Non-Malleable Cryptography *

Danny Dolev* Cynthia Dwork† Moni Naor‡

Abstract

The notion of *non-malleable* cryptography, an extension of semantically secure cryptography, is defined. Informally, the additional requirement is that given the ciphertext it is impossible to generate a *different* ciphertext so that the respective plaintexts are related. The same concept makes sense in the contexts of string commitment and zero-knowledge proofs of possession of knowledge. Non-malleable schemes for each of these three problems are presented. The schemes do not assume a trusted center; a user need not know anything about the number or identity of other system users.

Keywords: cryptography, cryptanalysis, randomized algorithms, nonmalleability

AMS subject classifications: 68M10, 68Q20, 68Q22, 68R05, 68R10

---

*A preliminary version of this work appeared in STOC '91

*Hebrew University Jerusalem, Israel

†IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120. E-mail: dwork@almaden.ibm.com.

‡Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. Most of this work performed while at the IBM Almaden Research Center. Research supported by an Alon Fellowship and a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences. E-mail: naor@wisdom.weizmann.ac.il.

1 Introduction

Consider the problem of *contract bidding*: Municipality M has voted to construct a new elementary school, has chosen a design, and advertises in the appropriate trade journals, inviting construction companies to bid for the contract. The advertisement contains a public-key E to be used for encrypting bids, and a FAX number to which encrypted bids should be sent. Company A places its bid of $\$1,500,000$ by FAXing $E(1,500,000)$ to the published number over an insecure line. Intuitively, the public-key cryptosystem is *malleable* if, having access to $E(1,500,000)$, Company B is more likely to generate a bid $E(\beta)$ such that $\beta \leq 1,500,000$ than Company B would be able to do without the ciphertext. Note that Company B *need not* be able to decrypt the bid of Company A in order to consistently just underbid. In this paper we describe a non-malleable public-key cryptosystem that prevents such underbidding. Our system does not even require Company A to know of the existence of Company B.

In another scenario, suppose Researcher A has obtained a proof that $P \neq NP$ and wishes to communicate this fact to Professor B. Clearly A will initially do this in a zero-knowledge fashion, but is zero-knowledge sufficient? Professor B may try to steal credit for this result by calling eminent Professor E and acting as a *transparent prover*. Any questions posed by Professor E to Professor B are relayed by the latter to A, and A's answers to Professor B are then relayed in turn to Professor E. We solve this problem with a *non-malleable zero-knowledge proof of knowledge*. Researcher A will get proper credit even without knowing of the existence of Professor E, and even if Professor E is unaware of Researcher A.

Goldwasser and Micali define a cryptosystem to be *semantically secure* if anything computable about the cleartext from the ciphertext is computable without the ciphertext [17]. This powerful type of security may be inadequate in the context of a distributed system, in which the mutual independence of messages sent by distinct parties often plays a critical role. In particular, a semantically secure cryptosystem may not solve the contract bidding problem. Informally, a cryptosystem is *non-malleable* if the ciphertext doesn't help: given the ciphertext it is no easier to generate a *different* ciphertext so that the respective plaintexts are related than it is to do so without access to the ciphertext. In other words, a system is non-malleable if, for every relation R, given a ciphertext $E(\alpha)$, one cannot generate a different ciphertext $E(\beta)$ such that $R(\alpha, \beta)$ holds any more easily than can be done without access to $E(\alpha)$[1]. We present a non-malleable public-key cryptosystem. Our cryptosystem does not assume a trusted center, nor does it assume that any given collection of users knows the identities of other users in the system. In contrast, all other research touching on this problem of which we are aware requires at least one of these assumptions (e.g., [6, 7, 27]).

To further motivate non-malleable public key cryptography, in Section 3.4 we discuss an extremely simple protocol for *public key authentication*, a relaxation of digital signatures that permits an authenticator A to authenticate messages m, but in which the authentication needn't be verifiable by a third party. The protocol requires a non-malleable public key cryptosystem, and is simply incorrect if the cryptosystem is malleable.

[1]Clearly, there are certain kinds of relations R that we cannot rule out. For example, if $R(\alpha, \beta)$ holds precisely when $\beta \in E(\alpha)$ then from $E(\alpha)$ it is trivial to compute β, and hence $E(\beta)$, such that $R(\alpha, \beta)$ is satisfied. For formal definitions and specifications see 2.

A second important scenario for non-malleability is in string commitment. Let A and B run a string commitment protocol. Assume that A is non-faulty, and that A commits to the string α. Assume that, concurrently, C and D are also running a commitment protocol in which C commits to a string β. If B and C are both faulty, then even though neither of these players know α, it is conceivable that β may depend on α. The goal of a non-malleable string commitment scheme is to prevent this.

We present a non-malleable string commitment scheme with the property that if the players have names (from a possibly unbounded universe), then for all relations R our scheme ensures that C is no more likely to be able to arrange that $R(\alpha, \beta)$ holds than it could do without access to the (A, B) interaction. Again, the scheme works even if A is unaware of the existence of C and D. If the players are anonymous, or the names they claim cannot be verified, then again if $\beta \neq \alpha$ then the two strings are no more likely to be related by R.

Intuitively, it is sufficient that C know the value to which it is committing in order to guarantee that α and β are unrelated. To see this, suppose C knows β and C also knows that $R(\alpha, \beta)$ holds. Then C knows "something about" α, thus violating the semantic security of string commitment. Proving possession of knowledge requires specifying a *knowledge extractor*, which, given the internal state of C, outputs β. In our case, the extractor has access to the (A, B) interaction, but it cannot rewind A. Otherwise it would only be a proof that *someone* knows β, but not necessarily that C does.

In the scenario we have been describing, there are (at least) two protocol executions involved: the (A, B) interaction and the (C, D) interaction. Even if both pairs of players are running string commitment protocols, the protocols need not be the same. Similar observations apply to the cases of non-malleable public-key cryptosystems and non-malleable zero-knowledge proofs of knowledge. Thus non-malleability of a protocol really only makes sense with respect to another protocol. All our non-malleable protocols are non-malleable with respect to themselves. A more general result is mentioned briefly in Section 5.

Using non-malleable string commitment as a building block, we obtain non-malleable zero-knowledge proofs of possession of knowledge, in the sense of Feige, Fiat, and Shamir [11]. Zero-knowledge protocols [18], may compose in an unexpectedly malleable fashion. A classic example is the so-called "Mafia scam" [8] attack on an identification scheme, similar in spirit to the transparent intermediary problem described above. Let A and D be non-faulty parties, and let B and C be cooperating faulty parties (they could even be the same party). Consider two zero-knowledge interactive proof systems, in one of which A is proving to B knowledge of some string α, and in the other C is proving to D knowledge of some string β. The two proof systems may be operating concurrently; since B and C are cooperating the executions of the (A, B) and (C, D) proof systems may not be independent. Intuitively, non-malleability says that if C can prove knowledge of β to D while A proves knowledge of α to B, then C could prove knowledge of β without access to the (A, B) interaction. We present a non-malleable scheme for zero-knowledge proof of possession of knowledge.

One delicate issue is the question of identities. Let α and β be as above. If the players have names, then our protocol guarantees that β is independent of α. The names may come from an unbounded universe. Note that there are many possibilities for names: timestamps, locations, message histories, and so on. If the players are anonymous, or the names they claim cannot be verified, then it is impossible to solve the *transparent prover* problem described earlier. However, the faulty prover must be completely transparent: if $\beta \neq \alpha$ then the two strings are unrelated by any relation R. In particular, recall the scenario described above in which (relatively unknown) Researcher A seeks credit for the $P \neq NP$ result and at the same time needs protection against the transparent prover attack. Instead of proving knowledge of a witness s that $P \neq NP$, Researcher A can prove knowledge of a statement $\alpha = A \cdot s$. In this case the only dependent statement provable by Professor B is α, which contains the name A.

We assume the existence of trapdoor functions in constructing our public-key cryptosystems. The string commitment protocols and zero-knowledge proofs require only one-way functions.

2 Definitions and System Model

Since non-malleability is a concept of interest in at least the three contexts of public key encryption, bit/string commitment, and zero-knowledge proofs, we give a single general definition that applies to all of these. Thus, when we speak of a *primitive* $\mathcal{P}$ we can instantiate any of these three primitives. For completeness, we give full definitions of each of these primitives in Section 2.2.

2.1 Definitions Specific to Non-Malleability

An *interactive* protocol $(A, B)[c, a, b]$ is an ordered pair of polynomial time probabilistic algorithms A and B to be run on a pair of interactive Turing machines with common input c and with private inputs a and b, respectively, where any of a, b, c might be null.

We distinguish between the algorithm A and the agent $\psi(A)$ that executes it. We also use $\psi(A)$ to denote a faulty agent that is "supposed" to be running A (that is, that the non-faulty participants expect it to be running A), but has deviated from the protocol.

In any interactive protocol (A, B) for primitive $\mathcal{P}$, party A has an *intended value*. In the case of encryption it is the value encrypted in $\psi(B)$'s public key; in string commitment it is the string to which $\psi(A)$ commits; in a zero-knowledge proof it is the theorem being proved interactively. We sometimes refer to the intended value as an *input* to A. We also sometimes refer to $\psi(A)$ as the *Sender* and to $\psi(B)$ as the *Receiver*. We use the verb *to send* to mean, as appropriate, to send an encrypted message, to commit to, and to prove knowledge of. Intuitively, in each of these cases information is being transmitted, or sent, from the Sender to the Receiver.

