain $E_1, E_2 \in_R \mathcal{G}(k)$; this takes time $O(k^{b'})$ for some $b' = b'(\mathcal{E})$. Using r_1, it then chooses $x^* \in_R \{0,1\}^\ell$ and computes $\alpha_i \in_R E_i(x^*)$ for $i = 1, 2$; this takes time $O(\ell k^{b'})$. It then computes $\beta_i = g(E_i, \alpha_i)$ for $i = 1, 2$. Because we have assumed that g can be computed in time $O(s(\ell) \cdot k^{b'})$ where f_ℓ can be computed by a circuit of size $s(\ell)$, this takes time $O(\ell^c \cdot k^{b'}) = O(k^{cd+b'})$. If f has $O(\ell^c)$-size circuits constructible in time $O(\ell^{c'})$, then $pre(P)$ can construct the circuit C_ℓ in additional time $O(\ell^{c'})$, so the precomputation advice v can be empty.

Now consider the protocol device. The advice $E_i, s_i, \alpha_i, \beta_i$ for $i = 1, 2$ satisfies the advice bound $O(\ell k)$ $(= O(k^{d+1}))$. We show that the protocol time is $O(\ell k^b) = O(k^{d+b})$ for some constant b. In particular, choose b large enough that $b \geq 1$ and $\mathrm{IL}(k) = O(k^{d+b})$. The sending of t, E_i, α_i, β_i for

[10]For example, $\mathcal{P}$ might be defined by giving explicit functions for the time and advice bounds of the cheating prover.

$i = 1, 2$ can be done in time $O(\ell k + \mathrm{IL}(k)) = O(k^{d+b})$, because the α_i and β_i have length at most ℓk and these strings have been precomputed.

We now show that the remaining task, computing and sending the second-round messages for the three zaps in steps 2(b) and 2(c), can be done in time $O(\ell k^b)$ $(= O(k^{d+b}))$ for some constant b. We base our zaps on the NIZK scheme of Kilian and Petrank, who obtain a NIZK of length $O(sk^2)$ for satisfiability of a circuit of size s, with security parameter k [26]. A detailed study of their protocol shows that the prover's effort is linear in the number of so-called committed bits, assuming decommitment requires $O(1)$ time. Decommitment actually requires time polynomial in the number of public random bits required for each committed bit – in this case k. Since the total number of committed bits is $O(sk)$ the NIZK can be carried out in time $O(s \cdot \mathrm{poly}(k))$. Each statement of the form "$t \in L \vee \alpha_{ij} \in E_i(x_j)$," where α_{ij} is the jth encryption in the bit-by-bit ciphertext α_i, can be expressed by a circuit of size polynomial in k, so $s = \mathrm{poly}(k)$.

Finally, the construction of the zap requires a number m of copies of the NIZK. Specifically, if $t_{\mathtt{zap}}$ is the statement being proved by the zap, $|\sigma|$ is the length of the common random string for the NIZK, and q is the error probability of the NIZK, then to achieve soundness error δ we must choose m satisfying (see [11])

$$q^{m/|\sigma|} < \frac{\delta}{2^{|t_{\mathtt{zap}}|+|\sigma|}}. \tag{5}$$

In our case, $|\sigma|$ and $|t_{\mathtt{zap}}|$ are both polynomial in k, so by choosing m to be a sufficiently large polynomial in k we can achieve $\delta \leq 4^{-k}$.

To prove part 4 of the lemma, let $t \in L$ and $w \in W_L(t)$. If $e \geq d + b$ and a is large enough, the protocol time bound $O(k^{d+b})$ will satisfy the time bound ak^e used by the verifier in step 3(a). The test $\beta_i = g(E_i, \alpha_i)$ for $i = 1, 2$ in step 3(b) is true because α_i and β_i are chosen by P in step 0 to satisfy this test. Because P has a witness for each of the three zaps in (2) and (3) and each zap verifier accepts with probability at least $1 - \nu(k)$ (resp., 1 if $\mathcal{Z}$ has perfect completeness), it is clear that (P, V) accepts input $\langle t, w \rangle$ with probability $1 - \nu(k)$ (resp., 1). ■

Remark 5.1 Suppose that the clocks of the prover and the verifier do not run at exactly the same rate, but these rates are within some constant factor γ. If the running time of *pro*(P) is $a'k^e$ then we should choose $a = \gamma a'$ in the protocol. Then the protocol is still correct even if the verifier's clock runs γ times faster than the prover's clock. None of our results depends on the exact value of a in the protocol. In other words, it doesn't hurt if, for example, *pro*(P^*) uses a constant factor more time than *pro*(P).

6 A Connection Between Provers and i.o. Proof Auditors

In this section we show that a cheating prover P^* can be used to obtain an i.o. proof auditor for f. The advice and time bounds of the i.o. proof auditor $\mathcal{A}$ are related to those of P^*, and the correctness bound of $\mathcal{A}$ is related to the cheating probability of P^*.

Fix the language L, the function f, the PPKC $\mathcal{E}$, and the zap scheme $\mathcal{Z}$ used in ZEBRA. Recall that $\mathcal{Z}$ has soundness error bound 4^{-k} when used in ZEBRA with security parameter k. We first define three constants that appear in the statement of the theorem. Let c_1 be such that time $O(k^{c_1})$ is sufficient to decrypt a single bit. Let c_2 be such that time $O(k^{c_2})$ is sufficient to check, given

$E \in \mathcal{G}(k)$ and $(D, w(E,D))$, that $(D, w(E,D))$ is a valid witness that D is a decryption algorithm for E. Let c_3 be such that the length of $(D, w(E,D))$ is $O(k^{c_3})$.

Theorem 6.1 *There are constants $c_i = c_i(\mathcal{E})$ for $i = 1,2,3$ such that the following holds. Let P^* be a cheating prover that has precomputation time (resp., advice) bound $T_1(k)$ (resp., $A_1(k)$), protocol advice bound $A_2(k)$, and nondeterminism bound $N(k)$. Assume that there is an infinite set K of numbers such that, for all $k \in K$, the cheating probability of (P^*, V) on k is at least $\varepsilon(k)$. Then there is an i.o. proof auditor $\mathcal{A}$ for f having the following bounds, where $k = \lfloor \ell^{1/d} \rfloor$:*

$$\begin{aligned}
\textit{advisor advice} &= A_1(k) \\
\textit{advisor time} &= O(T_1(k) + \ell) \\
\textit{auditor advice} &= A_2(k) \\
\textit{auditor time} &= O(k^e + \ell k^{c_1} + k^{c_2}) \\
\textit{proof length} &= O(N(k) + k^{c_3}) \\
\textit{correctness} &= \varepsilon(k) - 2 \cdot 4^{-k}.
\end{aligned}$$

Proof. When reading the proof it might be helpful to refer back to Figures 2 and 4. We describe the three parts of an i.o. proof auditor $\mathcal{A}$ for f: the advice sequence, the advisor $adv(\mathcal{A})$, and the auditor $aud(\mathcal{A})$.[11] Let $\kappa(\ell)$ equal $\lfloor \ell^{1/d} \rfloor$. (This implies that $\kappa(\ell)$ is the inverse of $\ell = \lambda(k) \stackrel{\text{def}}{=} \lceil k^d \rceil$ on those ℓ's in the range of λ.) It is easy to see that $\kappa(\ell)$ can be computed in time $O(\ell)$.

We first define the advice sequence $\{v_\ell\}$. Let $\{v_k^*\}$ be the advice sequence of P^*. Let $k = \kappa(\ell)$ and set $v_\ell = v_k^*$. Because $|v_k^*| = A_1(k)$, the advisor advice bound is obvious.

The advisor $adv(\mathcal{A})$ takes input $\langle 1^\ell, r_1, v_\ell \rangle$. The advisor computes $k = \kappa(\ell)$ and simulates $pre(P^*)(1^k, r_1, v_k^*)$ to obtain output $v^{*\prime}$, and itself outputs $v' = v^{*\prime}$. The time of $adv(\mathcal{A})$ is the time $O(T_1(k))$ to run the simulation plus the time $O(\ell)$ to compute k. The auditor advice bound is obvious because $|v^{*\prime}| = A_2(k)$.

The auditor $aud(\mathcal{A})$ takes input $\langle v', x, y, z, r_2 \rangle$ where $|x| = |y| = \ell$. The proof input has the form $z \in \{0,1\}^* \# \{0,1\}^*$. We assume for simplicity that the nondeterministic input of P^* has the form $\eta \in \{0,1\}^*$. First, $aud(\mathcal{A})$ divides its random input r_2 into two parts, $r_2 = \rho \cdot r_2^*$, and divides its proof input z into two parts, $z = \eta \# z'$. It then computes $k = \kappa(\ell)$, so $v' = v^{*\prime}$. It then simulates $pro(P^*)$ on input $\langle v^{*\prime}, 1^k, r_2^*, \eta, x, \rho \rangle$ (see Figure 4). Let a' be a constant such that if $pro(P^*)$ runs for more than $a'k^e$ steps then V will conclude that $pro(P^*)$ took more than ak^e steps, and therefore reject. (If the rates of the clocks of P^* and V are within a factor of γ, then take $a' = \gamma a$.) If $pro(P^*)$ runs for more than $a'k^e$ steps, or if $check(r_1, r_2^*, \eta, x, \rho) = 0$ (P^* does not accept η), or if the output of $pro(P^*)$ does not have the form t, E_i, α_i, β_i for $i = 1,2$, and π_i for $i = 1,2,3$, then $aud(\mathcal{A})$ outputs 0 and halts. Having obtained t, E_i, α_i, and β_i within time $a'k^e$, the auditor $aud(\mathcal{A})$ proceeds as follows. For $i = 1,2$: $aud(\mathcal{A})$ views the z' part of its proof input as a witness $(D, w(E_i, D))$ that D is a decryption key for E_i. It checks these two witnesses. If these witnesses are both invalid, then $aud(\mathcal{A})$ outputs 0. If the witness $(D, w(E_j, D))$ is valid (say $j = 1$ if both are valid), $aud(\mathcal{A})$ then tests whether $y = D(\beta_j)$, and outputs 1 if true or 0 if not.

