Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints

Cynthia Dwork[1] and Amit Sahai[2]

[1] IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120.. E-Mail: dwork@almaden.ibm.com.

[2] MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139, USA.. E-Mail: amits@theory.lcs.mit.edu***

Abstract. An interactive proof system (or argument) (P, V) is *concurrent zero-knowledge* if whenever the prover engages in polynomially many *concurrent* executions of (P, V), with (possibly distinct) colluding polynomial time bounded verifiers $V_1, \ldots, V_{poly(n)}$, the entire undertaking is zero-knowledge. Dwork, Naor, and Sahai recently showed the existence of a large class of concurrent zero-knowledge arguments, including arguments for all of NP, under a reasonable assumption on the behavior of clocks of nonfaulty processors. In this paper, we continue the study of concurrent zero-knowledge arguments. After observing that, without recourse to timing, the existence of a *trusted center* considerably simplifies the design and proof of many concurrent zero-knowledge arguments (again including arguments for all of NP), we design a preprocessing protocol, making use of timing, to simulate the trusted center for the purposes of achieving concurrent zero-knowledge. Once a particular prover and verifier have executed the preprocessing protocol, any polynomial number of subsequent executions of a rich class of protocols will be concurrent zero-knowledge.

1 Introduction

In order to be useful in the real world, cryptographic primitives and protocols must remain secure even when executed concurrently with other arbitrarily chosen protocols, run by arbitrarily chosen parties, whose identities, goals, or even existence may not be known. Indeed, this setting, characterized in [13] as a *distributed computing aggregate*, describes the Internet. Electronic interactions over an aggregate, such as economic transactions, transmission of medical data, data storage, and telecommuting, pose security risks inadequately addressed in computer science research. In particular, the issue of the security of *concurrent* executions is often[1] ignored.

*** Most of this work performed while at the IBM Almaden Research Center. Also supported by a DOD NDSEG doctoral fellowship, and DARPA grant DABT-96-C-0018.

[1] but not always, e.g. [1] in a different setting

A *zero-knowledge protocol* is supposed to ensure that no information is leaked during its execution. However, when zero knowledge interactions are executed *concurrently* both parties can be at risk. Consider the case of zero knowledge proofs: the verifier faces the possibility that the prover with which it is interacting is actually using some concurrently running second interaction as an "oracle" to help answer the verifier's queries – this is the classical chess master's problem. In the case of a proof of knowledge, the interaction may not actually yield a proof. This is an issue of potential *malleability* of the interactive proof system, and is addressed in [13]. In contrast, the prover faces the risk that concurrent executions of a protocol with many verifiers may leak information and may not be zero-knowledge *in toto*. In this case the interaction remains a proof but may fail to remain zero knowledge. This issue was first addressed in [16]. To overcome this difficulty, [16] introduce the notion of an (α, β) *constraint* for some $\alpha \leq \beta$:

> For any two (possibly the same) non-faulty processors P_1 and P_2, if P_1 measures α elapsed time on its local clock and P_2 measures β elapsed time on its local clock, and P_2 begins its measurement in real time no sooner than P_1 begins, then P_2 will finish after P_1 does.

As [16] points out, an (α, β) constraint is implicit in most reasonable assumptions on the behavior of clocks in a distributed system (*e.g.*, the linear drift assumption). According to the (standard) view that process clocks are under the control of an adversarial scheduler, the (α, β) constraint limits the choices of the adversary to schedules that satisfy the constraints.

Under an (α, β) constraint, [16] shows that there exist constant round concurrent zero-knowledge protocols of various kinds, for example, arguments for any language in NP[2]. In the protocols of [16], processors make explicit use of their local clocks in order to achieve concurrent zero-knowledge. The protocols require that certain timing constraints be met, which limit the kinds of protocol interleavings that can occur.

Our Contribution. In this work, we reduce the need for timing in achieving concurrent zero-knowledge. Specifically, for a rich class of interactive protocols, we are able push all use of timing into a constant round preprocessing phase; furthermore, the real time at which the preprocessing phase between a prover P and verifier V_1 occurs need not have any relation to the real time when P and a different verifier V_2 execute the preprocessing. After this preprocessing phase, the prover and the verifier can execute any polynomial number of a rich class of protocols without any further timing constraints, and the whole interaction will be concurrent zero-knowledge. We require the existence of a semantically secure public-key encryption scheme.

By limiting the use of timing to a single initial phase for each (P, V) pair, our methods can reduce the real execution time of protocols. This is because once preprocessing completes the parties never deliberately introduce timing delays in executing steps of future protocols. In contrast, in the protocols of [16] such deliberate delays play a critical role. For many applications, where two parties

[2] under verious standard computational assumptions

will be executing many zero-knowledge protocols, such as authentication with a system, these repeated delays may be expensive. Moreover, as we will see, our approach frequently yields simpler protocols that are easier to prove concurrent zero-knowledge.

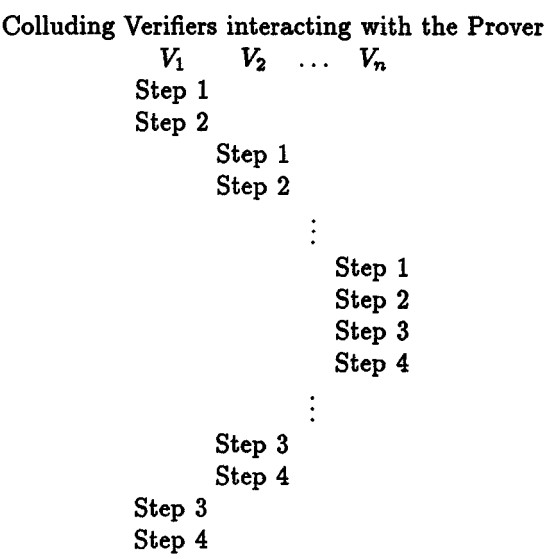

Diagram 1. A troublesome interleaving for concurrent zero-knowledge.