Interactive protocols (A, B), including the simple sending of an encrypted message, are executed in a context, and the participants have access to the history preceding the protocol execution. When $\psi(A)$ has intended value α, we assume both parties have access to $\mathrm{hist}(\alpha)$, intuitively, information about the history that leads to $\psi(A)$ running the protocol with intended value α.

In some cases we also assume an underlying probability distribution $\mathcal{D}$ on intended values, to which both parties have access (that is, from which they can sample in polynomial time).

An *adversarially coordinated* system of interactive protocols

$$\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$$

consists of two interactive protocols (A, B) and (C, D), an adversary $\mathcal{A}$ controlling the agents $\psi(B)$ and $\psi(C)$, the communication between these agents, and the times at which these controlled agents take steps.

Generally, we are interested in the situation in which $A = C$ and $B = D$, for example, when both interactive protocols are the same bit commitment protocol. Thus, for the remainder of the paper, unless otherwise specified, $(A, B) = (C, D)$, but $\psi(A), \psi(B), \psi(C), \psi(D)$ are all distinct.

Consider the adversarially coordinated system $\langle (A, B), (C, D), \ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$. In an execution of this system, $\psi(A)$ sends an intended value $\alpha \in_R \mathcal{D}$ in its conversation with $\psi(B)$, and $\psi(C)$ sends an intended value β in its conversation with $\psi(D)$. If $\psi(C)$ fails to do so we take β to be all zeros. Similarly, in the case of non-malleable encryption, we let α denote the plaintext of an encrypted message sent by $\psi(A)$ to $\psi(B)$, and we let β denote the plaintext of an encrypted message sent by $\psi(C)$ to $\psi(B)$. While we cannot prevent exact copying in any of the primitives we consider, the goal of non-malleability is to ensure that β be "unrelated" to α when $\alpha \neq \beta$.

A *relation approximator* R is a probabilistic polynomial time Turing machine taking two inputs[2] and producing as output either zero or one. The purpose of the relation approximator is to measure the correlation between α and β. That is, R measures how well the adversary manages to make β depend on α. We restrict our attention to the special class of relation approximators which on input pairs of the form (x, x) or $(x, 0^n)$ always output zero. The intuition here is that we cannot rule out exact copying or the case in which $\psi(C)$ refuses to send an intended value (or gets caught cheating), but intuitively these are not the cases in which the adversary "succeeds."

When we discuss composition we will extend the definition so that the first input is actually a vector V of length k. The intuition here is that C may have access to several interactions with, and intended values sent by, non-faulty players. In that case, the approximator must output zero on inputs (V, y) in which y is either a component of V, corresponding to the case in which $\psi(C)$ exactly copies one of the non-faulty players, or is of the form 0^n, again corresponding to the case in which $\psi(C)$ refuses to send an intended value.

Given a probability distribution on the pair of inputs, there is an *a priori* probability, taken over the choice of intended values and the coin flips of R, that R will output one. In order to measure the correlation between α and β we must compare R's behavior on input pairs (α, β) generated as described above to its behavior on pairs (α, γ), where γ is sent without access to the sending of α (although as always we assume that $\psi(C)$ has access to $\mathcal{D}$ and $\text{hist}(\alpha)$).

An *adversary simulator* for a commitment (zero-knowledge proof of knowledge) scheme $\mathcal{S}$ with input distribution $\mathcal{D}$ and polynomial time computable function hist, is a probabilistic polynomial time algorithm that, given hist, $\text{hist}(\alpha)$, and $\mathcal{D}$, produces an intended value γ.

Given an adversarially coordinated system of interactive protocols $\langle (A, B), (C, D), \ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ where (A, B) and (C, D) are both instances of $\mathcal{S}$, and given a relation approximator R, let $\pi(\mathcal{A}, R)$ denote the probability, taken over all choices of $\psi(A)$, $\psi(D)$, $\mathcal{A}$, and R, that $R(\alpha, \beta)$ outputs 1, where α is sent by $\psi(A)$ and β is sent by $\psi(C)$. Similarly,

[2]Sometime we will need R to take three inputs, the third being in plaintext

for an adversary simulator $\mathcal{A}'$ we let $\pi'(\mathcal{A}', R)$ denote the probability, taken over the choice of $\alpha \in \mathcal{D}$, the choices of $\mathcal{A}'$ in generating γ, and the choices of R, that $R(\alpha, \gamma) = 1$.

A scheme $\mathcal{S}$ for a primitive $\mathcal{P}$ is *non-malleable* with respect to itself if for all relation approximators R and all adversarially coordinated systems of interactive proof systems $\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ where $(A, B) = (C, D) = \mathcal{S}$, there exists an adversary simulator $\mathcal{A}'$ such that $|\, \pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\, |$ is subpolynomial.

The definition of a non-malleable cryptosystem is in exactly the same spirit as the definition for non-malleable interactive protocols. The details are postponed until Section 3.

2.2 Definitions of Primitives

In this section we review the definitions from the literature of probabilistic public key cryptosystems, string commitment, and non-interactive zero-knowledge proof systems, all of which are used as primitives in our constructions.

Probabilistic Public Key Encryption

A *probabilistic public key encryption* scheme (see [17]) consists of:

- GP, the *key generator*. A probabilistic machine that on input n, the security parameter, outputs a pair of strings (e, d) (e is the *public key* and d is the *secret key*)
- E, the encryption function, gets three inputs: the public key e, $b \in \{0, 1\}$, and a random string r of length $p(n)$, for some polynomial p. $E_e(b, r)$ is computable in polynomial time.
- D, the decryption function, gets two inputs: c which is a ciphertext and the private key d which was produced by GP. $D_d(c)$ is computable in expected polynomial time.
- if GP outputs (e, d), then

$$\forall b \in \{0, 1\}\ \forall r \in \{0, 1\}^{p(n)}\ \ D_d(E_e(b, r)) = b$$

- The system has the property of *indistinguishability*: for all polynomial time machines M

$$|Prob[M(e, E_e(0, r)) = 1] - Prob[M(e, E_e(1, r)) = 1]| < \frac{1}{poly(n)}$$

  where the probability is taken over the coin flips of GP, M and the choice of r.

For implementations of probabilistic encryption see [1, 5, 15, 22, 30]. In particular, such schemes can be constructed from trapdoor permutations.

Our version of semantic security is the following: Let R be a relation. We define two probabilities. Let $\mathcal{A}$ be an adversary that gets a key e and produces a distribution $\mathcal{M}$ on messages of length $\ell(n)$ (by producing a description of a polynomial time machine that generates $\mathcal{M}$). $\mathcal{A}$ is then given a *challenge* consisting of a ciphertext $c \in_R E_e(m)$, where $m \in_R \mathcal{M}$. In addition, $\mathcal{A}$ receives a "hint" about m in the form of hist(m), where hist is a polynomially computable function. $\mathcal{A}$ then produces a value β (β can be a single element of $\mathcal{M}$ or a vector of elements in $\mathcal{M}$). $\mathcal{A}$ is considered to have succeeded with respect to R if $R(m, \beta)$. Let $\pi(\mathcal{A}, R)$ be the probability that $\mathcal{A}$ succeeds with respect to R. The probability is over the choice of e, the flips of $\mathcal{A}$, and the choice of m, so in particular it is also over the choice of $\mathcal{M}$.

For the second probability, we have an adversary simulator $\mathcal{A}'$ who will not have access to the encryption, but has, however, the same computational power as $\mathcal{A}$. On input e, $\mathcal{A}'$ chooses a distribution $\mathcal{M}'$. Choose an $m \in_R \mathcal{M}'$ and give hist(m) to $\mathcal{A}'$. $\mathcal{A}'$ produces β. As above, $\mathcal{A}'$ is considered to have succeeded with respect to R if $R(m, \beta)$. Let $\pi'(\mathcal{A}', R)$ be the probability that $\mathcal{A}'$ succeeds.

A scheme $\mathcal{S}$ for public-key cryptosystems is *semantically secure with respect to relations* if for every relation $R(\alpha, \beta)$ and function hist(m) computable in probabilistic polynomial time and all probabilistic polynomial time adversaries $\mathcal{A}$ as above there exists a probabilistic polynomial time adversary simulator $\mathcal{A}'$ such that $|\,\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,|$ is subpolynomial.

Theorem 2.1 *A public key cryptosystem is semantically secure with respect to relations if and only if it has the indistinguishability property.*

Proof. Consider the following three experiments. Choose an encryption key e using GP. Given the public-key e, $\mathcal{A}$ produces a distribution $\mathcal{M}$. Sample $\alpha_1, \alpha_2 \in_R M$.

In the first experiment, $\mathcal{A}$ is given hist(α_1) and $E_e(\alpha_1)$ and produces β_1. Note that

$$\Pr[R(\alpha_1, \beta_1) \text{ holds}] = \pi(\mathcal{A}, R).$$

In the second experiment, $\mathcal{A}$ is given hist(α_1) and $E_e(\alpha_2)$ and produces β_2. Let

$$\chi = \Pr[R(\alpha_1, \beta_2) \text{ holds}].$$

Note that if $\pi(\mathcal{A}, R)$ and χ differ polynomially, then we have a distinguisher for encryptions of α_1 and α_2 (which may be converted via a hybrid argument into a distinguisher for encryptions of 0 and 1).