[11]The definition of $\mathcal{A}$ is important only when $\ell = \lceil k^d \rceil$ for some $k \in K$. For completeness, we give default values and computations when ℓ is not of this form.

We show that the time and proof length bounds stated in the theorem hold for $aud(\mathcal{A})$. The time of $aud(\mathcal{A})$ is of the order of the time to simulate $pro(P^*)$ (namely, $O(k^e)$), plus the time to check a witness $(D, w(E_i, D))$ that D decrypts E_i (namely, $O(k^{c_2})$), plus the time to decrypt a string β_j of length ℓ and compute $\kappa(\ell)$ (namely, $O(\ell k^{c_1})$). The proof length bound holds by definition of c_3; $|z| = |\eta| + |z'| + 1 \leq N(k) + k^{c_3} + 1$.

It remains to prove a bound on the correctness probability of $\mathcal{A}$ when $k \in K$. Fix an arbitrary $k \in K$. By definition of $adv(\mathcal{A})$, the advice string $v^{*\prime}$ was produced by $pre(P^*)$ on input $\langle 1^k, r_1, v_k^* \rangle$. Recall that $success(r_1, r_2^*, \eta, x, \rho)$ is the statement that $t \notin L$ and V accepts when the inputs to P^* are $\langle 1^k, r_1, r_2^*, \eta, x, \rho \rangle$, and that $check(r_1, r_2^*, \eta, x, \rho)$ is the check output of $pro(P^*)$. By assumption, with probability at least $\varepsilon(k)$ over r_1, r_2^*, x, ρ (which is the same as random choice over r_1, r_2, x):

$$\exists \eta \, (check(r_1, r_2^*, \eta, x, \rho) = 1) \;\wedge\; \forall \eta \, (check(r_1, r_2^*, \eta, x, \rho) = 1 \;\Rightarrow\; success(r_1, r_2^*, \eta, x, \rho)).$$

By definition of V, $success(r_1, r_2^*, \eta, x, \rho)$ is equivalent to:

S1. $pro(P^*)$ on input $\langle v^{*\prime}, 1^k, r_2^*, \eta, x, \rho \rangle$ uses time at most $a'k^e$ and produces outputs t, E_i, α_i, β_i for $i = 1, 2$, and π_i for $i = 1, 2, 3$;

S2. $t \notin L$;

S3. $\beta_i = g(E_i, \alpha_i)$ for $i = 1, 2$;

S4. V accepts the three zaps "proving" that

1. $E_1 \in \mathcal{G}(k) \vee E_2 \in \mathcal{G}(k)$;
2. $t \in L \vee \alpha_1 \in E_1(x)$;
3. $t \in L \vee \alpha_2 \in E_2(x)$.

Let $truth(r_1, r_2^*, \eta, x, \rho)$ be identical to S1–S4 except that S4 is changed to:

S4$'$. The following three statements are true:

1. $E_1 \in \mathcal{G}(k) \vee E_2 \in \mathcal{G}(k)$;
2. $t \in L \vee \alpha_1 \in E_1(x)$;
3. $t \in L \vee \alpha_2 \in E_2(x)$.

Recall the NP languages L_1 and L_2 defined following the description of ZEBRA. Recall that $\rho = \rho_1 \rho_2$ where ρ_i is the random string from the verifier used in the zap for L_i, and that we are using zaps with soundness error $\delta \stackrel{\text{def}}{=} 4^{-k}$ for each of L_1 and L_2. So, for $i = 1, 2$: with probability at least $1 - \delta$ over the random choice of ρ_i, the string ρ_i is "good" meaning that there does not exist a $\tau_i \notin L_i$ and a π_i such that V accepts $\langle \tau_i, \rho_i, \pi_i \rangle$. Because $r_2 = \rho \cdot r_2^* = \rho_1 \cdot \rho_2 \cdot r_2^*$, with probability at least $\varepsilon(k) - 2\delta$ over r_1, r_2, x, both ρ_1 and ρ_2 are good and V accepts the zaps S4(1), S4(2), and S4(3); in this case S4(1), S4(2), and S4(3) are true. Therefore, with probability at least $\varepsilon(k) - 2\delta$ over r_1, r_2, x:

$$\exists \eta \, (check(r_1, r_2^*, \eta, x, \rho) = 1) \;\wedge\; \forall \eta \, (check(r_1, r_2^*, \eta, x, \rho) = 1 \;\Rightarrow\; truth(r_1, r_2^*, \eta, x, \rho)). \tag{6}$$

Fix an arbitrary r_1, r_2, x such that (6) holds. We know that such an r_1, r_2, x occurs with probability at least $\varepsilon(k) - 2 \cdot 4^{-k}$ over random choice of r_1, r_2, x. We show that

$$\forall y\, (\, \exists z\, (aud(\mathcal{A})(x, y, z, r_2, adv(\mathcal{A})(1^\ell, r_1, v_\ell)) = 1) \;\Leftrightarrow\; y = f(x)\,).$$

Let y be arbitrary.

($\Leftarrow$) Let $y = f(x)$. By (6), there exists a η_0 such that $check(r_1, r_2^*, \eta_0, x, \rho) = 1$ and there holds $truth(r_1, r_2^*, \eta_0, x, \rho)$. In particular, by S1, $aud(\mathcal{A})$ will not stop and output 0 because $pro(P^*)$ runs too long or produces output of the wrong form. By S4′(1), we know that there is a j such that $E_j \in \mathcal{G}(k)$ (take $j = 1$ if this holds for $j = 1, 2$). For this j, there is some witness $(D, w(E_j, D))$ that D decrypts E_j. Combining $t \notin L$ (from S2) with either S4′(2) or S4′(3), it follows that $\alpha_j \in E_j(x)$. So $g(E_j, \alpha_j) \in E_j(f(x))$ by a property of g. From S3, $\beta_j = g(E_j, \alpha_j)$. Therefore,

$$D(\beta_j) = D(g(E_j, \alpha_j)) = f(x). \tag{7}$$

So for $z = \eta_0 \# binary(D, w(E_j, D))$, where $binary(D, w(E_j, D))$ is a binary encoded representation of $(D, w(E_j, D))$, the test $y = D(\beta_j)$ is true and $aud(\mathcal{A})$ outputs 1.

($\Rightarrow$) The proof is similar to the previous direction. Let $z = \eta \# binary(D, w(E, D))$ be such that $aud(\mathcal{A})$ outputs 1. By definition of $aud(\mathcal{A})$, $check(r_1, r_2^*, \eta, x, \rho) = 1$. So by (6), $truth(r_1, r_2^*, \eta, x, \rho)$ holds. Again by definition of $aud(\mathcal{A})$, it must be that D decrypts E_j for $j = 1$ or $j = 2$, and $y = D(\beta_j)$. Because $truth(r_1, r_2^*, \eta, x, \rho)$ holds, we again have (7). So $y = D(\beta_j) = f(x)$.

Therefore, $\mathcal{A}$ satisfies the correctness property (4) of an i.o. proof auditor for f on (the infinite number of) input lengths $\ell = k^d$ such that $k \in K$. ■

Remark 6.1 In the advice-bounded case, if the timeliness test is omitted from V (see Remark 3.1(1)), then the auditor time bound in Theorem 6.1 is also omitted.

7 Applications of the Connection

Theorem 6.1 is now used to show the existence of 2-round resource-bounded zero-knowledge proof systems for any language in NP, assuming the existence of a "moderately hard" function f, where "moderately" means that f can be computed by polynomial-size circuits, and "hard" means that f does not have an i.o. proof auditor meeting certain time, advice, proof length, and correctness bounds. We can give a variety of results with different assumptions about f and $\mathcal{E}$, leading to different conclusions about 2-round resource-bounded zero-knowledge for any language in NP.

Remark 7.1 All theorems in this section (except Theorem 7.6) assume the existence of a zap satisfying the definition in Section 2 and a semantically secure PPKC satisfying a malleability property. Using current technology, zaps can be based on non-uniform one-way permutations (for arbitrarily powerful provers) or trapdoor permutations (for efficient provers). For theorems that assume a semantically secure XOR-malleable PPKC, we can use the Goldwasser-Micali scheme, so we can replace this assumption by the Quadratic Residuousity Assumption (giving a more concrete but less general result).

7.1 Application to Time-Bounded Provers

We begin with an extreme example, where we make strong assumptions about f and $\mathcal{E}$ and obtain a correspondingly strong conclusion. Note that a correctness bound of $2^{-\ell}$ is easy for an i.o. proof auditor to achieve, because it is sufficient that f be efficiently computable on a single x of length ℓ for infinitely many ℓ. In the theorem, this correctness bound is increased only to $2^{-\ell^r}$ for arbitrary $r < 1$. In this theorem we make a strong assumption about the nonexistence of an i.o. proof auditor for f, namely, where no advisor advice and time bounds are assumed, but we obtain a correspondingly strong conclusion where the cheating prover P^* can use arbitrary precomputation time and advice.