Interleavings of Protocols. The difficulty in achieving concurrent zero-knowledge is due to the existence of certain "bad" interleavings of concurrently executing protocols. The bad interleavings revolve around the difficulty of simulating a transcript of multiple concurrent interactions (recall that the ability to simulate an interaction is the core of the definition of zero-knowledge). Consider the standard (computational) zero-knowledge protocol for 3-colorability[3] [22], which can be based on any information-theoretic commitment scheme.

Generic Zero-Knowledge Argument for 3-Colorability:
1) $V \longrightarrow P$: Information-theoretic commitment to queries.
2) $P \longrightarrow V$: Commitment to graphs and colorings.
3) $V \longrightarrow P$: Open queries.
4) $P \longrightarrow V$: Open queried graphs or colorings, which V then checks are valid.

The standard simulator, having access only to V, produces transcripts of this protocol as follows. First, it receives V's commitment in Step 1. Then, supplying V initially with "garbage" in Step 2, the simulator discovers the queries V committed to through V's Step 3 response. The simulator uses this knowledge to construct graphs and colorings which would fool these particular queries. Then

[3] This is the "'parallelized" version that has negligible error while remaining zero-knowledge.

the simulator "rewinds" the interaction to just after Step 1, and supplies V with a commitment to these new graphs and colorings in Step 2. Since V is already committed by Step 1, its Step 3 response cannot change. Thus, the simulator can open the graphs and colorings according to the queries, and V will accept.

This simulator fails in the context of concurrent interactions because of the rewinding. Consider the following interleaving of n colluding verifiers following the generic four-round protocol described above.

An adversary controlling the verifiers can arrange that the Step 1 commitments to queries made by verifiers $V_{i+1}, \ldots, V_n$ can depend on messages sent by the prover in Step 2 of its interaction with V_i. It is a well-known open problem how to simulate transcripts with this interleaving in polynomial time; the difficulty with the straightforward approach is that once the queries in the interaction with V_i are opened (in Step 3), it becomes necessary to re-simulate Step 2 of the interaction with V_i, and therefore the entire simulation of the interaction with verifiers $V_{i+1}, \ldots, V_n$ must be re-simulated. The most deeply nested transaction, with V_n, is simulated roughly 2^n times.

Remark on Commitment Schemes The literature discusses two types of bit or string commitment: *computational* and *information-theoretic*. In computational string commitment there is only one possible way of opening the commitment. Such a scheme is designed to be secure against a probabilistic polynomial time receiver and an arbitrarily powerful sender. In information theoretic commitment it is possible to open the commitment in two ways, but the assumed computational boundedness of the sender prevents him from finding a second way. Such a scheme is designed to be secure against an arbitrarily powerful receiver and a probabilistic polynomial time prover. See [13] for a formal definition of computational commitment.

The commitments in Step 1 of the generic zero-knowledge argument must be information-theoretic, meaning that information theoretically nothing is leaked about the committed values. This is for soundness, rather than for zero-knowledge. Our techniques require that the verifier only use computational commitments (for example, as in the 6-round zero-knowledgeargument for NP of Feige and Shamir [19], which we modify for technical reasons).

The Trusted Center Model. Consider a model in which a trusted center gives out signed public key, private key pairs (E, D) of some public key cryptosystem to every user over a secure private channel. As we now explain, in this model arguments such as the one given in [19] can be simulated *without rewinding*, provided that the commitments by V are performed using the public key E given to it by the trusted center. This is significant because, if there is no rewinding, then interleavings such as the one described above are not problematic.

The simulator for V simulates its interaction with the trusted center as well as with P. So, the simulator knows the private key D corresponding to the public key E used in V's commitments. Hence, the simulator never has to rewind to learn a committed value. We call such simulations, in which rewinding is

avoided, *straight-line*[4]. In the case of straight-line zero-knowledge protocols, it is clear that concurrent interactions pose no threat to simulability, since the simulator can simulate each interaction independently in a straight-line fashion. This trusted center model is extremely powerful, and a great many standard protocols become straight-line zero-knowledge, and hence concurrent zero-knowledge, in the trusted center model with only minor modifications. For example, aside from arguments for NP, we also exhibit natural protocols for deniable message authentication and coin flipping.

Although straight-line zero-knowledge implies concurrent zero-knowledge without any timing constraints in the trusted center model, the notion of a trusted center that knows everyone's private keys and communicates over secure channels is undesirable or infeasible in many contexts. The preprocessing protocol of Section 4 uses timing to permit P and V to agree on a key E_V for V to use for commitments in their future interactions. Intuitively, the interaction ensures (with overwhelming probability) that V (perhaps with the collusion of other verifiers, but with absolutely no help from P), "knows" the corresponding decryption key D_V. Formally, the preprocessing protocol will ensure that subsequent interactions between P and V that would have been straight-line zero-knowledge in the trusted center model, are actually straight-line zero-knowledge in the conventional model.

2 Model and Definitions

Timing. We assume that all parties in any interaction have access to local clocks. Furthermore, as proposed in [16], we assume that there are known constants α and $\beta \geq \alpha$, for which the following (α, β) *constraint* holds:

> For any two (possibly the same) non-faulty processors P_1 and P_2, if P_1 measures α elapsed time on its local clock and P_2 measures β elapsed time on its local clock, vand P_2 begins its measurement in real time no sooner than P_1 begins, then P_2 will finish after P_1 does.