For the third experiment, consider an $\mathcal{A}'$ that generates an e using GP and simulates $\mathcal{A}$ on e to get a distribution $\mathcal{M}$. It gives $\mathcal{M}$ as the distribution on which it should be tested. $\mathcal{A}'$ is then given hist(α) for an $\alpha \in_R \mathcal{M}$. $\mathcal{A}'$ generates $\alpha' \in_R \mathcal{M}$ and gives to the simulated $\mathcal{A}$ the hint hist(α) and the encryption $E_e(\alpha')$. $\mathcal{A}$ responds with some β, which is then output by $\mathcal{A}'$. Note that $\pi'(\mathcal{A}', R) = \chi$. Thus, if the cryptosystem has the indistinguishability property then $|\,\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,|$ is subpolynomial, so the cryptosystem is semantically secure with respect to relations.

We now argue that if a cryptosystem does not have the indistinguishability property then it is not semantically secure with respect to relations. If a system does not have the indistinguishability property then there exists a polynomial time machine M that distinguishes encryptions of 0 from encryptions of 1 on a polynomial fraction of the encryption keys. Given a key e, define a machine $\mathcal{A}$ that chooses $\mathcal{M} = \{0, 1\}$, where each of 0 and 1 has probability $1/2$, as the message distribution on which it is to be tested. The function hist is the trivial hist$(x) = 1$ for all x. Given an encryption $\gamma \in_R E_e(m)$, where $m \in_R \mathcal{M}$, $\mathcal{A}$ uses M to guess the value of m and outputs β, the resulting guess. The relation R that witnesses the fact that the cryptosystem is not semantically secure with respect to relations is equality. Since M is by assumption a distinguisher, having access to the ciphertext gives $\mathcal{A}$ a polynomial advantage at succeeding with respect to R over any $\mathcal{A}'$ that does not have access to the ciphertext. $\square$

Thus, a scheme is semantically secure with respect to relations if and only if it has the indistinguishability property. It follows from the results in [17, 13, 23] that the notions of of semantic security, indistinguishability and semantically secure with respect to relations are all equivalent.

String Commitment

A *string commitment* protocol between sender A and receiver B consists of two stages:

- The *commit* stage: A has a string α to which she wishes to commit to B. She and B exchange messages. At the end of the stage B has some information that represents α.
- The *reveal* stage: at the end of this stage B knows α. There should be only one string that A can reveal.

The input α is drawn from a polynomial time sampleable distribution $\mathcal{D}$. The distribution $\mathcal{D}$, a polynomial time computable function hist, and the value hist(α) are known to both players. The function hist models information about the Sender's input to which the receiver may have access. At the end of the commit stage the representation of α should be semantically secure (see [13] for exact definition).

Non-Interactive Zero-Knowledge Proof Systems

The following explanation is taken almost verbatim from [26]: A (single theorem) non-interactive proof system for a language L allows one party $\mathcal{P}$ to prove membership in L to another party $\mathcal{V}$ for any $x \in L$. $\mathcal{P}$ and $\mathcal{V}$ initially share a string U of length polynomial in the security parameter n. To prove membership of a string x in $L_n = L \cap \{0,1\}^n$, $\mathcal{P}$ sends a message p as a proof of membership. $\mathcal{V}$ decides whether to accept or to reject the proof. Non-interactive zero knowledge proof systems were introduced in [3, 4]. A scheme for any language in NP and may be based on *any* trapdoor permutation is given in [12]. Recently, Kilian and Petrank [20, 21] found more efficient implementations of such schemes. Their scheme is for the circuit satisfiability problem. Assuming a trapdoor permutation on k bits, the length of a proof of a satisfiable circuit of size L (and the size of the shared random string) is $O(Lk^2)$.

The shared string U is generated according to some distribution $\mathcal{U}(n)$ that can be generated by a probabilistic polynomial time machine. (In all the examples we know of it is the uniform distribution, although this is not required for our scheme.)

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

Let $\mathcal{P}(x, z, U)$ be the distribution of the proofs generated by P on input x, witness z, and shared string U. Suppose that $\mathcal{P}$ sends $\mathcal{V}$ a proof p when the shared random string is U. Then the pair (U, p) is called the conversation. Any $x \in L$ and $z \in WL(x)$ induces a probability distribution $\mathcal{CONV}(x, z)$ on conversations (U, p) where $U \in \mathcal{U}$ is a shared string and $p \in \mathcal{P}(x, z, U)$ is a proof.

For the system to be zero-knowledge, there must exist a simulator Sim which, on input x, generates a conversation (U, p). Let $Sim(x)$ be the distribution on the conversations that Sim generates on input x, let $Sim_U(x) = Sim_U$ be the distribution on the U part of the conversation, and let $Sim_P(x)$ be the distribution on the proof component. In the definitions of [4, 12] the simulator has two steps: it first outputs Sim_U without knowing x, and then, given x it outputs $Sim_P(x)$. (This requirement, that the simulator not know the theorem when producing U, is not essential for our purposes, however, for convenience our proof in Section 3.3 does assume that the simulator is of this nature.)

Let

$$ACCEPT(U, x) = \{p | \mathcal{V} \text{ accepts on input } R, x, p\}$$

and let

$$REJECT(U, x) = \{p | \mathcal{V} \text{ rejects on input } U, x, p\}.$$

The following is the definition of non-interactive proof systems of [3], modified to incorporate the tractability of $\mathcal{P}$. The uniformity conditions of the system are adopted from Goldreich [13].

Definition 2.1 *A triple $(\mathcal{P}, \mathcal{V}, \mathcal{U})$, where $\mathcal{P}$ is a probabilistic machine, $\mathcal{V}$ is a polynomial time machine, and $\mathcal{U}$ is a polynomial time sampleable probability distribution is a* non-interactive zero-knowledge proof system *for the language $L \in NP$ if:*

1. Completeness *(if $x \in L$ then $\mathcal{P}$ generates a proof that $\mathcal{V}$ accepts): For all $x \in L_n$, for all $z \in WL(x)$, with overwhelming probability for $U \in_R \mathcal{U}(n)$ and $p \in_R \mathcal{P}(x, z, U)$, $p \in ACCEPT(U, x)$. The probability is over the choice of the shared string U and the internal coin flips of $\mathcal{P}$.*

2. Soundness *(if $y \notin L$ then no prover can generate a proof that $\mathcal{V}$ accepts): For all $y \notin L_n$ with overwhelming probability over $U \in_R \mathcal{U}(n)$ for all $p \in \{0,1\}^*$, $p \in REJECT(U, y)$. The probability is over the choices of the shared string U.*

3. Zero-knowledge *(there is a probabilistic polynomial time machine Sim which is a simulator for the system): For all probabilistic polynomial time machines $\mathcal{C}$, if $\mathcal{C}$ generates $x \in L$ and $z \in WL(x)$ then,*

$$|Prob[\mathcal{C}(w) = 1 | w \in_R Sim(x)] - Prob[\mathcal{C}(w) = 1 | w \in_R \mathcal{CONV}(x, z)]| < \frac{1}{p(n)}$$

for all polynomials p and sufficiently large n.

2.3 System Model

We *do not* assume the usual model of a fixed number of mutually aware processors. Rather, we assume a more general model in which a given party does not know which other parties are currently using the system. For example, consider a number of interconnected computers. A user ("agent") can log into any machine and communicate with a user on an adjacent machine, without knowing whether a given third machine is actually in use at all, or if the second and third machines are currently in communication with each other. In addition, the user does not know the set of potential other users, nor need it know anything about the network topology.

Thus, we do not assume a given user knows the identities of the other users of the system. On the other hand, our protocols may make heavy use of user identities. One difficulty is that in general, one user may be able to impersonate another. There are several ways of avoiding this. For example, Rackoff and Simon [27] propose a model in which each sender possesses a secret associated with a *publicly known* identifying key.

In the scenario of interconnected computers described above, an identity could be composed of the computer serial number and a timestamp, possibly with the addition of the claimed name of the user. In the absence of some way of verifying claimed identities, *exact copying* of the pair, claimed identity and text, cannot be avoided, but we rule out essentially all other types of dependence between intended values.

We can therefore assume that the intended value α sent by $\psi(A)$ contains as its first component a user identity, which may or may not be verifiable. Fix a scheme $\mathcal{S}$ and an adversarially coordinated system of interactive protocols $\langle (A,B),(C,D),\,\mathcal{A}:\psi(B)\leftrightarrow\psi(C)\rangle$ where (A,B) and (C,D) are both instances of $\mathcal{S}$, and let α and β be sent by $\psi(A)$ and $\psi(C)$, respectively. Then, whether or not the identities can be checked, if $\alpha\neq\beta$ then β's dependence on α is limited to dependence on $\mathrm{hist}(\alpha)$. In addition, if the identities can be checked then $\alpha\neq\beta$.

In order to avoid assumptions about the lengths of intended values sent, we assume the space of legal values is prefix-free.

3 Non-Malleable Public Key Cryptosystems

A *public-key cryptosystem* allows one participant, the owner, to publish a public key, keeping secret a corresponding secret key. Any user that knows the public key can use it to send messages to the owner; no one but the owner should be able to read them. In this section we show how to construct non-malleable public key cryptosystems. The definitions apply, *mutatis mutandi*, to private key cryptosystems. When defining the security of a cryptosystem one must specify (a) the type of attack considered and (b) what it means to break the cryptosystem. For technical reasons we extend the relations R to have three components, one of which is itself a vector.