Theorem 7.1 *Assume that:*

A1. $\mathcal{E}$ is semantically secure and completely malleable,

A2. there are constants c, c' such that f has $O(\ell^c)$-size circuits constructible in time $O(\ell^{c'})$,

A3. m is a constant such that, for all $0 < r < 1$, there is no i.o. proof auditor for f with auditor advice $\mathrm{poly}(\ell)$*, auditor time $O(\ell^m)$, proof length* $\mathrm{poly}(\ell)$*, and correctness $\geq 2^{-\ell^r}$.*

A4. the language L belongs to NP,

A5. letting b, b' be the constants of Lemma 5.1 and c_1, c_2 be the constants of Theorem 6.1, choose the constants d, e in ZEBRA *such that*

$$d > 1, \quad d \geq \max\left(\frac{b}{m-1}, \frac{c_1}{m-1}, \frac{c_2}{m}\right), \quad \textit{and} \quad e = d + b.$$

Then using zaps, the (P, V) described in ZEBRA *is a zero-knowledge proof system for L, where the prover P has*

$$\begin{aligned} \textit{precomputation advice} &= 0 \\ \textit{precomputation time} &= O(k^{c'd} + k^{cd+b'}) \\ \textit{protocol advice} &= O(k^{d+1}) \\ \textit{protocol time} &= O(k^e), \end{aligned}$$

and the verifier V has soundness error 2^{-k} against any class $\mathcal{P}$ of cheating provers whose protocol advice and nondeterminism bounds are $\mathrm{poly}(k)$*.*

Proof. The time and advice bounds on P are immediate from Lemma 5.1 and $e = d + b$.

Assume for contradiction that V does not have soundness error 2^{-k} against the class of cheating provers whose protocol advice and nondeterminism bounds are $\mathrm{poly}(k)$. This means that there is a P^* in this class and an infinite set K of numbers such that, for all $k \in K$, the cheating probability of (P^*, V) on k is greater than 2^{-k}. Using Theorem 6.1, we show that this implies the existence of an i.o. proof auditor for f having auditor advice and proof length bounds $\mathrm{poly}(\ell)$, auditor time bound $O(\ell^m)$, and correctness bound $2^{-\ell^r}$ for some $0 < r < 1$. This is a contradiction to the assumption A3 that f does not have an i.o. proof auditor with these bounds. It is obvious from

Theorem 6.1 that the auditor advice and proof length bounds are poly(ℓ). Using $k = \ell^{1/d}$ and the choice of d and $e = d + b$ in A5, the time of the auditor is O of

$$k^e + \ell k^{c_1} + k^{c_2} = \ell^{1+b/d} + \ell^{1+c_1/d} + \ell^{c_2/d} \leq 3\ell^m. \tag{8}$$

The correctness bound is at least

$$2^{-k} - 2 \cdot 4^{-k} = 2^{-\ell^{1/d}} - 2 \cdot 4^{-k} \geq 2^{-\ell^r} \tag{9}$$

for some r with $1/d < r < 1$. ■

The first of our main results described in the Introduction is an immediate corollary.

Theorem 7.2 *Under the assumptions A1, A2, A3 of Theorem 7.1 on $\mathcal{E}$ and f, and using zaps, every language in NP has a 2-round timing-based zero-knowledge proof system (P, V) where the precomputation advice bound of P is zero and V has soundness error 2^{-k} against any class of cheating provers whose protocol advice and nondeterminism bounds are* poly(k).

The next theorem states several variations of Theorem 7.1.

Theorem 7.3 *Theorem 7.1 remains true under any combination of the following modifications.*

- *M1. In assumption A1, replace "completely malleable" by "XOR-malleable", and change A2 to: "there are constants c, c' such that f has $O(\ell^c)$-size XOR-circuits constructible in time $O(\ell^{c'})$".*
- *M2. Remove from A2 the phrase "constructible in time $O(\ell^{c'})$", change the precomputation advice of P to $O(k^{cd} \log k)$, and change the precomputation time of P to $O(k^{cd+b'})$.*
- *M3. In A3 replace the* poly(ℓ) *bound on the auditor advice with a specific function $A(\ell)$, and further restrict the class $\mathcal{P}$ of provers in the soundness statement to those that have protocol advice bound $A(k^d)$.*
- *M4. In A3 remove the auditor time bound $O(\ell^m)$, and in A5 remove the constraint $d \geq$* $\max(b/(m-1), c_1/(m-1), c_2/m)$.
- *M5. In A3 change the phrase "for all $0 < r < 1$" to "there exists an r with $0 < r < 1$ such that", and in A5 change $d > 1$ to $d > 1/r$.*
- *M6. In A3 add a specific advisor advice bound $A_1(\ell)$ and/or advisor time bound $O(T_1(\ell))$ on the i.o. proof auditor, and further restrict the class $\mathcal{P}$ of provers in the soundness statement to those that have precomputation advice bound $A_1(k^d)$ and/or precomputation time bound $O(T_1(k^d))$.*

Proof. In each case, the proof is by a simple modification to the proof of Theorem 7.1. M1 is obvious because the circuits computing f now contain only XOR-gates. M2 is immediate from Lemma 5.1. For M3, a protocol advice bound of $A(k^d)$ yields via Theorem 6.1 an auditor advice bound of $A(\ell)$, contradicting that f has no i.o. proof auditor with this bound, among other bounds.

A simple way to see M4 is to take m large enough that $d \geq \max(b/(m-1), c_1/(m-1), c_2/m)$ is implied by $d > 1$.[12] For M5, the constraint $d > 1/r$ is used in (9). M6 is obvious. ■

The modification M3 is interesting only if $A(k^d)$ is at least as large as the protocol advice bound $O(k^{d+1})$ of the good prover P. For example, if $A(\ell) = \ell^r$ for some constant r, then $r \geq (d+1)/d$ ensures that $A(k^d) = k^{rd} \geq k^{d+1}$.

The second main result described in the Introduction is an immediate corollary of Theorem 7.1 with modification M1.

Theorem 7.4 *Assume that there are constants c, c' such that f has $O(\ell^c)$-size XOR-circuits constructible in time $O(\ell^{c'})$ and that assumption A3 of Theorem 7.1 holds. Then using zaps and a semantically secure XOR-malleable PPKC, every language in NP has a 2-round timing-based zero-knowledge proof system (P, V) where the precomputation advice bound of P is zero and V has soundness error 2^{-k} against the class of cheating provers whose protocol advice and nondeterminism bounds are* $\mathrm{poly}(k)$.

Remark 7.2 It seems possible to broaden the class of functions f to include those which have a small number of AND gates. The prover would then have to send additional information to help the verifier check the AND computations.

Using modification M2, both Theorem 7.2 and 7.4 have a version where the phrases "constructible in time $O(\ell^{c'})$" and "the precomputation advice bound of P is zero and" are removed.

7.2 Application to Advice-Bounded Provers

In this section we combine a modification of Theorem 7.1 (Corollary 7.7) with a result on the nonexistence of i.o. proof auditors for certain functions f (Theorem 7.6) to obtain the third result (Theorem 7.8) discussed in the Introduction: the only limitation on the i.o. proof auditors in Theorem 7.6 is that the auditor advice be sufficiently small; we place no bound on the advisor time or advice nor on the auditor time and proof length, and the auditor may have arbitrary amounts of "scratch space." As a result, the verifier in Theorem 7.8 is sound against any class of cheating provers having a sufficiently small bound on their protocol advice, even though their precomputation time and advice, protocol time, scratch space, and nondeterminism are not bounded. Because the protocol time of the cheating prover is not bounded, soundness holds even if V does not perform the timing test 3(a) in ZEBRA.[13] However, the good prover P in Theorem 7.8 does have bounds on its precomputation and protocol advice as well as $\mathrm{poly}(k)$ bounds on its precomputation and protocol time. Specifically, if $\ell = k^d$, then the precomputation advice of P is $\ell^2 = k^{2d}$ and its protocol advice is $O(\ell k) = O(k^{d+1})$. The bound imposed on the (binary) protocol advice of cheating provers in Theorem 7.8 is any function bounded above by δk^{2d} where $\delta < 1$. Note that

[12] Another way is to note that the $d \geq \max(\ldots)$ constraint is used only in bounding the time of the auditor in (8), and the bound (8) is no longer needed because the auditor time bound has been removed from A3.

[13] A conceptual simplification that can be made when the time of $pre(P^*)$ is unbounded is that $pre(P^*)$ does not use randomization. Given input $\langle 1^k, v_k \rangle$, $pre(P^*)$ finds a string v' that maximizes the probability that $t \notin L$ and V accepts when interacting with $pro(P^*)(1^k, r_2, v')$, where the probability is over r_2 and the random choices of V.

this bound is significantly larger than P's protocol advice bound $O(k^{d+1})$ (for large enough d) and it is only a little smaller than P's *precomputation* advice bound k^{2d}.