Zero-Knowledge and Concurrent Zero-Knowledge. In the original "black box" forumulation of zero-knowledge proof systems [24], an interactive proof system (P, V) for a language L is computational (or perfect) *zero-knowledge* if there exists a probabilistic, expected polynomial time oracle machine S, called the *simulator*, such that for every probabilistic polynomial time verifier strategy V^*, the distributions $(P, V^*)(x)$ and $S^{V^*}(x)$ are computationally indistinguishable (or identical) whenever $x \in L$. Here, formally, the machine V^* is assumed to take as input a partial conversation transcript, along with a random tape, and output the verifier's next response. This defintion also holds in the case of arguments [7], or computationally-sound proofs, where the prover and verifier are both probabilistic polynomial time machines.

[4] Note that without a trusted center or some other extra source of power, straight-line zero-knowledge is not an interesting concept, since any language that premits a straight-line zero-knowledge proof in the conventional sense must be in BPP – the simulator could act as the prover.

Following [16], to investigate preservation of zero-knowledge in a distributed setting, we consider a probabilistic polynomial time adversary that controls many verifiers simultaneously. Here, we consider an adversary A that takes as input a partial conversation transcript of a prover interacting with several verifiers concurrently, where the transcript includes the local times on the prover's clock when each message was sent or received by the prover. The output of A will either be a tuple $(\text{receive}, V, \alpha, t)$, indicating that P receives message α from V at time t on P's local clock, or (send, V, t), indicating that P must send a message to V at time t on P's local clock. The adversary must output a local time for P that is greater than all the times given in the transcript that was input to A (the adversary cannot rewind P), and standard well-formedness conditions must apply. If these conditions are not met, this corresponds to a non-real situation, so such transcripts are simply discarded. Note that we assume that if the adversary specifies a response time t for the prover that violates a timing constraint of the protocol with V, the prover should answer with a special null response which invalidates the remainder of the conversation with verifier V. The distribution of transcripts generated by an adversary A interacting with a prover P on common input x is denoted $(P \leftrightarrow A)(x)$.

We say an argument or proof system (P, V) for a language L is computational (or perfect) *concurrent zero-knowledge* if there exists a probabilistic, expected polynomial time oracle machine S such that for every probabilistic polynomial time adversary A, the distributions $(P \leftrightarrow A)(x)$ and $S^A(x)$ are computationally indistinguishable (or identical) whenever $x \in L$.

Note that since we assume that the prover acts honestly and follows the protocol, it does not matter if there is a single entity that is acting as the prover for all verifiers, or if there are many entities that are acting as provers for subsets of the verifiers, since the actions of the provers would be the same, and in our model, the timing of events is controlled by the adversary.

NIZK. Some of our protocols make use of non-interactive zero-knowledge (NIZK) proof constructions [5, 20, 2, 4] for languages in NP. Note, however, that although typically one considers NIZK in a model where all parties share a public random string, we do *not* make any such assumptions in any model we consider. In a NIZK proof, the prover P and verifier V have a common input x and also share a random string σ, called the reference string, of length polynomial in the length of x. The prover wishes to convince the verifier of the membership of x in some fixed NP language L. To this end, the prover is allowed to send the verifier a single message $m = P(x, \sigma)$, computed (probabilistically) as a function of x, σ. The probabilistic polynomial time verifier must decide whether or not to accept as a function of x, σ, and m. Such an interaction (P, V) is an NIZK proof system for L if: (1) If $x \in L$, for all σ, $(P, V)(x, \sigma)$ accepts. (2) If $x \notin L$, for all P^*, the probability over σ and the random coins of P and V that $(P^*, V)(x, \sigma)$ accepts is negligible. (3) There exists a probabilistic polynomial time simulator S such that, if $x \in L$, then the distributions $S(x)$ and $(\sigma, P(x, \sigma))$, where in the latter distribution σ is chosen uniformly, are computationally indistinguishable. We further ask that the prover be probabilistic polynomial time, but also allow

that in the case when $x \in L$, the prover is given a witness w for the membership of $x \in L$. We require, however, that the distribution $(\sigma, P(x, \sigma, w))$ be computationally indistinguishable from $S(x)$ no matter how the witness w is chosen.

[20,2] show that such NIZK proof systems with efficient provers exist for every language in NP assuming trapdoor permutations exist. Note that the definition above gives "bounded" NIZK proof systems, *i.e.*, a given reference string σ can be used to prove only one NP statement. We also require unbounded NIZK proof systems, in which any polynomial number of NP statements can be proven in zero-knowledge using the same reference string σ. [12,20] have shown that the existence of a bounded NIZK proof system for an NP-complete language L with an efficient prover implies the existence of unbounded NIZK proof systems with efficient provers for any language in NP. A precise definition of unbounded NIZK can be found in [2,4,20].

Note that NIZK proofs are truly non-interactive only if the prover and the verifier already agree on a random string σ, which we do not assume. Furthermore, if the distribution of σ is far from uniform, then the zero-knowledge condition fails to hold. This issue motivates our concurrently simulable random selection protocol, described below.

Assumptions. We require a probabilistic public key cryptosystem that is semantically secure [23]. The encryptions must be uniquely decodable (so the scheme must be undeniable [9]). We will use the public key cryptosystem for *computationally secure string commitment* as follows: V uses an encryption key E to commit to a string s by simply sending an encryption e of s using a random string r. To open s, V sends s, r; the receiver checks that $e = E(s, r)$.