The cryptosystem we construct is secure against chosen ciphertext attacks. In fact it is secure against a more severe attack suggested by Rackoff and Simon [27]: The attacker knows the ciphertext she wishes to crack while she is allowed to experiment with the decryption mechanism. She is allowed to feed it with any ciphertext she wishes, except for the exact one she is interested in. Thus the attacker is like a student who steals a test and can ask the professor any question, except the ones on the test. This is the first public key cryptosystem to be provably secure against such attacks. In contrast, RSA [28] and the implementation of probabilistic encryption based on quadratic residuousity [17] are *malleable* (*i.e.,* not non-malleable).

Malleability specifies what it means to "break" the cryptosystem. Informally, given a relation R (restricted as in Section 2.1) and a ciphertext of a message α, the attacker $\mathcal{A}$ is considered successful if it creates a *ciphertext* of β such that $R(\alpha,\beta)=1$. The cryptosystem is non-malleable if for every $\mathcal{A}$ there is an $\mathcal{A}'$ that, without access to the ciphertext of α succeeds with similar probability as $\mathcal{A}$ in creating a ciphertext of γ such that $R(\alpha,\gamma)=1$. Non-malleability is clearly an extension of semantic security.

We now define precisely the power of the adversary $\mathcal{A}$. Let R be a polynomial time computable relation. Let n be the security parameter. $\mathcal{A}$ receives the public key $e \in_R GP(n)$ and can adaptively choose a sequence of ciphertexts $c_1, c_2, \ldots$. On each of them $\mathcal{A}$ gets the corresponding plaintext. It then produces a distribution $\mathcal{M}$ on messages of length $\ell(n)$, for some polynomial ℓ, by giving the polynomial time machine that can generate this distribution. $\mathcal{A}$ then receives as a challenge a ciphertext $c \in_R e(m)$ where $m \in_R \mathcal{M}$, together with some "side-information" about m in the form of $\text{hist}(m)$, where hist is some polynomially computable function. $\mathcal{A}$ then engages in a second sequence of adaptively choosing ciphertexts $c'_1, c'_2, \ldots$. The only restriction is that $\mathcal{A}$ cannot ask on c itself. At the end of the process, $\mathcal{A}$ produces a polynomial bounded length vector of ciphertexts $(f_1, f_2, \ldots)$, with each $f_i \in e(\beta_i)$, and a *cleartext* string σ[3]. Let $\beta = (\beta_1, \beta_2, \ldots)$. $\mathcal{A}$ is considered to have succeeded with respect to R if $R(m, \beta, \sigma)$. Let $\pi(\mathcal{A}, R)$ be the probability that $\mathcal{A}$ succeeds.

Let $\mathcal{A}'$ be an adversary simulator that does not have access to the encryptions or to the decryptions, but has, however, the same computational power as $\mathcal{A}$ and can pick the distribution $\mathcal{M}$. On input e, and additional information $\text{hist}(m)$, but without the benefit of the chosen ciphertext attack, $\mathcal{A}'$ produces a vector of ciphertexts $(f_1, f_2, \ldots)$, where each $f_i \in e(\beta_i)$, and a string σ. Let $\beta = (\beta_1, \beta_2, \ldots)$. As above, $\mathcal{A}'$ is considered to have succeeded with respect to R if $R(m, \beta, \sigma)$. Let $\pi'(\mathcal{A}', R)$ be the probability that $\mathcal{A}'$ succeeds.

A scheme $\mathcal{S}$ for public-key cryptosystems is *non-malleable* if for all relations $R(\alpha, \beta, \sigma)$ computable in probabilistic polynomial time and all probabilistic polynomial time adversaries $\mathcal{A}$ as above there exists a probabilistic polynomial time adversary simulator $\mathcal{A}'$ such that $|\,\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,|$ is subpolynomial.

Overview of the scheme

The process of encryption consists of 4 parts. An "identity" is chosen for the message by creating a public signature key. The message is encrypted under several encryption keys chosen from a set of such keys as a function of the public signature key chosen in the first step. A (non-interactive zero-knowledge) proof of consistency is provided; *i.e*, that the encryption under all keys is the same. Finally the encryptions plus the proof are signed using the private key of the signature chosen in the first step.

Thus, the public key consists of 3 parts: a collection of n pairs of keys $\langle e_i^0,\ e_i^1 \rangle$, a random string U for providing zero-knowledge proofs of consistency in a non-interactive proof system, and a universal one-way hash function providing a mapping that defines a choice of a subset of the encryption keys. When a message is decrypted it must first be verified that the encryptions are consistent, and only then is the (now well defined) plaintext extracted.

Non-malleability comes from the fact that the choice of the subsets and the signature each authenticate the other. Moreover, as in [26], anyone can decide whether a ciphertext is legitimate, *i.e.,* decrypts to some meaningful message. Finally, whenever deciding whether or not a string represents a legitimate ciphertext is easy, non-malleability implies security against a Rackoff-Simon attack. The intuition is that given $E(\alpha)$ an attacker with access

[3] In the public key context σ serves no purpose, as in this situation from σ it is always possible to compute an encryption of σ, so we could always add an additional $f_i \in e(\sigma)$ to our vector of ciphertexts. However, we introduce σ so that the definition can apply to symmetric, or private key, encryption.

to a decryption mechanism can generate a legal ciphertext $E(\beta)$ and learn β, but non-malleability implies that an adversary simulator can generate γ without access to $E(\alpha)$ that is distributed essentially as β is distributed.

3.1 The Tools

We require a probabilistic public key cryptosystem as described in Section 2.2. Recall that GP denotes the key generator, e and d denote the public and private keys, respectively, and E and D denote, respectively, the encryption and decryption algorithms.

For public keys $e_1, e_2, \ldots e_n$ a *consistent encryption* is a string w that is equal to

$$E_{e_1}(b, r_1), E_{e_2}(b, r_2), \ldots, E_{e_n}(b, r_n)$$

for some $b \in \{0,1\}$ and $r_1, r_2, \ldots, r_n \in \{0,1\}^{p(n)}$, for some polynomial p. The language of consistent encryptions $L = \{e_1, e_2, \ldots e_n, w | w \text{ is a consistent encryption}\}$ is in NP. For a given word $w = E_{e_1}(b, r_1),\, E_{e_2}(b, r_2),\, \ldots, E_{e_n}(b, r_n)$ the sequence $r_1,\ r_2, \ldots, r_n$ is a witness for its membership in L. In order to prove consistency we need a *non-interactive zero-knowledge proof system* for L, as defined in Section 2.2. Recall that the system consists of a prover, a verifier, and a common random string U known to both the prover and the verifier.

The cryptosystem uses a *universal family of one-way hash functions* as defined in [25]. This is a family of functions H such that for any x and a randomly chosen $h \in_R H$ the problem of finding $y \neq x$ such that $h(y) = h(x)$ is intractable. The family we need should compress from any polynomial in n bits to n bits. In [29] such families are constructed from any one-way function.

Finally we need a one-time *signature scheme*, which consists of GS, the scheme generator that outputs F, the public-key of the signature scheme, and P the private key. Using the private key P any message can be signed in such a way that anyone knowing F can verify the signature and no one who does not know the private key P can generate a valid signature on any message except the one signed. For exact definition and history see [2, 19, 25].

3.2 The Non-Malleable Public Key Encryption Scheme

We are now ready to present the scheme $\mathcal{S}$.

Key generation:

1. Run $GP(n)$, the probabilistic encryption key generator, $2n$ times. Denote the output by
$$(e_1^0, d_1^0), (e_1^1, d_1^1), (e_2^0, d_2^0), (e_2^1, d_2^1), \ldots (e_n^0, d_n^0), (e_n^1, d_n^1).$$

2. Generate random U.

3. Generate $h \in_R H$.

4. The public encryption key is $\langle h, e_1^0, e_1^1, e_2^0, e_2^1, \ldots, e_n^0, e_n^1, U\rangle$. The corresponding private decryption key is $\langle d_1^0, d_1^1, d_2^0, d_2^1, \ldots d_n^0, d_n^1\rangle$.

Encryption: To encrypt a message $m = b_1, b_2, \ldots b_k$:

1. Run $GS(n)$, the signature key generator. Let F be the public signature key and P be the private signature key.
2. Compute $h(F)$. Denote the output by $v_1 v_2 \ldots v_n$.
3. For each $1 \le i \le k$
   (a) For $1 \le j \le n$
      i. generate random $r_{ij} \in_R \{0,1\}^{p(n)}$
      ii. generate $c_{ij} = E_{e_j^{v_j}}(b_i, r_{ij})$, an encryption of b_i using $e_j^{v_j}$.
   (b) Run $\mathcal{P}$ on $c_i = e_1^{v_1}, e_2^{v_2}, \ldots, e_n^{v_n}, c_{i1}, c_{i2}, \ldots c_{in}$, with witness $r_{i1}, r_{i2}, \ldots, r_{in}$ and string U to get a proof p_i that $c_i \in L$.
4. Create a signature s of the sequence (c_1, p_1), (c_2, p_2), $\ldots, (c_k, p_k)$ using the private signature key P.