By bounding the protocol advice, we are essentially bounding the size of the protocol device. As a specific example, suppose that we model devices as circuits. Then we let the protocol part of P^* be a (uniform) "circuit simulator" that views its advice input as the description of a circuit C and then simulates the computation of C on the remaining inputs. Thus, if we show that V has a certain soundness guarantee against the circuit simulator with advice bound $A_2(k)$, then V has this soundness guarantee against any cheating prover whose protocol part is a circuit that can be described (to the circuit simulator) as a string of length $A_2(k)$; for example, using a standard way of describing circuits, V has the soundness guarantee against any cheating prover whose protocol part is a circuit with $c \cdot A_2(k)/\log A_2(k)$ gates where c is a specific constant. Generally in this section, we assume that the device model has a *universal device* $\mathcal{U}$, such that for each device $\mathcal{M}(x,y,z,r,v'')$ (where we view v'' as a binary advice input to $\mathcal{M}$) and for each $v'' \in \{0,1\}^*$ there is a string $v'(\mathcal{M},v'') \in \{0,1\}^*$ with $|v'(\mathcal{M},v'')| \leq |v''| + c_{\mathcal{M}}$ such that $\mathcal{U}(x,y,z,r,v'(\mathcal{M},v'')) = \mathcal{M}(x,y,z,r,v'')$ for all x,y,z,r,v'', where $c_{\mathcal{M}}$ is a constant depending only on $\mathcal{M}$. For both the Turing machine and the random-access models, there is a universal device having these properties.

Theorem 7.8 is nonconstructive in that we do not describe a specific function f to use. However, we show that, for each k and $\ell = k^d$, if f_ℓ (f restricted to $\{0,1\}^\ell$) is chosen at random from a certain specific class of functions then with high probability the chosen f_ℓ will yield a protocol with soundness error 2^{-k}.

In this section we restrict attention to functions f such that, for all $\ell \geq 1$, f_ℓ is a linear function from the vector space $\mathrm{GF}(2)^\ell$ to itself. Let $\mathcal{LIN}_\ell$ denote the class of linear functions $f : \mathrm{GF}(2)^\ell \to \mathrm{GF}(2)^\ell$, and let $\mathcal{LIN}$ denote the class of functions f such that $f_\ell \in \mathcal{LIN}_\ell$ for all $\ell \geq 1$. There is a one-to-one correspondence between the $f_\ell \in \mathcal{LIN}_\ell$ and the $\ell \times \ell$ matrices F over $\mathrm{GF}(2)$, given by $f_\ell(x) = F_{f_\ell} x$ for all $x \in \mathrm{GF}(2)^\ell$. Therefore, $\mathcal{LIN}_\ell$ contains exactly 2^{ℓ^2} functions. Note also that if $f \in \mathcal{LIN}$ then f has ℓ^2-size XOR-circuits, and a ℓ^2-size circuit for f_ℓ is described by the matrix F_{f_ℓ}, *i.e.*, a binary string of length ℓ^2.

We next state the main technical result of this section (Lemma 7.5). Roughly, it states that if $\delta, \theta > 0$ are sufficiently small constants, then "most" functions in $\mathcal{LIN}_\ell$ have no proof auditor with auditor advice bound $\delta\ell^2$ and correctness bound $2^{-\theta\ell}$. Both bounds are close to the best possible: if the advice bound is ℓ^2, then v' can be the matrix F_{f_ℓ}, and $\mathcal{U}$ can test in time $O(\ell^2)$ whether $y = f_\ell(x)$; if the correctness bound is $2^{-\ell}$ then the $\mathcal{U}$ that outputs 1 iff $x = y = 0^\ell$ satisfies this bound (it is correct for at least one x). This result might be expected: because $\mathcal{LIN}_\ell$ contains 2^{ℓ^2} functions, an obvious counting argument shows that with $\delta\ell^2$ ($\delta < 1$) bits we can describe only a vanishingly small fraction of the functions in $\mathcal{LIN}_\ell$. Our counting argument needs to be more involved because a proof auditor for f does not have to correctly check for *all* x that z is a proof that $y = f(x)$; it suffices that it be correct on some set X_f of x's, where X_f contains a small ($2^{-\theta\ell}$) fraction of all the x's of length ℓ. In particular, a single advice string can give a proof auditor that is correct for many functions f. Another complication is that there can be a different X_f for each f.

Lemma 7.5 *Let $\delta, \theta > 0$. Let $\chi(\ell) = 2^{-\theta\ell+1}$ and $\lambda(\ell) = 1 - 2^{-\ell^2(1-\delta-3\theta-6\ell^{-1})}$. Let $\mathcal{U}$ be a deterministic device with inputs x, y, z, r, v' where $|x| = |y|$, $v' \in \{0,1\}^*$, and $|v'| \leq \delta\ell^2 - 1$, where $\ell = |x|$.*

Let $\ell \geq 4$. Choose f_ℓ at random from $\mathcal{LIN}_\ell$. Then with probability at least $\lambda(\ell)$ (over the choice of f_ℓ), "for no v' is $\mathcal{U}$ a proof auditor for f_ℓ with correctness bound $\chi(\ell)$"; precisely (cf. (4)), for all v' with $|v'| \leq \delta\ell^2 - 1$,

$$\Pr_{x,r}\left[\forall y \left(\exists z \left(\mathcal{U}(x,y,z,r,v') = 1\right) \Leftrightarrow y = f_\ell(x)\right)\right] < \chi(\ell). \tag{10}$$

Before proving this lemma, we apply it to 2-round zero-knowledge where cheating provers have limited protocol advice. First we use it to show the existence of a "moderately hard" function in $\mathcal{LIN}$.

Theorem 7.6 *Let $\delta, \theta > 0$ be constants with $\delta + 3\theta < 1$. There exists $f \in \mathcal{LIN}$ such that f has no i.o. proof auditor with auditor advice bound $\delta\ell^2$ and correctness bound $2^{-\theta\ell}$.*

Proof. Choose $\tilde{\delta}, \tilde{\theta} > 0$ such that $\delta < \tilde{\delta}$, $\theta < \tilde{\theta}$, and $\tilde{\delta} + 3\tilde{\theta} < 1$. Then there is an ℓ_0 such that $\lambda(\ell) = 1 - 2^{-\ell^2(1-\tilde{\delta}-3\tilde{\theta}-6\ell^{-1})} > 0$ for all $\ell \geq \ell_0$. Let $\mathcal{U}$ be a universal device such that, for every device $\mathcal{M}(x,y,z,r,v'')$, there is a binary string $v'(\mathcal{M}, v'')$ of length at most $|v''| + c_{\mathcal{M}}$ such that $\mathcal{M}(x,y,z,r,v'') = \mathcal{U}(x,y,z,r,v'(\mathcal{M},v''))$ for all x,y,z,r,v''. By Lemma 7.5, for each $\ell \geq \ell_0$ there is a $f_\ell \in \mathcal{LIN}_\ell$ such that for no v' with $|v'| \leq \tilde{\delta}\ell^2 - 1$ is $\mathcal{U}$ a proof auditor for f_ℓ with correctness bound $\chi(\ell) = 2^{-\tilde{\theta}\ell+1}$. This defines $f \in \mathcal{LIN}$ (define f_ℓ arbitrarily for $\ell < \ell_0$).

Assume for contradiction that there is an i.o. proof auditor $\mathcal{A}$ for f with auditor advice bound $\delta\ell^2$ and correctness bound $2^{-\theta\ell}$. Let $\mathcal{M} = aud(\mathcal{A})$. As noted above in footnote 13, because the time of $adv(\mathcal{A})$ is not bounded, we can assume without loss of generality that, for each ℓ, $adv(\mathcal{A})$ deterministically outputs some string v'' of length $\delta\ell^2$. Now for all sufficiently large ℓ,

$$|v'(\mathcal{M}, v'')| \;\leq\; |v''| + c_{\mathcal{M}} \leq \delta\ell^2 + c_{\mathcal{M}} \leq \tilde{\delta}\ell^2 - 1 \tag{11}$$

$$2^{-\theta\ell} \;\geq\; 2^{-\tilde{\theta}\ell+1} = \chi(\ell). \tag{12}$$

Thus, there are infinitely many ℓ such that, for some v', $|v'| \leq \tilde{\delta}\ell^2 - 1$ and $\mathcal{U}(x,y,z,r,v')$ is a proof auditor for f_ℓ with correctness bound $\chi(\ell)$; this is a contradiction. ■

Next we state the relevant variation of Theorem 7.1, obtained by applying modifications M1 (with $c = 2$), M2, M3 (with $A(\ell) = \delta\ell^2$), and M4 of Theorem 7.3. We also remove the poly(ℓ) bound on the proof length of the i.o. proof auditor in A3 (this just gives a stronger assumption) to emphasize that there is no proof length bound on the i.o. proof auditor in Theorem 7.6. Also, we save a factor of $\log k$ in the precomputation advice bound in M2 because, as we have noted above, an XOR-circuit of size ℓ^2 for $f_\ell \in \mathcal{LIN}_\ell$ can be described by a binary string of length $\ell^2 = k^{2d}$.

Corollary 7.7 *Assume that:*

A1′. $\mathcal{E}$ is semantically secure and XOR-malleable,

A2′. $f \in \mathcal{LIN}$,

A3′. there is a $\delta > 0$ such that, for all $0 < r < 1$, there is no i.o. proof auditor for f with auditor advice $\delta\ell^2$ and correctness $\geq 2^{-\ell^r}$,

A4′. the language L belongs to NP,

A5′. letting b, b' be the constants of Lemma 5.1, choose $d > 1$ and $e = d + b$.