3 Straight-Line Zero-Knowledge in The Trusted Center Model

In order to define the class of protocols for which our construction applies, we consider a trusted center model, in which the trusted center communicates to each participant a public key, private key pair (E, D) for a public key cryptosystem, over a secure private channel. More formally, we assume the existence of a trusted center with which any party can interact using a secure private channel. In our model, before any protocol (P, V) begins, first V receives a public key and private key pair (E, D) from the trusted center over a secure channel, whereas P receives only E from the trusted center. Then the interaction takes place as in the normal model. Also in our trusted center model, we modify the definition of zero-knowledge to require the simulator to also simulate the interaction with the trusted center, which in particular means that the simulator produces the public key E and private key D given to any verifier.

We use the trusted center model only for definitional purposes; our protocols *do not* assume a trusted center. In particular, we will define the class of protocols that are *straight-line zero-knowledge in the trusted center model* and argue that any protocol in this class is concurrent zero-knowledge. We will show that this

class is rich, and, in the next section, show how to use timing to replace the role of the trusted center by means of a preprocessing phase.

As noted above, the trusted center model is extremely powerful; in particular, in this model Rackoff and Simon were able to define and construct noninteractive zero-knowledge proofs of knowledge [27]. These noninteractive proofs could then be used to prove "plaintext awareness" of encrypted messages (intuitively, that the sender "knows" what he is sending), resulting in a cryptosystem secure against the most general chosen ciphertext attacks (called *chosen ciphertext in the post-processing mode* in [13]; the construction of the cryptosystem in [27] also requires the trusted center to establish a digital signature scheme). This is equivalent to non-malleable security against a post-processing chosen ciphertext attack [13].

The trusted center model also proves useful in the context of concurrent zero-knowledge. In particular, if the protocol requires that commitments be made using the key E chosen by the trusted center, then the construction of concurrent zero-knowledge protocols becomes a simpler task: the simulator simulates each party's interaction with the trusted center, and hence knows the secret key D for each public key E later used in the protocol.

We will prove concurrent zero-knowledge in the trusted center model by establishing an even stronger property, *straight-line zero-knowledge*. Intuitively, a protocol is straight-line zero-knowledge if there exists a simulator that does no "re-winding" in order to produce its transcript. Formally, a zero-knowledge interactive protocol (P, V) is *straight-line zero-knowledge* if the simulator S for the protocol is of a special form. Recall S is in general an expected polynomial time machine that uses a verifier strategy V^* as an oracle, giving it partial transcripts and obtaining the verifier's next message as its response. Define $\mathcal{O}(S^{V^*}(x;r)) = (q_1, q_2, \ldots, q_m)$ to be the ordered sequence of oracle queries (partial transcripts) given by S to V^*, on input x and using random bits r. We require for every V^*, x, and r, letting $\mathcal{O}(S^{V^*}(x;r)) = (q_1, q_2, \ldots, q_m)$, that the transcript q_i is a prefix of q_{i+1} for $1 \leq i \leq m-1$. Such a simulator S is called a *straight-line simulator*.

It is immediate that a straight-line zero-knowledge protocol is also concurrent zero-knowledge, since interaction with many verifiers simultaneously can be simulated by simply simulating the interaction with each verifier separately using the straight-line simulator.

Note that, in the conventional framework for zero-knowledge proofs, straight-line zero-knowledge proofs can only exist for languages in BPP, since the polynomial time simulator can act as the prover. In the trusted center model, this need not concern us, since the real prover cannot impersonate the trusted center, whereas the simulator can.

3.1 Examples

The class of protocols with straight-line simulators in the trusted center model is rich.

Generic NP. The generic argument for membership in an NP language, described in Section 1 requires information-theoretic commitments on the part of

the verifier, and therefore does not fit our model. The 6-round protocol of Feige and Shamir [19], which can be based on any one-way function, can be modified to be straight-line zero-knowledge in the trusted center model provided that the verifier's commitments are made using the key received from the trusted center. The modification involves using an information-theoretic commitment scheme for an additional initial commitment by the prover (details are omitted for lack of space). Thus, there is a constant round straight-line zero-knowledge argument in the trusted center model for every language in NP based only on the existence of semantically secure public-key encryption and information-theoretic commitment schemes. However, there is a simpler, four round scheme, based on the existence of trapdoor permutations, which we present below.

Random String Selection. We next describe a random string selection (coin-flipping) protocol, by which two parties P and V can select a random string which will be random as long as one of the parties is honest. The random selection protocol has the following extremely useful property: in the trusted center model, the simulator can force any desired string as the outcome of the protocol; moreover, the distribution of simulated transcripts is identical to the distribution on actual transcripts, conditioned on any given output. In the sequel, let E denote the public key assigned to the verifier by the trusted center. The natural cryptographic protocol for random selection of strings of length k, due to [3], can be made straight-line simulable as follows[5]:

1) $V \longrightarrow P : E(r_V) : r_V \in_R \{0,1\}^k$
2) $P \longrightarrow V : r_P : r_P \in_R \{0,1\}^k$
3) $V \longrightarrow P$: Reveal coins used to generate $E(r_V)$.

The output of the protocol is $r_V \oplus r_P$.