The encrypted message is

$$\langle F, s, (c_1, p_1), (c_2, p_2) \ldots (c_k, p_k) \rangle.$$

Decryption: to decrypt a ciphertext $\langle F, s, (c_1, p_1), (c_2, p_2), \ldots, (c_k, p_k) \rangle$:

1. Verify that s is a signature of (c_1, p_1),(c_2, p_2),$\ldots, (c_k, p_k)$ with public signature key F.
2. For all $1 \le i \le k$ verify that c_i is consistent by running the verifier $\mathcal{V}$ on c_i, p_i, U.
3. Compute $h(F)$. Denote the output by $v_1 v_2 \ldots v_n$.
4. If $\mathcal{V}$ accepts in all k cases, then for all $1 \le i \le k$ retrieve b_i by decrypting using any one of $\langle d_1^{v_1}, d_2^{v_2}, \ldots, d_n^{v_n} \rangle$. Otherwise the output is null.

Note that, by the proof of consistency, the decryptions according to the different keys in Step 4 are identical with overwhelming probability.

From this description it is clear that the generator and the encryption and decryption mechanisms can be operated in polynomial time. Also if the decryption mechanism is given a legitimate ciphertext and the right key it produces the message encrypted.

3.3 Non-Malleable Security

We now prove the non-malleability of the public key encryption scheme $\mathcal{S}$. The security of the system rests on the semantic security against chosen *plaintext* attacks of the following cryptosystem $\mathcal{S}'$ (this corresponds to system S' in Section 4).
The Cryptosystem $\mathcal{S}'$:

1. Run $GP(n)$, the probabilistic encryption key generator, n times. Denote the output by
$$(e_1, d_1), (e_2, d_2), \ldots (e_n, d_n).$$
   The public key is the n-tuple $\langle e_1, \ldots, e_n \rangle$; the private key is the n-tuple $\langle d_1, \ldots, d_n \rangle$.

2. To encrypt a message $m = b_1, b_2, \dots b_k$

3. For $1 \leq j \leq n$

   - For $1 \leq i \leq k$

     (a) generate random $r_{ij} \in_R \{0,1\}^{p(n)}$

     (b) generate $c_{ij} = E_{e_j}(b_i, r_{ij})$, an encryption of b_i under public key e_j using random string r_{ij}.

   - Let $c_j = c_{1j}, c_{2j}, \dots, c_{kj}$ (c_j is the jth encryption of m).

4. The encryption is the n-tuple $\langle c_1, c_2, \dots, c_n \rangle$.

5. To decrypt an encryption $\langle \alpha_1, \dots, \alpha_n \rangle$, compute $m_j = D_{d_j}(\alpha_j)$ for $1 \leq j \leq n$. If $m_1 = m_2 = \dots = m_n$ then output m_1; otherwise output "invalid encryption."

Lemma 3.1 *The public key encryption scheme $\mathcal{S}'$ is semantically secure with respect to relations under chosen plaintext attack.* □

We will prove non-malleability of $\mathcal{S}$ by reduction to the semantic security of $\mathcal{S}'$. To this end, we define an adversary $\mathcal{A}'$ that, on being given an encryption under $\mathcal{S}'$, generates an encryption under $\mathcal{S}$. We abuse notation slightly: given a public key E in $\mathcal{S}$ (respectively, E' in $\mathcal{S}'$), we let $E(m)$ (respectively, $E'(m)$) denote the set of encryptions of m obtained using the encryption algorithm for $\mathcal{S}$ (respectively, for $\mathcal{S}'$) with public key E (respectively, E').

Procedure for $\mathcal{A}'$: Given a public key $E' = \langle e_1, \dots, e_n \rangle$ in $\mathcal{S}'$:
Preprocessing Phase:

1. Generate n new (e, d) pairs.

2. Run the simulator for the non-interactive zero-knowledge proof of consistency to generate a random string U (the simulator should be able to produce polynomially many proofs of consistency for sets of n encryptions that will be given to it later on).

3. Choose a random hash function $h \in_R H$.

4. Run $GS(n)$ to obtain a signature scheme (F, P), where F is the public verification key.

5. Compute $h(F)$. Arrange the original n keys and the n new keys so that the keys "chosen" by $h(F)$ are the original n. Let E denote the resulting public key (instance of $\mathcal{S}$).

Simulation Phase:

1. Run $\mathcal{A}$ on input E. $\mathcal{A}$ adaptively produces a polynomial length sequence of encryptions $x_1, x_2, \dots$. For each x_i produced by $\mathcal{A}$, $\mathcal{A}'$ verifies the signatures and the proofs of consistency. If these verifications succeed, $\mathcal{A}'$ decrypts x_i by using one of the new decryption keys generated in Step 1, and returns the plaintext to $\mathcal{A}$.

2. $\mathcal{A}$ produces a description of $\mathcal{M}$, the distribution of messages it would like to attack. $\mathcal{A}'$ outputs $\mathcal{M}$. We will show that if $\mathcal{S}$ is malleable then $\mathcal{S}'$ is not semantically secure with respect to relations on $\mathcal{M}$.

3. $\mathcal{A}'$ is given $c' \in_R E'(m)$ and hist(m) for $m \in_R \mathcal{M}$. It produces a ciphertext $c \in E(m)$ using the simulator of Step 2 of the preprocessing phase to obtain a proof of consistency and the private key P generated at step 5 to obtain the signature.

4. Give $\mathcal{A}$ the ciphertext c and hist(m). As in Step 1, $\mathcal{A}$ adaptively produces a sequence of encryptions $x'_1, x'_2, \ldots$ and $\mathcal{A}'$ verifies their validity, decrypts and returns the plaintext to $\mathcal{A}$.

Extraction Phase:
$\mathcal{A}$ produces the vector of encryptions $(E(\beta_1), E(\beta_2), \ldots)$. $\mathcal{A}'$ produces $\beta = (\beta_1, \beta_2, \ldots)$ by decrypting each $E(\beta_i)$ as in the simulation phase. $\mathcal{A}'$ outputs β and σ.

This concludes the description of $\mathcal{A}'$.

The next lemma shows that the simulation phase runs smoothly.

Lemma 3.2 *Let $\mathcal{A}$ be an adversary attacking the original scheme $\mathcal{S}$. On input $c' \in_R E'(m)$, let E be generated by $\mathcal{A}'$ as above, and let c be the encryption of m under E. Let $\zeta \neq c$ be any ciphertext under E, generated by $\mathcal{A}$. If the signatures in ζ are valid (can be verified with the public signature key in ζ), then $\mathcal{A}'$ can decrypt ζ.*

Proof. Let F' be the public signature key in ζ. If $F' \neq F$ then, by the security of the universal one-way hash functions, $h(F') \neq h(F)$ (otherwise using $\mathcal{A}$ one could break H). Thus, at least one of the encryption keys generated in Step 1 of the procedure for $\mathcal{A}'$ will be used in ζ. Since $\mathcal{A}'$ generated this encryption key and its corresponding decryption key, $\mathcal{A}'$ can decrypt.

If $F' = F$ then by the security of the signature scheme, only the original ciphertext c and the proof of consistency of Step 2 can be signed. (Otherwise $\mathcal{A}$ could be used to break the signature scheme.) $\square$

Lemma 3.3 *For any relation approximator R (restricted as in Section 2.1), let $\pi'(\mathcal{A}', R)$ denote the probability that $\mathcal{A}$ in the simulation breaks the generated instance of $\mathcal{S}$ with respect to R; i.e., that $\mathcal{A}$ (interacting with $\mathcal{A}'$, as described in the Simulation Phase of the Procedure for $\mathcal{A}'$) generates a vector of encryptions $(E(\beta_1), E(\beta_2), \ldots)$ and a string σ such that $R(m, \beta, \sigma)$ holds, where $\beta = (\beta_1, \beta_2, \ldots)$.*

Similarly, let $\pi(\mathcal{A}, R)$ denote the probability that $\mathcal{A}$ breaks a random instance of $\mathcal{S}$ with respect to R. Then $\pi'(\mathcal{A}', R)$ and $\pi(\mathcal{A}, R)$ are subpolynomially close.

Proof. The only difference between the instance of $\mathcal{S}$ generated by $\mathcal{A}'$ and the instance of $\mathcal{S}$ generated at random is in the proofs of consistency: in the former case these are produced by the simulator and in the latter case they are authentic. The lemma therefore follows immediately from the definition of non-interactive zero knowledge: any difference between the two probabilities can be translated via a hybrid argument into an ability to distinguish a simulated proof from a true proof. $\square$

Theorem 3.4 *The scheme $\mathcal{S}$ is a non-malleable public key encryption scheme.*

Proof. Let R be any relation approximator, restricted as in Section 2.1, and let $\mathcal{A}$ be any polynomially bounded adversary.