Then using zaps, the (P, V) described in ZEBRA *is a zero-knowledge proof system for L where: the prover P has precomputation advice k^{2d}, precomputation time $O(k^{2d+b'})$, protocol advice $O(k^{d+1})$, and protocol time $O(k^e)$; and letting $\mathcal{P}$ be the class of cheating provers having protocol advice bound δk^{2d}, the verifier V (even without the test 3(a)) has soundness error 2^{-k} for L against the class $\mathcal{P}$ for all sufficiently large k.*

The next main result follows easily from Theorem 7.6 and Corollary 7.7.

Theorem 7.8 *Let L belong to NP, $d > 1$, and $\delta < 1$. Then using zaps and a semantically secure XOR-malleable PPKC, there is a 2-round zero-knowledge proof system (P, V) for L where: (i) the prover P has protocol advice bound $O(k^{d+1})$; and (ii) letting $\mathcal{P}$ be the class of cheating provers having protocol advice bound δk^{2d}, the verifier V (even without the test 3(a)) has soundness error 2^{-k} for L against the class $\mathcal{P}$ for all sufficiently large k.*

Moreover, letting b, b' be the constants of Lemma 5.1:

1. *soundness is not based on any cryptographic assumptions;*
2. *P has precomputation advice k^{2d}, precomputation time $O(k^{2d+b'})$, and protocol time $O(k^{d+b})$;*
3. *letting $n = k^{2d}$ and $0 < \mu < 1 - \delta$, the soundness error (ii) holds for all but a $2^{-\mu n}$ fraction of the binary precomputation advice strings v_k of length n.*

Proof. Given $\delta < 1$, choose $\theta > 0$ such that $\delta + 3\theta < 1$. The function f of Theorem 7.6 satisfies assumptions A2′ and A3′ of Corollary 7.7, which proves the theorem except for item 3. The proof of 3 follows similarly from Lemma 7.5, choosing $\theta > 0$ such that $\delta + 3\theta < 1 - \mu$ (possible because $\mu < 1 - \delta$). ■

Remark 7.3 Theorem 7.8 is stated in terms of "all sufficiently large k" for simplicity. But given a particular δ such that we want δk^{2d} to be the bound on the cheating prover's advice, and given the probability $p < 1$ with which we want the random choice of f to imply soundness error 2^{-k}, the inequalities $\lambda(\ell) \geq p$, (11), and (12) give a specific lower bound on ℓ, which gives a specific lower bound on $k = \lfloor \ell^{1/d} \rfloor$. The choice of d involves a trade-off. On the one hand, a larger d gives a larger gap between the protocol advice bound $O(k^{d+1})$ of P and the limitation δk^{2d} on the protocol advice of P^*. On the other hand, choosing larger d increases the time and advice bounds of the good prover P.

Remark 7.4 Recall that we have assumed $|E(0)| = |E(1)| = k$ for simplicity. This assumption affects only the protocol advice bound $O(k^{d+1})$ of the good prover. If $|E(0)| = |E(1)| = k^c$, then this bound becomes $O(k^{d+c})$. Because c depends only on $\mathcal{E}$ and d can be chosen arbitrarily large, we can still choose d such that the protocol advice $O(k^{d+c})$ of the good prover is significantly less than the limitation δk^{2d} on the protocol advice of a cheating prover.

We now prove Lemma 7.5. Before getting into technical details, we give a high-level outline of the proof. Roughly speaking, we want to show that most functions $f \in \mathcal{LIN}_\ell$ do not have a proof auditor $\mathcal{U}(x, y, z, r, v')$ where v' is a "short" string. In the proof we will not actually use that v' is short, but use only that the number of "short" strings is much smaller than $|\mathcal{LIN}_\ell| = 2^{\ell^2}$. Recall that $\mathcal{U}(x, y, z, r, v')$ is a proof auditor for f if, with sufficiently large probability χ over the random choice of x and r: $\forall y\,(\,\exists z\,(\mathcal{U}(x, y, z, r, v') = 1) \;\Leftrightarrow\; y = f(x)\,)$. Defining $\mathcal{U}'(x, y, r, v') = 1$ iff $\exists z(\mathcal{U}(x, y, z, r, v') = 1)$, this condition can be written

$$\forall y\,(\,\mathcal{U}'(x, y, r, v') = 1 \;\Leftrightarrow\; y = f(x)\,). \tag{13}$$

Say first that we want (13) to hold with probability 1, that is, for all x and r. In this case it is easy to see that for each v' there is at most one f such that (13) holds. (This is the "standard" counting argument.)

Say now that we still want (13) to hold for all r, but now only with probability χ over random x. Now it can happen for a fixed v' that several f's satisfy this. Say that v' *handles* f if v' and f satisfy (13) with probability $\geq \chi$ over random x. For each v', we must upper bound the number of f's that v' can handle. It is clear that if v' handles f, then there is a "somewhat large" set X_f (precisely, $|X_f|$ contains at least a fraction χ of the x's in $\mathrm{GF}(2)^\ell$) such that (13) holds for all $x \in X_f$. As in the first case, v' determines $f(x)$ for all $x \in X_f$. Because f is linear, f is completely determined by its values on any ℓ linearly independent vectors in $\mathrm{GF}(2)^\ell$. We can find $c \approx \log |X_f|$ linearly independent vectors in X_f (Lemma 7.10). So the freedom we have in choosing f is limited to f's values on $\ell - c$ (linearly independent) vectors. A complication is that v' can handle many f's, each having a different X_f. To deal with this, a probabilistic argument (Lemma 7.9) is used to show that there is a sufficiently small collection $\mathcal{J} = \{J_1, \ldots, J_j\}$ such that each J_i is linearly independent and $|J_i| = c$, and if v' handles f then there is a $J \in \mathcal{J}$ with $J \subseteq X_f$. This multiplies the number of handled f's by $|\mathcal{J}|$.

Finally, if (13) must hold only with some probability p over random r, then a pigeonhole argument (Claim 7.3) shows that the number of f's handled by each v' is multiplied by $1/p$ at most.

The detailed proof follows. First we give two lemmas that are used in the proof.

Lemma 7.9 *Let $\mathcal{I}$ and $\mathcal{H}$ be sets with $m = |\mathcal{I}|$ and $n = |\mathcal{H}|$. For each $f \in \mathcal{H}$ let $S_f \subseteq \mathcal{I}$. Let s be such that $|S_f| \geq s$ for all $f \in \mathcal{H}$. Then there is a $\mathcal{J} \subseteq \mathcal{I}$ with $|\mathcal{J}| = \lceil (m/s) \log n \rceil$ such that $\mathcal{J} \cap S_f \neq \emptyset$ for all $f \in \mathcal{H}$.*

Proof. Let $\mathcal{J} \subseteq \mathcal{I}$ with $|\mathcal{J}| = j \stackrel{\text{def}}{=} \lceil (m/s) \log n \rceil$ be chosen at random. For each fixed $f \in \mathcal{H}$, the probability that $\mathcal{J} \cap S_f = \emptyset$ is at most

$$\binom{m-s}{j} \Big/ \binom{m}{j} = \prod_{i=0}^{j-1} \frac{m-s-i}{m-i} \leq \left(\frac{m-s}{m}\right)^j < \frac{1}{n}\,.$$

Therefore, $\Pr[\,(\exists\, f \in \mathcal{H})(\mathcal{J} \cap S_f = \emptyset)\,] < |\mathcal{H}|/n = 1$. ■

The second lemma gives bounds on the number of linearly independent subsets of a set $X \subseteq \mathrm{GF}(2)^\ell$. A set $I \subseteq \mathrm{GF}(2)^\ell$ is *linearly independent* if the vectors in I are linearly independent.

Lemma 7.10 *Let $X \subseteq \mathrm{GF}(2)^\ell$ and let c be a positive integer with $2^{c-1} < |X|$. Let $N(X,c)$ be the number of linearly independent sets I such that $I \subseteq X$ and $|I| = c$. Then*

$$\frac{(|X|/2)^{c-1}}{c\,!} \leq N(X,c) \leq \frac{|X|^c}{c\,!}\,.$$

Proof. The upper bound is obvious because it is an upper bound on the number of size-c subsets of X. To prove the lower bound, we show that the number of ordered linearly independent sets I, such that $I \subseteq X$ and $|I| = c$, is at least $(|X|/2)^{c-1}$; a lower bound on the number of unordered I is then obtained by dividing by $c!$. Consider the number of choices for $I = \{\iota_1, \ldots, \iota_c\}$. First, ι_1 can be any member of X except the zero-vector. Having chosen $\iota_1, \ldots, \iota_i$ for some $i \geq 1$, the choice of ι_{i+1} can be any vector not in the linear span of $\iota_1 \ldots, \iota_i$. Because the size of this span is 2^i, there are $|X| - 2^i$ choices for ι_{i+1}. Because $2^{c-1} < |X|$, this number of choices is positive if $i \leq c-1$. Therefore, the total number of choices for ordered I is at least $\prod_{i=0}^{c-1}(|X| - 2^i)$, and this is at least $(|X|/2)^{c-1}$ because $|X| - 2^i \geq |X|/2$ for $0 \leq i \leq c-2$ (this follows from $2^{c-1} < |X|$). ■

Proof of Lemma 7.5.
Let $\mathcal{U}$ be a device as in the statement of the lemma, and fix $\delta, \theta > 0$ and $\ell \geq 4$. Having fixed ℓ, abbreviate $\chi(\ell)$ and $\lambda(\ell)$ as χ and λ respectively. To further simplify the notation, we write f instead of f_ℓ because ℓ is fixed. Let $c = \ell - \lceil \theta\ell \rceil$, and note that $c \leq \ell - \theta\ell$ and $\ell - c \leq \theta\ell + 1$. Let $p = 2^{c-\ell}$, so $2p \leq 2^{-\theta\ell+1} = \chi$. Because $2p \leq \chi$, it suffices to prove the lemma with $2p$ in place of χ in (10).