Since $E(r_V)$ is a computationally secret commitment to any party that does not know D, it is clear that this protocol achieves the desired random selection properties. Here, the simulator, on input $x \in \{0,1\}^k$, (after having simulated the trusted center, *i.e.*, having supplied V with a public key, private key pair (E, D)), receives the verifier's Step 1 message. Using D, it recovers r_V. The simulator then supplies V^* with the Step 2 message $x \oplus r_V$. In Step 3, V^* must decommit to r_V, and so the output of the protocol is x. If the input x is chosen uniformly at random, then the simulator's Step 2 message, $x \oplus r_V$ will also be uniformly random. Hence, the simulator's distribution will actually be identical to the distribution of actual transcripts.

Alternative NP. We can use the random string selection protocol to give a 4-round alternative to the straight-line zero-knowledge argument system for NP that is also straight-line zero-knowledge in the trusted center model, if we make the stronger assumption that trapdoor permutations exist. By [20,2], assuming trapdoor permutations exist, there are efficient prover NIZK proofs for NP. Recall that NIZK proofs require a random string to be shared by the parties P and V. We will use the random selection protocol above to select this shared

[5] It is interesting that if the roles of P and V are reversed, P's new role in the protocol is no longer known to be simulable.

randomness. The following protocol is therefore an argument proving $x \in L$ for any language $L \in NP$:

1) $V \longrightarrow P : E(r_V) : r_V \in_R \{0,1\}^{\text{poly}(|x|)}$
2) $P \longrightarrow V : r_P : r_P \in_R \{0,1\}^{\text{poly}(|x|)}$
3) $V \longrightarrow P$: Reveal coins used to generate $E(r_V)$.
4) $P \longrightarrow V$: NIZK proof that $x \in L$ using reference string $\sigma = r_V \oplus r_P$.

Note that Step 4 can later be repeated any (polynomial) number of times to prove different statements in NP once the reference string has been established (using an unbounded NIZK proof system such as the one given in [20]).

A straight-line simulator proceeds as follows. First, it calls the simulator $S_{NIZK}(x)$ of the NIZK proof system, which produces a reference string σ and an NIZK proof p. The straight-line simulator given above for the random selection protocol is then invoked to produce a transcript for Steps 1-3 that force $r_V \oplus r_P = \sigma$. The simulator then outputs p as the prover's Step 4 message, and terminates the simulation. Since the distribution $S_{NIZK}(x)$ is computationally indistinguishable from the distribution (σ, p) where σ is truly random, and p is generated by the NIZK prover's algorithm, we have that the distribution of our straight-line simulator will be computationally indistinguishable from the distribution of actual conversation transcripts of this protocol.

Deniable Message Authentication. NIZK proofs can also be useful for constructing straight-line zero-knowledge protocols for other applications. Consider the problem of *deniable message authentication* [13, 15, 16]. Here, the prover wishes to authenticate a message m to the verifier, in such a way that no other party can verify the authentication. In particular, we require that verifiers cannot prove to anyone else that the prover authenticated m. It suffices that the protocol be concurrent zero-knowledge, since if the verifiers could generate the transcript of their conversations with the prover on their own, then certainly it will not be possible to prove that the prover authenticated m to any other party. We exhibit a very natural protocol for this task, with a slight modification to make it straight-line zero-knowledge. For this protocol, we will require a non-malleable public key encryption scheme. Note that the encryption scheme must be non-malleable in the conventional model, *not* just in the trusted center model! Let the prover's public non-malleable encryption key be E_P, for which it alone knows the private key.

1) $V \longrightarrow P : E(r_V)$, where $r_V \in_R \{0,1\}^{\text{poly}(|x|)}$
2) $P \longrightarrow V : r_P$, where $r_P \in_R \{0,1\}^{\text{poly}(|x|)}$
3) $V \longrightarrow P$: Reveal coins used to generate $E(r_V)$. Choose $r \in_R \{0,1\}^k$.
   Send $E_P(m \circ r)$, $y = E(r)$, and an NIZK proof that
   the two encryptions sent are consistent with some $r \in \{0,1\}^k$
   using reference string $\sigma = r_V \oplus r_P$.
4) $P \longrightarrow V : r$

Note that the first two steps can be omitted if there is some other source of a random string to use as the reference string for the NIZK proof. Such a string

could, for example, be found as part of the public key for the prover's encryption scheme, as it is in the construction of [13]. The straight-line simulator simulates the first 2 steps trivially. In Step 3, after checking the NIZK, the simulator uses D to decrypt y, yielding r_S. Note that if the NIZK proof was accepted, then the decryption will correctly give $r_S = r$ with all but negligible probability. Hence, the simulator simply outputs r_S as the prover's final message, and terminates. The simulator will thus fail with no more than a negligible probability.

These examples illustrate not only that the class of straight-line zero-knowledge protocols in the trusted center model is rich, but also that it is not difficult to construct proofs that fit this definition. In the next section, we show how to eliminate the trusted center for the purpose of concurrent zero-knowledge using a preprocessing protocol based on timing constraints.

4 The Preprocessing Protocol

In this section, we show how to how to achieve concurrent zero-knowledge without the trusted center for all protocols that are straight-line zero-knowledge in the trusted center model. This is accomplished by replacing the trusted center with a preprocessing protocol that employs timing constraints. This eliminates the trusted center for the purpose of maintaining concurrent zero-knowledge.

Let G be the generator for a public-key cryptosystem which requires $\ell(n)$ random bits. We will write $G(1^n, \sigma) = (E, D)$ to mean that G, when given security parameter n and random bits $\sigma \in \{0,1\}^{\ell(n)}$, produces the public encryption algorithm E and private decryption algorithm D. Let C be a secure commitment scheme, such as the elegant scheme of [26]. The protocol uses the Basic Commit with Knowledge protocol (**BCK**) of [13].