Beginning with an encryption key E' in $\mathcal{S}'$ and a ciphertext $c' = E'(m)$, $\mathcal{A}'$ generates an encryption key E in $\mathcal{S}$ and a ciphertext $c = E(m)$ and presents E and c to $\mathcal{A}$. If $\mathcal{A}$ produces valid encryptions $E(\beta_i)$ such that $E(\beta_i) \neq E(m)$, then by Lemma 3.2, $\mathcal{A}'$ can extract the β_i. Let $\beta = (\beta_1, \beta_2, \ldots)$. By Lemma 3.3, the probability that $R(m, \beta, \sigma)$ holds is subpolynomially close to $\pi(\mathcal{A}, R)$. Moreover, from the semantic security of $\mathcal{S}'$ we know there exists a procedure $\mathcal{A}''$ that, without access to $E'(m)$, produces β'' such that the probability of $R(m, \beta'', \sigma)$ is subpolynomially close to the probability of $R(m, \beta, \sigma)$, and hence to $\pi(\mathcal{A}, R)$. Therefore $(\mathcal{A}, R)$ cannot witness the non-malleability of $\mathcal{S}$. $\square$

3.4 Public Key Authentication

In this section we informally describe a method for obtaining a public key authentication scheme based on any non-malleable public key cryptosystem. Our goal is to demonstrate a protocol that allows cheating in case the public-key cryptosystem used is malleable.

In a public key authentication scheme, an authenticator A chooses a public key E. The scheme permits A to *authenticate* a message m of her choice to a second party B. Similar to a digital signature scheme, an authentication scheme can convince B that A is willing to authenticate m. However, unlike the case with digital signatures, an authentication scheme need not permit B to convince a third party that A has authenticated m.

Our notion of security is analogous to that of *existential unforgeability under an adaptive chosen plaintext attack* for signature schemes [19], where we must make sure to take care of "man-in-the-middle" attacks. Let $\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ be an adversarially coordinated system in which $(A, B) = (C, D)$ is a public key authentication protocol. We assume that A is willing to authenticate any number of messages $m_1, m_2, \ldots$, which may be chosen adaptively by $\mathcal{A}$. We say that $\mathcal{A}$ successfully attacks the scheme if $\psi(C)$ (under control of $\mathcal{A}$ and pretending to have A's identity) succeeds in authenticating to D a message $m \neq m_i$, $i = 1, 2, \ldots$.

Protocol for A to authenticate message m to B; A's public key is E, chosen according to a non-malleable public key cryptosystem $\mathcal{S}$.

1. A sends to B: "A wishes to authenticate m." (This step is unnecessary if m has previously been determined.)

2. B chooses $r \in_R \{0,1\}^n$ and computes and sends the *query* $\gamma \in_R E(m, r)$ to A.

3. A decrypts γ and retrieves r and m. If the decryption is of the right format (i.e. the first component of the decrypted pair corresponds to the message that is to be authenticated), then A sends r to B.

Lemma 3.5 *Given an adversary $\mathcal{A}$ that can break the above authentication protocol with probability ρ, one can construct an adversary $\mathcal{A}'$ for breaking the non-malleable encryption scheme with probability at least $\rho/p(n) - 2^{-n}$ for some polynomial p.*

Proof. The procedure for $\mathcal{A}'$ to attack the cryptosystem is as follows. Assume A's public key is E and that the adversary $\mathcal{A}'$ has access to a decryption box for E. Therefore $\mathcal{A}'$ can simulate the system $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$. Run the system $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$ until $\psi(C)$, under control of $\mathcal{A}$, is about to authenticate to D a message $m \neq m_i$, $i = 1, 2\ldots$ not authenticated by A. (In case it is not clear whether D accepts or not, then we just guess when this occurs, therefore the polynomial degradation of ρ.) The distribution $\mathcal{M}$ on messages that $\mathcal{A}'$ will attempt to maul is $\mathcal{M}_m = \{(m,r) | r \in_R \{0,1\}^n\}$. Given γ as the challenge ciphertext, $\mathcal{A}'$ lets $\psi(D)$ send it as the query. Let r' be $\psi(C)$'s reply. Output $\theta \in_R E(m, r')$.

The distribution that $\mathcal{A}$ sees is exactly his usual, therefore the probability of success in authenticating m is ρ and with probability ρ the value r' is the correct one. The relation that is violated is equality: θ and γ encrypt the same string, whereas given the distribution $\mathcal{M}_m$ the probability of producing the correct r is 2^{-n}. □

Remark 3.6 *If the cryptosystem $\mathcal{S}$ is malleable and in particular if given an encryption of a message with prefix λ it is easy to generate an encryption of a message with prefix λ' (many cryptosystems have this property), then there is a simple attack on the protocol proposed: as before $\psi(C)$ is pretending to be A. To forge a message m, when D sends query γ (which by definition has m as the prefix of the corresponding plaintext) $\psi(B)$ asks A to authenticate a message m' and produce as a query γ' with the prefix changed to m'. When A replies with r, $\psi(C)$ sends r to D, who accepts.*

Remark 3.7 *As mentioned above, this protocol provides a weaker form of authentication than digital signatures (no third parties verification). However, this can be viewed as a feature: there may be situations in which a user does not wish to leave trace of the messages the user authenticated ("plausible deniability"). We do not know whether the protocol presented is indeed zero-knowledge in this sense, i.e. that the receiver could have simulated the conversation alone. However, it seems that by adding a (malleable) proof of knowledge to the string r this is remedied.*

This solution will be of practical use as soon as the current constructions of non-malleable cryptosystems are improved to be more practical.

4 A Non-Malleable Scheme for String Commitment

In this section we present a non-malleable scheme $\mathcal{S}$ for string commitment. We first present $\mathcal{S}$ and show some properties of $\mathcal{S}$ important in proving its security. We then describe a knowledge extractor algorithm that works not on $\mathcal{S}$ but on $\mathcal{S}'$ which is a (malleable) string commitment protocol with a very special relation to $\mathcal{S}$: we show that knowledge extraction for $\mathcal{S}'$ implies non-malleability of $\mathcal{S}$. Thus, in this section $\mathcal{S}'$ plays a role analogous to the role of $\mathcal{S}'$ in Section 3.

Our non-malleable scheme for string commitment requires as a building block a (possibly malleable) string commitment scheme. Such a scheme, based on pseudo-random generators, is presented in [24]. The protocol described there is interactive and requires two phases: first the receiver sends a string and then the sender actually commits. However, the first step of the protocol can be shared by all subsequent commitments. Thus, following the first commitment, we consider string commitment to be a one-phase procedure. In the sequel, when we refer to the string commitment in [24], we consider only the second stage of that protocol there.

We also require *zero-knowledge proofs* obeying the security requirements in [13]. These can be constructed from any bit commitment protocol.

Before we continue it would be instructive to consider the protocol of Chor and Rabin [7]. They considered the "usual" scenario, where all n parties know of one another and the communication is synchronous and proceeds in rounds. Their goal was for each party to prove to all other parties possession of knowledge of a decryption key. Every participant engages in a sequence of proofs of possession of knowledge. In some rounds the participant acts as a prover, proving the possession of knowledge of the decryption key, and in others it acts as a verifier. The sequence is arranged so that every pair of participants A, C is separated at least once, in the sense that there exists a round in which C is proving while A is not. This assures that C's proof is independent of A's proof.

Running this protocol in our scenario is impossible; for example, (1) we make no assumptions about synchrony of the different parties, and (2) in our scenario the parties involved do not know of one another. However, we achieve a similar effect to the technique of Chor and Rabin by designing a carefully ordered sequence of actions a player must make, as a function of an identifier composed of its external identity, if one exists, and some other information described below.

4.1 The Scheme $\mathcal{S}$

Protocol $\mathcal{S}$ consists of two general stages. The first is a string commitment as in [24]. The second stage, called Basic Commit with Knowledge, consists of the application of many instances of a new protocol, called **BCK**, to the string committed to in the first stage.

Following the commit stage of two string commitment protocols, deciding whether they encode the same string is in NP. Therefore there exists a zero-knowledge proof for equality of two committed values. This will be done repeatedly during each execution of **BCK**, which we now describe. In all the following protocols n is a security parameter.

Protocol BCK(α):

Concurrently run n instances of the following three steps. All instances of each step are performed at once.

- **BCK1**: Committer selects random $x_0, x_1 \in \{0,1\}^k$, where $k = |\alpha|$, and commits to both of them using the protocol in [24].
- **BCK2**: Receiver sends Committer a random bit $r \in \{0,1\}$.
- **BCK3**: Committer reveals x_r and $x_{1-r} \oplus \alpha$, and engages in a proof of consistency of $x_{1-r} \oplus \alpha$ with the initial commitment to α and the commitment to x_{1-r} in **BCK**1.

Note that given x_r, x_{1-r}, and the proof of consistency, one can obtain α.

In the rest of the paper we consider each **BCK**i as single operation, thus it can be viewed as an operation on an n-dimensional vector or array. We frequently refer to an instance of **BCK** as a *triple*.

(A, B) **interaction**	(C, D) **interaction**
$\mathbf{BCK}1(\alpha)$	
	$\mathbf{BCK}1(\beta)$
	$\mathbf{BCK}2(\beta)$
$\mathbf{BCK}2(\alpha)$	
$\mathbf{BCK}3(\alpha)$	
	$\mathbf{BCK}3(\beta)$

Figure 1: $\mathbf{BCK}(\alpha)$ is useful to $\mathbf{BCK}(\beta)$

In the Basic Commit with Knowledge stage of $\mathcal{S}$ we apply **BCK** repeatedly for the same string, α. However, **BCK** may itself be malleable. To see this, conceptually label the three steps of **BCK** as commitment, challenge, and response, respectively. Consider an $\langle (A, B), (C, D), \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ in which $(A, B) = (C, D) = \mathbf{BCK}$. Then $\psi(C)$ can make its commitment depend on the commitment of $\psi(A)$; $\psi(B)$ can make its challenge to $\psi(A)$ depend on the challenge that $\psi(D)$ poses to $\psi(C)$, and $\psi(C)$ can respond to the challenge with the "help" of $\psi(A)$'s response to $\psi(B)$ (see Figure 1 for the timing of events). In this case the triple between $\psi(A)$ and $\psi(B)$ is, intuitively, useful to $\psi(C)$. The Basic Commit with Knowledge stage of $\mathcal{S}$ interleaves executions of **BCK** so as to ensure that in every execution there is some triple that is *exposed* (defined precisely below) – that is, for which no other triple is useful.