Say that v' *handles* $f \in \mathcal{LIN}_\ell$ if $\mathcal{U}(\ldots, v')$ is an i.o. proof auditor for f with correctness bound $2p$, *i.e.* (cf. (4)),

$$\Pr_{x,r}[\forall y\,(\,\exists z\,(\mathcal{U}(x,y,z,r,v') = 1) \;\Leftrightarrow\; y = f(x)\,)\,] \geq 2p. \tag{14}$$

To prove Lemma 7.5, it suffices to prove the following:

Claim 7.1 *For each v', there are at most $2^{3\theta\ell^2+6\ell}$ functions $f \in \mathcal{LIN}_\ell$ such that v' handles f.*

Before proving this claim, we show how the lemma follows easily from it. The number of strings $v' \in \{0,1\}^*$ with $|v'| \leq \delta\ell^2 - 1$ is at most $2^{\delta\ell^2}$. There are 2^{ℓ^2} functions in $\mathcal{LIN}_\ell$. Therefore, the probability that a randomly chosen $f \in \mathcal{LIN}_\ell$ is handled by *some* v' is at most

$$2^{-\ell^2} \cdot 2^{\delta\ell^2} \cdot 2^{3\theta\ell^2+6\ell} = 2^{-\ell^2(1-\delta-3\theta-6\ell^{-1})} = 1 - \lambda.$$

In the rest of the proof we prove Claim 7.1. Fix a v', and let $\mathcal{H}$ be the set of f that are handled by v'. Define the function $\mathcal{U}'(x,y,r)$ (where $|x| = |y| = \ell$) by

$$\mathcal{U}'(x,y,r) = 1 \quad \Leftrightarrow \quad \exists z\, \mathcal{U}(x,y,z,r,v') = 1.$$

So if $f \in \mathcal{H}$ then (cf. (14))

$$\Pr_{x,r}[\forall y\,(\mathcal{U}'(x,y,r) = 1 \;\Leftrightarrow\; y = f(x)\,)\,] \geq 2p. \tag{15}$$

Claim 7.2 *If $f \in \mathcal{H}$ then there is an $X_f \subseteq \mathrm{GF}(2)^\ell$ such that $|X_f| = 2^c$ and*

$$\forall x \in X_f, \quad \varphi(x, f) \stackrel{\text{def}}{=} \Pr_r[\forall y\, (\mathcal{U}'(x, y, r) = 1 \;\Leftrightarrow\; y = f(x)\,)\,] \geq p. \tag{16}$$

Proof. Note that the size of X_f is $2^c = 2^{c-\ell}2^\ell = p2^\ell$. If there is no X_f of this size satisfying (16), then for a randomly chosen $x \in \mathrm{GF}(2)^\ell$, the probability that $\varphi(x, f) < p$ is at least $1-p$. Therefore, the probability in (15) is less than $(1-p)p + p = 2p - p^2 < 2p$, and this contradicts the assumption that $f \in \mathcal{H}$. ■

For the rest of the proof, for each $f \in \mathcal{H}$ fix a particular X_f with $|X_f| = 2^c$ and (16).

For $x, y \in \mathrm{GF}(2)^\ell$, say that y *is possible for* x if there exists an $f \in \mathcal{H}$ such that $x \in X_f$ and $y = f(x)$.

Claim 7.3 *For each $x \in \mathrm{GF}(2)^\ell$ there are at most p^{-1} y's such that y is possible for x.*

Proof. Fix an x. By definition, if y is possible for x, then there exists an $f \in \mathcal{H}$ such that $x \in X_f$ and $y = f(x)$. Let $\mathcal{F}(x)$ be the set of all f such that $f \in \mathcal{H}$ and $x \in X_f$. For each such f, we have $x \in X_f$, and this implies that $\varphi(x, f) \geq p$ (see (16)). For each $f \in \mathcal{F}(x)$, let

$$R_f = \{\, r \mid \forall y\, (\mathcal{U}'(x, y, r) = 1 \;\Leftrightarrow\; y = f(x)\,)\,\}.$$

For each $f \in \mathcal{F}(x)$, because $\varphi(x, f) \geq p$, it follows that R_f contains at least a fraction p of the possible values for r.

We show that if $f(x) \neq f'(x)$ then $R_f \cap R_{f'} = \emptyset$. Suppose otherwise that $r_0 \in R_f \cap R_{f'}$. Let $y_0 = f(x)$. Then we would have the contradiction that $\mathcal{U}'(x, y_0, r_0) = 1$ (because $r_0 \in R_f$ and $y_0 = f(x)$) and $\mathcal{U}'(x, y_0, r_0) = 0$ (because $r_0 \in R_{f'}$ and $y_0 \neq f'(x)$).

Assume for contradiction that there are q different values $y_1, \ldots, y_q$ that are possible for x, where $q > p^{-1}$. Therefore, for each i $(1 \leq i \leq q)$ there is a $f_i \in \mathcal{F}(x)$ such that $f_i(x) = y_i$. By the argument above, the sets R_{f_i} for $1 \leq i \leq q$ are pairwise disjoint and each contains at least a fraction p of the r's; this contradiction proves Claim 7.3. ■

Recall that we want to prove $|\mathcal{H}| \leq 2^{3\theta\ell^2 + 6\ell}$. Let

$$\mathcal{I} = \{\, I \subseteq \mathrm{GF}(2)^\ell \mid I \text{ is linearly independent and } |I| = c\,\}.$$

Let $f \in \mathcal{H}$. Using $|X_f| = 2^c > 2^{c-1}$, it follows from Lemma 7.10 that there is an $I \in \mathcal{I}$ such that $I \subseteq X_f$. Assume for the moment that there is a single $I \in \mathcal{I}$ such that $I \subseteq X_f$ for all $f \in \mathcal{H}$. By adding $\ell - c$ more vectors, enlarge I to a set $I' \supseteq I$ such that I' is linearly independent and $|I'| = \ell$. By definition and $I \subseteq X_f$, if $x \in I$ and $y = f(x)$ for some $f \in \mathcal{H}$, then y is possible for x. By Claim 7.3, for each $x \in I$ there are at most $p^{-1} = 2^{\ell-c}$ possible values for $f(x)$. And for each $x \in I' - I$ there are at most 2^ℓ possible values for $f(x)$. Because $f \in \mathcal{LIN}_\ell$, I' is linearly independent and $|I'| = \ell$, f is completely determined by the ℓ values $f(\iota)$ for $\iota \in I'$. Therefore, the total number of possibilities for $f \in \mathcal{H}$ is at most

$$(2^{\ell-c})^{|I|}(2^\ell)^{|I'-I|} = (2^{\ell-c})^c(2^\ell)^{\ell-c} = 2^{(\ell+c)(\ell-c)} \leq 2^{(2\ell)(\theta\ell+1)} = 2^{2\theta\ell^2+2\ell}. \tag{17}$$

(Except for the factors of $\theta\ell^2$ and ℓ in the exponent, this bound has the same form as the bound in Claim 7.1.)

The final step is to remove the assumption that there is a single $I \in \mathcal{I}$ such that $I \subseteq X_f$ for all $f \in \mathcal{H}$. To do this we use Lemmas 7.9 and 7.10 to show that there is a small enough $\mathcal{J} \subseteq \mathcal{I}$ (so each $J \in \mathcal{J}$ is a linearly independent set of size c) such that for each $f \in \mathcal{H}$, there exists a $J \in \mathcal{J}$ such that $J \subseteq X_f$. Then by including the factor $|\mathcal{J}|$ in the calculation (17), we obtain an upper bound on $|\mathcal{H}|$.

To apply Lemma 7.9, let $\mathcal{I}$ and $\mathcal{H}$ be defined as above. For $f \in \mathcal{H}$, let $S_f = \{ I \in \mathcal{I} \mid I \subseteq X_f \}$, *i.e.*, S_f is the set of I such that I is linearly independent, $|I| = c$, and $I \subseteq X_f$. We must place bounds on $m = |\mathcal{I}|$, $n = |\mathcal{H}|$, and $s = \min\{ |S_f| \mid f \in \mathcal{H} \}$. Because each $I \in \mathcal{I}$ is a linearly independent subset of $\mathrm{GF}(2)^\ell$ (a set of size 2^ℓ) with $|I| = c$, Lemma 7.10 implies that $m \leq (2^\ell)^c/c!$. Because $\mathcal{H} \subseteq \mathcal{LIN}_\ell$, $n \leq 2^{\ell^2}$. By the definition of S_f and $|X_f| = 2^c$, Lemma 7.10 implies that $s \geq (2^c/2)^{c-1}/c!$. Therefore, by Lemma 7.9, there is a $\mathcal{J} \subseteq \mathcal{I}$ such that $\mathcal{J} \cap S_f \neq \emptyset$ for all $f \in \mathcal{H}$, and

$$|\mathcal{J}| = \left\lceil \frac{m}{s} \cdot \log n \right\rceil \leq \frac{(2^\ell)^c}{(2^{c-1})^{c-1}} \cdot \ell^2 \leq 2^{\ell c - (c-1)^2 + \ell}.$$