Preprocessing Protocol:

0. V : Generates random strings $\sigma, r_0^1, r_0^2, \ldots r_0^n, r_1^1, r_1^2, \ldots, r_1^n \in \{0,1\}^{\ell(n)}$
   V runs $G(1^n, \sigma)$ to produce E and D, and sets up the scheme C.
1. $V \longrightarrow P$: $E, C(\sigma), C(r_0^1), C(r_0^2), \ldots, C(r_0^n), C(r_1^1), C(r_1^2), \ldots, C(r_1^n)$.
2. $P \longrightarrow V$: Random bits $b_1, b_2, \ldots, b_n$.
3. $V \longrightarrow P$: For each i, opens $C(r_{b_i}^i)$ and sends $r_{(1-b_i)}^i \oplus \sigma$.
4. $V \longleftrightarrow P$: Verifier gives a ZK argument (e.g. [19]) of consistency of the Step 1-3 messages above with some σ such that $G(1^n, \sigma)$ produces E.
5. $P \longrightarrow V$: If Step 4 is accepted, send "READY."

Timing Constraints: (1) P requires the Step 3 message to be received before time α has elapsed on its local clock since the Step 1 message was sent. If a verifier V fails to respond in this allotted time, we say V has *timed out.*
(2) P does not send the Step 5 signal until time β has elapsed on its local clock since Step 1.

For zero-knowledge, we assume that the adversary is constrained by an (α, β)-constraint. For completeness we must also assume that V can send its Step 3 message in the real time required for time α to elapse on P's local clock.

The following theorem shows that these timing constraints effectively eliminate all problematic interleavings.

Theorem 1. *Assuming that a semantically secure public-key cryptosystem exists, the Preprocessing Protocol, followed by any polynomial number of protocols that are straight-line zero-knowledge in the trusted center model, is (computational) concurrent zero-knowledge.*

Furthermore, using the Preprocessing Protocol does not give the Prover any advantage it would not have in the original protocol. In particular, using the Preprocessing Protocol does not endanger the soundness of any protocol that was sound in the trusted center model.

Theorem 2. *Let $\pi = (P, V)$ be any protocol in the trusted center model, and let $\pi' = (P', V')$ be the Preprocessing Protocol followed by π in the normal model. Then, for any probabilistic polynomial time prover P^*, there exists another prover P^{**} such that the distribution of transcripts from (P^*, V') (not including the transcript of the Preprocessing Protocol) is computationally indistinguishable from the distribution of transcripts from (P^{**}, V) (not including the initial interaction with the trusted center).*

Proof (of Theorem 2). Theorem 2 follows from the fact that the verifier's role in the Preprocessing Protocol is simulable without knowing D or a σ such that $G(1^n, \sigma) = (E, D)$. P^{**} behaves as follows: First it simulates the Preprocessing-protocol with P^*, and then it simply behaves as P^* does afterwards.

From the trusted center, it first receives E. Then it generates random strings σ, $r_0^1, r_0^2, \ldots r_0^n, r_1^1, r_1^2, \ldots, r_1^n \in \{0,1\}^{\ell(n)}$, and sets up the commitment scheme C with P^*. It sends $E, C(\sigma), C(r_0^1), C(r_0^2), \ldots, C(r_0^n), C(r_1^1), C(r_1^2), \ldots, C(r_1^n)$ to P^*, who responds with some random bits $b_1, b_2, \ldots, b_n$. P^{**} then for each i, opens $C(r_{b_i}^i)$ and sends $r_{(1-b_i)}^i \oplus \sigma$. Note that all this time σ had absolutely nothing to do with E. However, by the zero-knowledge property of the argument of Step 4, P^{**} can simulate Step 4 as if $G(1^n, \sigma) = (E, D)$. By this zero-knowledge condition, the transcript of all steps so far is computationally indistinguishable from a partial transcript of a real interaction of V' and P^*. Since the state of P^* at the end of Step 4 (i.e. the beginning of π) is computationally indistinguishable from its state at this point in a real interaction with V', the theorem follows.

Proof (of Theorem 1, using the proof techniques of [16].). First, we observe that the timing constraints yield the following interleaving contraint:

Interleaving Constraint: While any verifier is in Steps 1-3 and has not timed out, no new interaction can be started and complete Step 5.

Note that this remains true by the (α, β)-constraint even if there are many provers with different local clocks, since the time between Steps 1-3 must always be less in real time than the minimum delay in real time between Steps 1-5.

Note also that in all steps the prover needs no special information to carry out her side of the protocol. Hence, for any particular verifier, all steps in the Preprocessing protocol are trivial to simulate (perfectly) in a straight-line fashion. To be able to simulate any subsequent protocol that is straight-line zero-knowledge in the trusted center model, we appeal to the following Lemma:

Lemma 1. *For any verifier V, if the simulator has Step 3 responses for two different Step 2 queries (for the same Step 1 message), then the simulator can simulate all subsequent executions of protocols with V that are straight-line zero-knowledge in the trusted center model (computationally) in a straight-line fashion with all but negligible probability.*

Proof. Since the Step 2 queries were different, there exists an i such that V opened both r_0^i and r_1^i, and (supposedly) supplied both $a = r_1^i \oplus \sigma$ and $b = r_0^i \oplus \sigma$. The simulator can test both $a \oplus r_1^i$ and $b \oplus r_0^i$ by running G with these inputs. If G produces V's encryption key E, then we know that the decryption key D produced must be V's secret key. If this is the case, the simulator has V's secret key and can simulate all future protocols that are straight-line zero-knowledge in the trusted center model (computationally) in a straight-line fashion, by assumption.