The next two protocols perform a pair of distinct instances of $\mathbf{BCK}(\alpha)$ in two different interleaved orders. To distinguish between the two instances we will refer to the operation taking place at each stage and the associated variables. Thus α_i and α_{i+1} are two distinct applications of These Sixplet protocols will be used to ensure the existence of an exposed triple in the Basic Commit with Knowledge. The spacing of the presentation intends to clarify the difference between the protocols. It has no meaning with respect to the execution of the protocols.

0-**sixplet**	1-**sixplet**
$\mathbf{BCK}1(\alpha_i)$	$\mathbf{BCK}1(\alpha_i)$
$\mathbf{BCK}2(\alpha_i)$	
$\mathbf{BCK}3(\alpha_i)$	$\mathbf{BCK}1(\alpha_{i+1})$
	$\mathbf{BCK}2(\alpha_{i+1})$
$\mathbf{BCK}1(\alpha_{i+1})$	$\mathbf{BCK}3(\alpha_{i+1})$
$\mathbf{BCK}2(\alpha_{i+1})$	
$\mathbf{BCK}3(\alpha_{i+1})$	$\mathbf{BCK}2(\alpha_i)$
	$\mathbf{BCK}3(\alpha_i)$

The difference between the two protocols is the order in which we interleave the stages of the two distinct instances of the **BCK** protocol.

Using these sixplets we can present the scheme $\mathcal{S}$. The identifier I used in the scheme is the concatenation of the original identity with the commitment for α at stage 1 (by the "commitment" we mean a transcript of the conversation). I_j denotes the jth bit of I. To force an exposed triple we will use the fact that every two identifiers differ in at least one bit. This fact was exploited by Chor and Rabin in [7].

$\mathcal{S}$: Non-Malleable Commitment to String α:

- Commit to α using the protocol in [24].
- For $j = 1$ to $|I|$
  Execute an I_j-sixplet
  Execute a $(1 - I_j)$-sixplet
  End

For simplicity we will assume that all identifiers I are n bits long. Each I_j-sixplet and each $(1 - I_j)-$sixplet involves two executions of **BCK**, and each of these in turn requires n concurrent executions of **BCK**1, followed by n concurrent executions of **BCK**2 and then of **BCK**3. Thus, a non-malleable string commitment requires invoking each **BCK**i a total of $4n^2$ times.

We now show some properties of $\mathcal{S}$ that allow us to prove its non-malleability. Suppose that two pairs of sender and receiver (A, B) and (C, D) are conducting $\mathcal{S}$ and suppose further that adversary $\mathcal{A}$ controls B and C. Let x be the identifier used by $\psi(A)$ and y that used by $\psi(C)$. If the original identities of $\psi(A)$ and $\psi(C)$ are different or if the strings to which they commit are different, then $x \neq y$. (Thus the only case not covered is exact copying, for which nothing can be done.)

Each run of the two interactions determines specific times at which the two pairs of machines exchange messages. The adversary can influence these times, but the time at which an interaction takes place is well defined. Let σ_x and σ_y be the respective schedules. For $1 \leq i \leq 2n$, let

- τ_1^i be the time at which **BCK**1 begins in the ith instance of **BCK** in σ_x;
- τ_2^i be the time at which **BCK**1 ends in the ith instance of **BCK** in σ_x.

In contradistinction, let

- δ_1^i be the time at which **BCK**1 ends in the ith instance of **BCK** in σ_y;
- δ_2^i be the time at which **BCK**2 begins in the ith instance of **BCK**2 in σ_y.

Finally, let

- τ_3^i and δ_3^i denote the times at which **BCK**3 ends in the ith instances of **BCK** in σ_x and σ_y, respectively.

These values are well defined because each **BCK**i involves sequential operations of a single processor.

We can now formalize the intuition, described above, of what it means for a triple in σ_x to be useful to a triple in σ_y. Formally, the ith triple in σ_x is *useful* to the jth triple in σ_y if three conditions hold: (1) $\tau_1^i < \delta_1^i$; (2) $\delta_2^j < \tau_2^i$; and (3) $\delta_3^j > \tau_2^i$.

Let $\Gamma^{(i)} = \{j \mid \delta_1^j > \tau_1^i \ \wedge \ \delta_2^j < \tau_2^i \ \wedge \ \delta_3^j > \tau_2^i\}$. $\Gamma^{(i)}$ is the set of indices of triples between $\psi(C)$ and $\psi(D)$ for which the ith triple between $\psi(A)$ and $\psi(B)$ can be useful. We say that a triple j is *exposed* if $j \notin \Gamma^{(i)}$ for all i. Our goal is to show that there is at least one exposed triple in any schedule. Intuitively, exposed triples are important because the committer is forced to act on its own, without help from any other concurrent interaction. Technically, exposed triples are important because they allow the knowledge extractor to explore the adversary's response to two different queries, without the cooperation of $\psi(A)$.

Claim 4.1 $\forall i \quad |\Gamma^{(i)}| \leq 1$.

Proof. By inspection of the possible interleavings, there exists at most one j for which $\delta_2^j < \tau_2^i$ and $\delta_3^j > \tau_2^i$. $\square$

Claim 4.2 *If $j_1 \in \Gamma^{(i_1)}$ and $j_2 \in \Gamma^{(i_2)}$ and $j_1 < j_2$, then $sixplet(i_1) \leq sixplet(i_2)$, where $sixplet(i)$ denotes the index of the sixplet containing the ith triple.*

Proof. Assume to the contrary that $sixplet(i_2) < sixplet(i_1)$. This implies that $\tau_2^{i_2} < \tau_1^{i_1}$. By definition, $j_1 \in \Gamma^{(i_1)}$ implies $\tau_1^{i_1} < \delta_1^{j_1}$. Similarly, $j_2 \in \Gamma^{(i_2)}$ implies $\delta_1^{j_2} < \tau_2^{j_2}$. Thus, $\delta_1^{j_2} < \delta_1^{j_1}$. This contradicts the assumption that $j_1 < j_2$. $\square$

Claim 4.3 *Let triples $2k-1, 2k$ form a 0-sixplet in σ_x, and let triples $2\ell-1, 2\ell$ form a 1-sixplet in σ_y. Then there exists a $j \in \{2\ell-1, 2\ell\}$ such that neither $2k-1$ nor $2k$ is useful to triple j in σ_y.*

Proof. Assume to the contrary that the claim does not hold. Thus, both triples have a useful triple in $\{2k-1, 2k\}$. By Claim 4.1 $|\Gamma^{(2k-1)}| \leq 1$, and $|\Gamma^{(2k)}| \leq 1$. Therefore, each of the two triples should be useful. A simple look at the time scale implies that for either matching between the pairs, it should be the case that $\tau_1^{2k} < \delta_1^{2\ell}$. Thus, $\tau_2^{2k-1} < \delta_2^{2\ell}$ and $\tau_2^{2k-1} < \delta_2^{2\ell-1}$. This implies that $2k-1$ is not useful to either of the triples, a contradiction. $\square$

Notice that the reverse claim does not hold.

Lemma 4.1 *For any $x \neq y$ and for any two sequences σ_x and σ_y, there exists an exposed triple in σ_y.*

Proof. From Claims 4.1 and 4.2, if none of triples 1 through j are exposed and $j \in \Gamma^{(i)}$, then $sixplet(i) \geq sixplet(j)$. Since $x \neq y$, there exists a bit, say the jth one, at which their ID's differ. Since the scheme uses both an I_j-sixplet and $1 - I_j$-sixplet, there exists some k such that the kth sixplet in σ_x is a 0-sixplet while the kth sixplet in σ_y is a 1-sixplet. The Lemma now follows from Claim 4.3. $\square$ $\square$

4.2 The Knowledge Extractor

Consider an adversarially coordinated system $\langle (A,B), (C,D), \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ where (A,B) and (C,D) are both instances of $\mathcal{S}$. Intuitively, if α and β are the strings committed to by $\psi(A)$ and $\psi(C)$, respectively, our goal is to extract β. To achieve this we devise a somewhat different protocol, called $\mathcal{S}'$, on which the extractor operates, and from which it extracts β. This new protocol is a string commitment protocol that is not necessarily non-malleable. In the next section we prove that extraction of β from $\mathcal{S}'$ implies the non-malleability of $\mathcal{S}$.

The string commitment scheme $\mathcal{S}'$ consists of a Committer P and a Receiver Q and takes a parameter m.