(In the last inequality we use $\ell^2 \leq 2^\ell$ for all $\ell \geq 4$.) Thus, for each $f \in \mathcal{H}$ there is a $J \in \mathcal{J} \cap S_f$; this means that J is linearly independent, $|J| = c$, and $J \subseteq X_f$. For each $f \in \mathcal{H}$, arbitrarily choose one J with these properties and denote it J_f. We now partition $\mathcal{H}$ according to the members of $\mathcal{J}$; *i.e.*, for each $J \in \mathcal{J}$ define $\mathcal{H}_J = \{ f \in \mathcal{H} \mid J_f = J \}$. Thus, for each J, for all $f \in \mathcal{H}_J$, we have $J \subseteq X_f$. By the calculation (17) and the argument preceding it, $|\mathcal{H}_J| \leq 2^{(\ell+c)(\ell-c)}$ for each $J \in \mathcal{J}$. Therefore, using $c \leq \ell$ and $\ell - c \leq \delta\ell^2 + 1$,

$$\begin{aligned} |\mathcal{H}| \leq |\mathcal{J}| \cdot 2^{(\ell+c)(\ell-c)} &\leq 2^{\ell c - (c-1)^2 + \ell + (\ell+c)(\ell-c)} \\ &\leq 2^{(\ell+2c)(\ell-c)+2c+\ell} \\ &\leq 2^{(3\ell)(\theta\ell+1)+3\ell} \\ &= 2^{3\theta\ell^2+6\ell}. \end{aligned}$$

This proves Claim 7.1 and Lemma 7.5. ■

8 Zero Knowledge

In this section we argue that Protocol ZEBRA is zero-knowledge. We remark that if $E_1 = E_2$ then the proof that E_1 or E_2 is in $\mathcal{G}(k)$ will not be witness-hiding, but this occurs only with vanishing probability and in the remainder of this section we assume this is not the case.

Let V' be a "cheating verifier" and let $t \in L$ be arbitrary. First, the simulator executes Step 0(a), choosing encryption keys E_1, E_2. On input x, ρ from V', the simulator

(a) computes $\alpha_i \in_R E_i(x)$, $i = 1, 2$, and $\beta_i = g(E_i, \alpha_i)$, $i = 1, 2$;

(b) prepares a zap for "$E_1 \in \mathcal{G}(k) \vee E_2 \in \mathcal{G}(k)$";

(c) with the random bits used in generating the α_i's as witness, prepares the zaps of "$t \in L \vee \alpha_i \in E_i(x)$", $i = 1, 2$.

These are assembled, together with t, into a simulated Round 2 message. Note that the simulator is neither time- nor advice-bounded (beyond the usual restriction to probabilistic polynomial time), and so computation of β_1 and β_2 is not a problem.

Intuitively, indistinguishability from real interactive proofs of $t \in L$ comes from the witness-indistinguishability of the zaps and semantic security of $\mathcal{E}$. Formally, we construct a hybrid argument with the sets of distributions listed below. In the sequel, let w be the witness to $t \in L$ used by the real prover in real executions. The first distribution is the set of real executions, and the last is the set of simulated executions. We show they are indistinguishable in nonuniform polynomial time. That is, they are indistinguishable even to a circuit with t and w "hard-wired" in.

Remark 8.1 Recall that, in order to make the good prover maximally efficient, for each of $i = 1, 2$, the zap of "$t \in L \vee \alpha_i \in E_i(x)$" is really ℓ zaps of statements of the form: "$t \in L \vee \alpha_{ij} \in E_i(x_j)$". For conceptual simplicity, we ignore this for the current discussion. However, nothing substantive changes: the complete hybrid argument simply has an additional $\ell-1$ steps between Distributions 1 and 2, and an additional $\ell - 1$ steps between Distributions 3 and 4.

1. Real executions: $\alpha_1 \in E_1(x^*)$, $\alpha_2 \in E_2(x^*)$, for $x^* \in_R \{0,1\}^\ell$, and w is the witness used in constructing both zaps in Step (c).

2. Executions as in Distribution 1 except now $\alpha_1 \in_R E_1(x)$, but the witness used in constructing the zap of "$t \in L \vee \alpha_1 \in E_1(x)$" is still the witness w for $t \in L$ (so in particular α_2 is as in real executions: an encryption of an unrelated, random string);

3. Executions as in Distribution 2, so $\alpha_1 \in_R E_1(x)$ and α_2 is as in real executions, but now the witness used in constructing the zap of "$t \in L \vee \alpha_1 \in E_1(x)$" is the string of random bits used in generating α_1;

4. Executions as in Distribution 3, so $\alpha_1 \in_R E_1(x)$ and the witness used in constructing the zap of "$t \in L \vee \alpha_1 \in E_1(x)$" is the string of random bits used in generating α_1, except that now $\alpha_2 \in_R E_2(x)$, although the witness used in constructing the zap of "$t \in L \vee \alpha_2 \in E_2(x)$" is still the witness w for $t \in L$;

5. Executions as in Distribution 4, in that $\alpha_1 \in_R E_1(x)$ and the witness used in constructing the zap of "$t \in L \vee \alpha_1 \in E_1(x)$" is the string of random bits used in generating α_1, and $\alpha_2 \in_R E_2(x)$, but now the witness used in constructing the zap of "$t \in L \vee \alpha_2 \in E_2(x)$" is the string of random bits used in constructing α_2. This is precisely the distribution produced by the simulator.

Nonuniform polynomial-time indistinguishability of Distributions 1 and 2 follows from nonuniform indistinguishability of encryptions (Definition 2.2).

Nonuniform polynomial-time indistinguishability of Distributions 2 and 3 follows from the witness-indistinguishability of the zap of "$t \in L \vee \alpha_1 \in E_1(x)$".

Nonuniform polynomial-time indistinguishability of Distributions 3 and 4 follows from the semantic security of the cryptosystem under chosen plaintext attack (as above).

Nonuniform polynomial-time indistinguishability of Distributions 4 and 5 follows from witness-indistinguishability of the zap of "$t \in L \vee \alpha_2 \in E_2(x)$".

Thus, the simulated executions are nonuniform polynomial (in k) time indistinguishable from real executions.

In the remainder of this section we discuss concurrent zero-knowledge. In concurrent zero-knowledge [12], the prover carries out N interactive proofs concurrently with N different verifiers, each for a different input $\langle t_j, w_j \rangle$, $1 \leq j \leq N$. The adversary who is trying to defeat zero-knowledge controls all N cheating verifiers (they can collude) and specifies the interleaving of messages.

In [12] no distinction is made between the case in which a single prover is running multiple copies of the proof system, and the case in which there are multiple provers, each running just one copy of the proof system. Indeed, if only the verifier needs to act in a timely fashion, these two cases are equivalent. However, if the prover is time bounded, then a too large N will prevent even a good prover from having enough time to quickly respond to each of the N verifiers' initial messages, if these are all scheduled simultaneously.

Our results hold for concurrent zero-knowledge, assuming that $N \leq k^c$ for some constant c that is known in advance (of course, c may be chosen, in advance, to accomodate the desired level of concurrency). There are four cases to be considered depending on whether the prover is advice-bounded or time-bounded and whether concurrent zero-knowledge is single-prover or multi-prover. Fortunately, there is much similarity among the cases. In particular, the modified zero-knowledge argument, which is outlined for the case of a single advice-bounded prover, is the same in the other cases.

Regarding soundness, the proof of the connection between cheating provers and proof auditors (Theorem 6.1) requires modification in the multiple-interaction case. The result is the same, except that the correctness probability of the proof auditor is divided by the factor N. So in our main theorems, the soundness error is multiplied by N. (As an alternative to modifying Theorem 6.1, it is enough to note that soundness composes as one expects: if a single-interaction protocol with soundness error $s(k)$ is run N times concurrently, then the probability that a cheating prover can cause any one of the N verifiers to accept in error is at most $N \cdot s(k)$.) Because our soundness errors are exponentially small in k, they remain so when multiplied by $N = k^c$.

Case Single Advice-Bounded Prover. During precomputation, the prover prepares N independently chosen sets of advice $\{E_{1,j}, E_{2,j}, s_{i,j}, s_{2,j}, \alpha_{1,j}, \alpha_{2,j}, \beta_{1,j}, \beta_{2,j}\}$ for $1 \leq j \leq N$, and uses the j-th choice for the j-th proof. All proofs are done independently. When proving concurrent zero-knowledge, the simulator operates as the one above for a single proof whenever the adversary requires the prover to send its second-round message for the jth proof; the simulator uses the x and ρ produced by V_j^*. The non-uniform polynomial-time indistinguishability of the real and simulated proofs follows by a hybrid argument as above, but having a factor of N more steps.[14] In the modified protocol, the precomputation time remains $\text{poly}(k)$. The length of the protocol advice increases to $O(k^{c+d+1})$. Therefore, to achieve soundness, d must be chosen sufficiently large to give a sufficiently large gap between the protocol advice bounds $O(k^{c+d+1})$ and δk^{2d} of the good prover and the cheating prover, respectively.

Case Multiple Advice-Bounded Provers. In this case, each copy of the prover chooses just one

[14] Obtaining a multiple-proof simulator from a single-proof simulator can be troublesome when the single-proof simulator involves "rewinding" the cheating verifier until it gives a desired response. If the N proofs are badly nested, the straightforward multiple-proof simulator can use time exponential in N. This is not a problem for our single-proof simulator, because it does not rewind the verifier.

set of advice $\{E_{1,j}, E_{2,j}, \ldots\}$, so its protocol advice bound is $O(k^{d+1})$. Because the k^c cheating provers can collude, each one should have an advice bound of δk^{2d-c} in order that the total length of the advice be at most δk^{2d}. Again we can obtain an arbitrarily large gap between $O(k^{d+1})$ and δk^{2d-c} by choosing d large enough.