If not, for any future simulation of V, when the simulator obtains a Step 3 response γ from V, it again checks with the responses it already has to see if this new response will yield a valid secret key for V. If this is not the case, then the simulator has proof that γ is inconsistent with the commitments in V's Step 1 message. Hence with all but negligible probability, V will not pass Step 4 of the protocol, by the soundness of the ZK Argument. Note that the soundness condition for ZK Arguments requires that no PPT prover can succeed in proving a false statement with more than negligible probability. The interaction of the PPT A and the PPT simulator together certainly still comprise a PPT system, and hence cannot prove the false statement with more than negligible probability. Thus with all but negligible probability, V never makes it past Step 4, and the Lemma follows.

We now describe a subroutine of the simulator called `Extract`. The subroutine takes two arguments, the name of a verifier V_i, and a partial starting transcript T that includes the Step 1 message of V_i. `Extract`(V_i, T) is only called if the simulator already has obtained one Step 3 response from V_i. The purpose of calling `Extract` on V_i is to create for V_i the situation required by the Lemma.

In `Extract`(V_i, T) the simulator repeats the following procedure *as many times as needed* to obtain another Step 3 response from V.

- Starting with partial transcript T, begin a simulation until either V_i gives a Step 3 response or more than time α has passed since Step 1 of V_i. During this simulation:
  - For any verifiers introduced after the Step 1 message of V_i, by the Interleaving Constraint, we know these verifiers will never proceed past Step 5 in the time allotted, so simulate the interaction with them perfectly.

- If any verifier V_j which was introduced before the Step 1 message of V_i gives a Step 3 response:
  - If the simulator has already obtained two Step 3 responses from V_j, by the Lemma all interaction with V_j can be simulated in a straight-line fashion.
  - If not, the simulator executes $\texttt{Extract}(V_j, T)$.

Thus, we are guaranteed that after executing $\texttt{Extract}(V_i, T)$ we have received two Step 3 responses from V_i. If the two responses received are the same, the simulator fails. This can only happen if the random bits chosen by the simulator in Step 2 were identical, an event with exponentially small probability. We will later argue that the simulator is expected polynomial time. Hence the simulator only gets an expected polynomial number of chances to fail in this manner, and so its probability of failure in this way is negligible. If the two Step 3 responses are different, such a verifier that has satisfied the conditions of the Lemma is called *neutralized.*

Now, to generate its output transcript the simulator begins a straight-line simulation with the adversary. Whenever a verifier V that has not already been neutralized gives a Step 3 response, the simlator calls $\texttt{Extract}(V, T)$, where T is the partial transcript up to and including the Step 1 message of V. When `Extract` terminates, V has been neutralized and thus, by the Lemma, all future interaction with V can be simulated in a straight-line fashion. This continues until the simulation is concluded.

By construction, the distribution of transcripts produced by such a simulation is computationally indistinguishable from those arising from an actual interaction with the adversary, since the transcript is produced in a straight-line manner.

We must confirm that the expected running time of the simulator is polynomially bounded. Each trial within *Extract*, not counting time taken by recursive function calls, certainly takes polynomial time, say $O(n^c)$. Let us analyze the expected running time of a function call to *Extract*. Now, conditioned on some partial transcript T, let X_i denote the random variable for the time to complete *Extract*(V_i, T). Let p_i denote the probability over simulations starting at T that V_i will give its Step 3 response during a simulation trial. If V_1 is the first verifier appearing in T, then during *Extract*(V_1, T), no recursive calls can be made for other verifiers. Hence, $X_1 \leq O(n^c)(1+p_1+(1-p_1)(1+X_1))$, and so by linearity of expectation, $E(X_1) \leq O(n^c)\frac{2}{p_1}$. More generally, if V_i is the i'th verifier appearing in the transcript T, then $X_i \leq O(n^c)(1+\sum_{j=1}^{i-1} p_j X_j + p_i + (1-p_i)(1+X_i))$, and a simple induction shows that $E(X_i) \leq O(n^c)\frac{2i}{p_i}$. Now, in the simulation, conditioned on a partial transcript T, the probability that *Extract*(V_i, T) will be called (from outside *Extract*) is exactly p_i. Thus, the expected amount of time the simulation will spend on *Extract*(V_i, T) is $O(n^c) \cdot 2i$. Since this does not depend on T, we can remove the conditioning and conclude that the expected amount of time the simulation will spend in `Extract` for V_i will be $O(n^c) \cdot 2i$. We note that the total number of verifiers that can be present in the final transcript is

bounded by the amount of time the adversary runs for. Hence, if the adversary's expected running time is $t(n)$, the expected amount of time the simulation will spend in *Extract* for all of the verifiers is $O(n^c t(n)^2)$. The rest of the simulation will certainly take no more than expected $O(n^c)t(n)$ time, and so we conclude that the expected running time is polynomial in n.

5 Additional Remarks and Future Research

Recently, Kilian, Petrank and Rackoff [25] have shown that any 4-round nontrivial zero-knowledge interactive proof is not black-box simulatable under concurrent executions. In a companion paper, Richardson and Kilian [28] have shown that for any $\varepsilon > 0$ and any upper bound k on the amount of concurrency to be tolerated, there exists a zero-knowledge proof for any language in NP requiring k^ε messages. We believe that the Kilian-Richardson protocol can be used (with some modifications) as a preprocessing protocol to allow subsequent constant round concurrent zero-knowledge arguments for any statement in NP; we are in the process of checking the details.

The difficulties in achieving non-malleability and concurrent zero knowledge both stem from potentially bad protocol interleavings. In [16] and in the preprocessing protocol described in this paper, timing is used explicitly to proscribe certain interleavings in order to achieve concurrent zero-knowledge. Can timing be used in a natural way to achieve non-malleability? For concreteness, consider non-malleable string commitment. Since non-malleable string commitment protocols exist without recourse to timing [13], any timing-based solution would be interesting only if it is efficient or simple.

Can timing be used to achieve other cryptographic objectives?

References

1. M. Bellare and P. Rogaway, *Provably Secure Session Key Distribution - The Three Party Case*, Proc. 27th STOC, 1995, pp 57–64.
2. M. Bellare and M. Yung. *Certifying permutations: Noninteractive zero-knowledge based on any trapdoor permutation*, Journal of Cryptology, 9(3):149-166, 1996.
3. M. Blum. *Coin flipping by telephone: A protocol for solving impossible problems.* In Allen Gersho, editor, Advances in Cryptology: A Report on CRYPTO 81, pages 11-15, 24-26 August 1981. Department of Electrical and Computer Engineering, U. C. Santa Barbara, ECE Report 82-04, 1982.
4. M. Blum, A. De Santis, S. Micali, and G. Persiano. *Noninteractive zero-knowledge*, SIAM Journal on Computing, 20(6):1084-1118, 1991.
5. Blum M., P. Feldman and S. Micali, *Non-Interactive Zero-Knowledge Proof Systems*, Proc. 20th ACM Symposium on the Theory of Computing, Chicago, 1988, pp 103-112.
6. G. Brassard, C. Crepeau and M. Yung, *Constant-Round Perfect Zero-Knowledge Computationally Convincing Protocols.* Theoretical Computer Science 84, 1991.
7. G. Brassard, D. Chaum, C. Crepeau, *Minimum Disclosure Proofs of Knowledge.* JCSS, Vol. 37, 1988, pp. 156-189.
8. S. Brands and D. Chaum, *Distance-Bounding Protocols* Advances in Cryptology – EUROCRYPT'93, 1993.

9. R. Canetti, C. Dwork, M. Naor, R. Ostrovsky, *Deniable Encryption*, "Security in Communication Networks" workshop, Amalfi, Italy 1996 and CRYPTO'97
10. D. Chaum and H. van Antwerpen, *Undeniable Signatures*, Advances in Cryptology–CRYPTO '89, G. Brassard (Ed.), Springer-Verlag, pp. 212-216.
11. R. Cramer and I. Damgard *New Generation of Secure and Practical RSA-Based Signatures*, Advances in Cryptology–CRYPTO '96. Springer-Verlag, 1996.
12. A. De Santis and M. Yung. *Cryptographic Applications of the Metaproof and Many-prover Systems*, Proc. CRYPTO'90, Springer-Verlag, 1990.
13. D. Dolev, C. Dwork and M. Naor, *Non-malleable Cryptography*, Preliminary version: Proc. 21st STOC, 1991. Full version: *submitted for publication* (available from the authors).
14. C. Dwork and M. Naor, *Pricing via Processing -or- Combatting Junk Mail*, Advances in Cryptology – CRYPTO'92, Lecture Notes in Computer Science
15. C. Dwork and M. Naor, *Method for message authentication from non-malleable crypto systems*, US Patent No. 05539826, issued Aug. 29th 1996.
16. C. Dwork, M. Naor, and A. Sahai, *Concurrent Zero Knowledge*, to appear, STOC'98
17. U. Feige, A. Fiat and A. Shamir, *Zero Knowledge Proofs of Identity*, J. of Cryptology 1 (2), pp 77-94. (Preliminary version in STOC 87).
18. U. Feige and A. Shamir, *Witness Indistinguishable and Witness Hiding Protocols* Proc. 22nd STOC, 1990, pp. 416–426.
19. U. Feige and A. Shamir, *Zero Knowledge Proofs of Knowledge in Two Rounds, Advances in Cryptology – Crypto* 89 *Proceedings*, Lecture Notes in Computer Science Vol. 435, G. Brassard ed., Springer-Verlag, 1989.
20. U. Feige, D. Lapidot and A. Shamir, *Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String*, Proceedings of 31st Symposium on Foundations of Computer Science, 1990, pp. 308-317.
21. O. Goldreich, **Foundations of Cryptography** (Fragments of a Book), 1995. Electronic publication: http://www.eccc.uni-trier.de/eccc/info/ECCC-Books/eccc-books.html (Electronic Colloquium on Computational Complexity).
22. O. Goldreich and H. Krawczyk. *On the Composition of Zero Knowledge Proof Systems*. SIAM J. on Computing, Vol. 25, No. 1, pp. 169–192, 1996.
23. S. Goldwasser and S. Micali. *Probabilistic Encryption*, Journal of Computer and System Sciences, Vol. 28, April 1984, pp. 270–299.
24. S. Goldwasser, S. Micali, and C. Rackoff, *The Knowledge Complexity of Interactive Proof Systems*. SIAM Journal on Computing, Vol. 18, 1 (1989), pp. 186-208.
25. J. Killian, E. Petrank, and C. Rackoff, *Zero Knowledge on the Internet*. Manuscript, 1998.
26. M. Naor, *Bit Commitment Using Pseudo-Randomness*, Journal of Cryptology, vol 4, 1991, pp. 151-158.
27. C. Rackoff and D. Simon, *Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Cipherext Attack*, Proc. CRYPTO'91, Springer-Verlag, 1992, pp. 433 – 444
28. R. Richardson and J. Killian. *Non-Synchronized Composition of Zero-Knowledge Proofs*. Manuscript, 1998.