Case Single Time-Bounded Prover. As in the case of a single advice-bounded prover, the prover's precomputation prepares k^c sets of advice. The difficulty in this case is that if all k^c first-round messages arrive simultaneously, the good prover P will need a factor k^c more time to respond to all the verifiers. So we must now let the timing bound used by each verifier be $k^e = k^{d+b+c}$ where $b = b(L, \mathcal{E}, \mathcal{Z})$ comes from Lemma 5.1. But this gives a cheating prover a factor k^c more time to cause one of the verifiers to accept in error. Thus we must choose d larger to make the computation of f (on inputs of length k^d) harder. This can be done simply by replacing b with $b + c$ wherever it appears. (For example, A5 in Theorem 7.1 is now $d > 1$, $d \geq \max(\,(b{+}c)/(m{-}1), c_1/(m{-}1), c_2/m\,)$, and $e = d + b + c$.) Because the exact value of b was not used in the proofs, replacing b by the larger $b + c$ does not affect the proofs.

Case Multiple Time-Bounded Provers. Each copy of the good prover is identical to the original P. The time bound used by each verifier can remain $k^e = k^{d+b}$ because each good prover must respond to only one verifier. A difficulty here is that k^c cheating provers can collude to compute f if this computation is sufficiently parallelizable. (This is obviously true for functions in $\mathcal{LIN}$.) So we must again increase d as in the previous case.

9 Future Work

Although two rounds are clearly necessary for this form of *time*-based interactive proof and argument systems, the same is not necessarily true for *advice*-bounded provers. The impossibility proof for 1-round systems does not apply. Indeed, consider the following protocol for a prover to convince a verifier of knowledge of the factorization of a number N. Let ζ be a fixed very large exponent, say, of size $|N|^3$. The prover is restricted to $|N|$ bits of advice and polynomial precomputation time. The good prover computes $z = \zeta \bmod \phi(N)$ during the precomputation phase, and uses z as its advice. During the protocol phase, the good prover chooses a random $x \in \mathbb{Z}_N^*$ and produces a very long sequence $x \bmod N, x^z \bmod N, x^{z^2} \bmod N, x^{z^3} \bmod N, \ldots$. Zero-knowledge is obvious: the simulator does not have to obey the advice bound, so the simulator can choose a random x and simply work with ζ instead of z. The (unproven!) intuition for soundness is that, without knowing $\phi(N)$, a cheating prover cannot find a short advice string that will allow it to sufficiently "compress" ζ to carry out the proof. The above protocol is suggestive, nothing more. We have not even defined an appropriate notion of a proof of knowledge for the 1-round setting.

Acknowledgements We are grateful to Russell Impagliazzo, David Jefferson, Bruce Kapron and John Mitchell for helpful discussions on what it means for a statement not to have a short proof, and to Marcos Aguilera, Mark Manasse, Li Zhang, and Yunhong Zhou for suggesting the version of the 1-round protocol described in Section 9.

References

[1] M. Ajtai and C. Dwork, A public-key cryptosystem with worst-case/average-case equivalence, *Proc. 29th ACM Symp. on Theory of Computing*, 1997.

[2] B. Barak, How to go beyond the black-box simulation barrier, *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, 2001.

[3] B. Barak, O. Goldreich, S. Goldwasser, and Y. Lindell, Resettably sound zero-knowledge and its applications, *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, 2001.

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

[5] M. Blum, P. Feldman, and S. Micali, Non-interactive zero-knowledge and its applications, *Proc. 20th ACM Symp. on Theory of Computing*, 1988, pp. 103–112.

[6] D. Boneh and M. Naor, Timed commitments, *Advances in Cryptology - CRYPTO 2000*, LNCS Vol. 1880, Springer-Verlag, 2000, pp. 236–254.

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

[8] R. Canetti, J. Kilian, E. Petrank, and A. Rosen, Black-box concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds, *Proc. 33rd ACM Symp. on Theory of Computing*, 2001.

[9] D. Dolev, C. Dwork, and M. Naor, Non-malleable cryptography, *SIAM J. Comput.* 30(2) (2000), pp. 391–437; preliminary version appeared in STOC'91.

[10] C. Dwork and M. Naor, Pricing via processing (or: combatting junk mail), *Advances in Cryptology - CRYPTO'92*, LNCS Vol. 740, Springer-Verlag, 1993, pp. 139–147.

[11] C. Dwork and M. Naor, Zaps and their applications, *Proc. 41st IEEE Symp. on Foundations of Computer Science*, 2000, pp. 283–293.

[12] C. Dwork, M. Naor, and A. Sahai, Concurrent zero-knowledge, *Proc. 30th ACM Symp. on Theory of Computing*, 1998, pp. 409–418.

[13] C. Dwork and A. Sahai, Concurrent zero-knowledge: Reducing the need for timing constraints, *Advances in Cryptology - CRYPTO'98*, LNCS Vol. 1462, Springer-Verlag, 1998, pp. 442–457.

[14] U. Feige, D. Lapidot, and A. Shamir, Multiple noninteractive zero knowledge proofs under general assumptions, *SIAM J. Comput.* 29(1) (1999), pp. 1–28.

[15] U. Feige and A. Shamir, Zero knowledge proofs of knowledge in two rounds, *Advances in Cryptology - CRYPTO'89*, LNCS Vol. 435, Springer-Verlag, 1989.

[16] U. Feige and A. Shamir, Witness indistinguishable and witness hiding protocols, *Proc. 22nd ACM Symp. on Theory of Computing*, 1990, pp. 416-426.

[17] O. Goldreich, Foundations of Cryptography (Fragments of a Book); see `http://www.wisdom.weizmann.ac.il/~oded/`.

[18] O. Goldreich, S. Goldwasser, and S. Halevi, Eliminating decryption errors in the Ajtai-Dwork cryptosystem, *Advances in Cryptology - CRYPTO'97*, pp. 105–111.

[19] O. Goldreich and A. Kahan, How to construct constant-round zero-knowledge proof systems for NP, *J. Crypto.* 9(3) (1996), pp. 167–190.

[20] O. Goldreich and H. Krawczyk, On the composition of zero knowledge proof systems, *SIAM J. Comput.* 25(1) (1996), pp. 169–192.

[21] O. Goldreich and Y. Oren, Definitions and properties of zero-knowledge proof systems, *J. Crypto.* 7(1) (1994), pp. 1–32.

[22] S. Goldwasser and J. Kilian, Almost all primes can be quickly certified, *Proc. 18th ACM Symp. on Theory of Computing*, 1986, pp. 316–329.

[23] S. Goldwasser and S. Micali, Probabilistic encryption, *J. Comput. Syst. Sci.* 28 (1984), pp. 270–299.

[24] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM J. Comput.* 18(1) (1989), pp. 186–208.

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

[26] J. Kilian and E. Petrank, An efficient non-interactive zero-knowledge proof system for NP with general assumptions, *J. Crypto.* 11(1) (1998), pp. 1–27.

[27] J. Kilian, E. Petrank, and C. Rackoff, Lower bounds for zero knowledge on the internet, *Proc. 39th IEEE Symp. on Foundations of Computer Science*, 1998, pp. 484-492.

[28] J. Kilian, E. Petrank, and R. Richardson, On concurrent and resettable zero-knowledge proofs for NP, `http://www.cs.technion.ac.il/~erez/publications.html`.

[29] M. Naor and M. Yung, Public-key cryptosystems provably secure against chosen ciphertext attacks, *Proc. 22nd ACM Symp. on Theory of Computing*, 1990, pp. 427–437.

[30] R. Richardson and J. Kilian, On the concurrent composition of zero-knowledge proofs, *Advances in Cryptology – EUROCRYPT'99*, LNCS Vol. 1592, Springer-Verlag, 1999, pp. 415–431.

[31] R. Rivest, L. Adleman, and M. Dertouzos, On data banks and privacy homomorphisms, in: *Foundations of Secure Computation*, Academic Press, 1978.

[32] A. Rosen, A note on the round-complexity of concurrent zero-knowledge, *Advances in Cryptology - CRYPTO 2000*, LNCS Vol. 1880, Springer-Verlag, 2000, pp. 451–468.

A Formal Definition of NIZKs

Let

$$ACCEPT(\sigma, t) = \{\, \pi \mid V \text{ accepts on input } \sigma, t, \pi \,\}$$

and let

$$REJECT(\sigma, t) = \{\, \pi \mid V \text{ rejects on input } \sigma, t, \pi \,\}.$$

Definition A.1 *A pair (P, V), where P (resp., V) is a probabilistic (resp., deterministic) polynomial-time machine with shared random string σ, is a* non-interactive zero-knowledge proof system *for the language $L \in$ NP if:*

Completeness: For all $t \in L$, for all $w \in W_L(t)$ and for random σ, with overwhelming probability for $\pi \in_R P(t, w, \sigma)$ we have $\pi \in ACCEPT(\sigma, t)$. The probability is over the choice of the shared string σ and the internal coin flips of P.

Soundness: For all $t' \notin L$ with overwhelming probability over σ and for all $\pi' \in \{0,1\}^*$ we have $\pi' \in REJECT(\sigma, t')$. The probability is over the choices of the shared string σ.

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 $t \in L$, and $w \in W_L(t)$,

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

where where the probability space is taken over the random choices of σ and over the random choices of Sim and P.

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