P **Commits to a string** α**:**

- Commit to α using the protocol in [24].
- Repeat m times:
  1. Q chooses a bit b and chooses whether or not it wishes to see an additional proof of consistency in step **BCK3** of either triple in the b-sixplet; Q requests a b-sixplet, possibly augmented by an additional proof of consistency;
  2. P and Q run a (possibly augmented) b-sixplet;

From the semantic security of the regular string commitment protocol and from the zero-knowledge properties, a standard simulation argument yields the following lemma:

Lemma 4.2 *For any strategy of choosing the sixplets and for any receiver Q', the above protocol is a string commitment protocol which is semantically secure.* □

We provide an adapter that allows us to emulate to $\mathcal{A}$ (and its controlled machines) a player A that executes $\mathcal{S}$, whereas in reality $\psi(B)$ (under control of $\mathcal{A}$) communicates with the sender P of $\mathcal{S}'$. $\mathcal{S}'$ has been designed so that it can actually tolerate communicating with many copies of $\mathcal{A}$, with messages from the different copies being "multiplexed" by the adapter. (P cannot distinguish this scenario from the one in which it is communicating with a single receiver Q.)

In more detail, suppose that player $\psi(P)$ is running the sender part of $\mathcal{S}'$ and that player $\psi(B)$ is supposed to run the receiver part of $\mathcal{S}$. ($\psi(B)$ might deviate from the protocol as written, but the communication steps are as in $\mathcal{S}$.) It is not hard to construct an adapter that operates between Q and B: whenever (A, B) calls for a b-sixplet the adapter "pretends it is Q" and asks for a b-sixplet; then $\psi(B)$ and $\psi(P)$ run the b-sixplet. It should be clear that the distribution of conversations that $\psi(B)$ sees when it participates in $\mathcal{S}$ and the distribution of conversations it sees when it participates through the adapter in $\mathcal{S}'$ are identical.

We are now ready to present the extractor. Suppose that in the adversarially coordinated system the probability that $\psi(C)$ completes its part successfully is ρ. Following the commit stage (during which C may or may not have committed in any meaningful way), we cannot in general hope to extract β with probability greater than ρ. However we can get arbitrarily close: we will show that for any ϵ we can successfully extract β with probability at least $\rho - \epsilon$.

Fix $\epsilon > 0$. The knowledge extractor begins to run $\mathcal{S}' = (P, Q)$ and $\mathcal{S} = (C, D)$ with the adapter arranging that $\mathcal{A}$ cannot distinguish this from the adversarially coordinated system $\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ (see Figure 2).

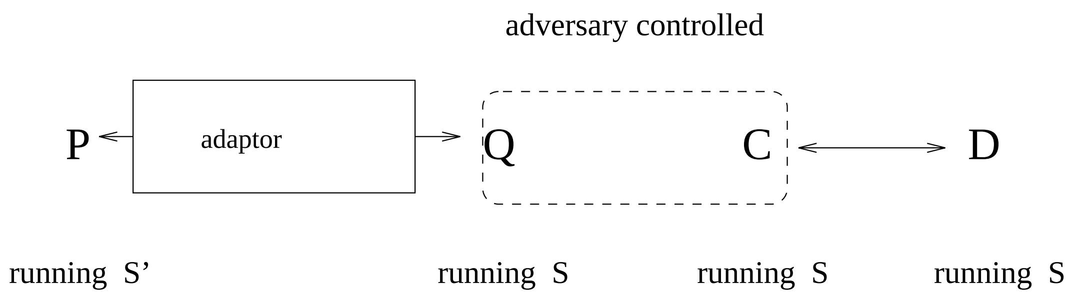

Figure 2: The $\mathcal{S}'$-adaptor-$\mathcal{S}$ system used in constructing the Extractor

Once $\psi(C)$ completes the first (commitment) stage of $\mathcal{S}$, the extractor freezes the random tape of $\mathcal{A}$.

$\mathcal{A}$ now defines a tree according to all possible communication exchanges between A and D. The tree contains A-nodes and D-nodes, according to which of the two is the next one to send a message. The root of the tree corresponds to the point at which the tape was frozen. Thus, the branching at each node is all possible messages that either A or D can send at that point. To enable us to follow more than a single path (that is, to *fork*) in the tree, we keep at each D-node a snapshot of the current state, i.e. a copy of $\mathcal{A}$'s tapes and the states of A and D.

A node v is *good* if all the communication between C and D up to v is legal (according to the non-malleable protocol $\mathcal{S}$) and C successfully opened and proved whenever it was asked to do so. Our goal is to identify two nodes having the following properties: (1) at each of the two, C has just completed a **BCK**3 step; (2) the paths to the two nodes depart in a branching at a D-node. As noted above, given two such nodes we can extract β.

To identify such a pair of nodes, choose $\ell = \frac{16}{(\epsilon^2 \log \epsilon)}$, and run the following extraction procedure ℓ times. Each time start again at the root of tree.

Recall that by Lemma 4.1 every path to a good leaf contains an exposed triple. Run the $\mathcal{S}'$-adapter-$\mathcal{S}$ system until an exposed triple j in σ_y is reached (or we reach a bad node). We partition the exposed triples into two types:

- j is of the first type if $\forall i\ \tau_1^i > \delta_1^j$ (nothing happened yet in σ_x between $\psi(A)$ and $\psi(B)$) or $\forall i$ s.t. $\tau_1^i < \delta_1^j$ we have $\tau_2^i < \delta_2^j$ (the challenge in σ_x ends before the challenge in σ_y begins).

- j is of the second type if it is not of the first type and $\forall i$ s.t. $\tau_1^i < \delta_1^j$ and $\tau_2^i > \delta_2^j$ we have $\tau_2^i > \delta_3^j$ (the challenge in σ_x ends after the reply in σ_y ends, so $\psi(C)$ can't use the answers from (A, B) to help it answer challenges from $\psi(D)$).

In the first type of exposure, for each i there are two possible interleavings:

τ_1^i τ_1^i

$$\begin{array}{ccccc} & \delta_1^j & \text{or} & \tau_2^i & \\ \tau_2^i & & & & \delta_1^j \\ & \delta_2^j & & & \delta_2^j \end{array}$$

Thus, there exists a time t, $\delta_1^j \leq t \leq \delta_2^j$ such that for all i, $\tau_1^i \leq t \Rightarrow \tau_2^i \leq t$. The time t is the maximum of δ_1^j and the maximum over all i such that $\tau_1^i \leq \delta_1^j$, of τ_2^i. In this case, intuitively, for every i such that the values committed to by $\psi(C)$ in $\mathbf{BCK}(j)$ may depend on the values committed to by $\psi(P)$ in $\mathbf{BCK}(i)$, the queries made by $\psi(Q)$ to $\psi(P)$ about these values are independent of the queries made by $\psi(D)$ to $\psi(C)$. It follows that $\psi(C)$ can't get any help from $\psi(P)$ in answering $\psi(D)$'s queries in $\mathbf{BCK}(j)$.

At the point t defined above, P has no triples of which step $\mathbf{BCK}1$ has completed but $\mathbf{BCK}2$ has not yet begun, thus A doesn't play a part right now. At this point we fork: the extractor creates a new copy of $\mathcal{A}$ and D, and runs both this copy and the original, with each copy of D making independent random queries. Note that with overwhelming probability any two such vectors differ in some position. The vectors are the various choices associated for all the concurrent instances in the protocol. The adapter multiplexes to P the vectors of queries made by the two copies of B.

In the second type of exposed triple, the exposure does not become evident until δ_3^j. At any point in time there are at most two triples between A and B that are *open*, in that step $\mathbf{BCK}1$ has been executed but $\mathbf{BCK}2$ has not. Say that at δ_1^j the two open triples are the i and $i+1$ ones. We know that $\tau_1^i < \tau_1^{i+1} < \delta_1^j$ and $\tau_2^i > \tau_2^{i+1} > \delta_1^j$ and $\tau_2^i > \delta_3^j$. We distinguish between two cases: (a) $\tau_2^{i+1} < \delta_2^j$ and (b) $\tau_2^{i+1} > \delta_3^j$ (since j is exposed it cannot be the case that $\delta_2^j < \tau_2^{i+1} < \delta_3^j$.) In case (a) we fork right after τ_2^{i+1} by running a copy of $\mathcal{A}$ until the conclusion of triple j in the copy. Note that we can assume that $\tau_3^{i+1} < \delta_2^j$ because the replies and proofs that P sends in $\mathbf{BCK}3$ of triple $i+1$ are completely determined by $\mathbf{BCK}1$ and $\mathbf{BCK}2$, and are therefore are completely determined by time τ_2^{i+1}. Moreover, the challenges sent in $\mathbf{BCK}2(j)$ by D are independent of $\mathbf{BCK}(i+1)$ (by definition, because j is exposed). D makes independent random choices in this run. We will not run the original beyond δ_3^j. The communication with A is limited in the original execution to the zero knowledge proof of consistency. However, since in $\mathcal{S}'$ the committer P is willing to repeat this proof, when running the copy we simply ask for such a repetition and continue as before. We stop when the copy finishes $\mathbf{BCK}3$ of the j triple. Note that in the copy the jth triple need not be exposed (this depends on τ_2^i). Case (b) is even simpler: we fork right after δ_1^j. In the original $\psi(B)$ does not communicate with P until δ_3^j, so we simply continue with the copy until it finishes $\mathbf{BCK}3$ of the jth triple. Here again we have that j need not be exposed in the copy.

In exploiting either type of exposure, if in both branches (the original and the copy) the proof of consistency in $\mathbf{BCK}3(j)$ succeeds, then in triple j the extractor obtains the answers to two different and independent queries, hence β can be extracted. The significance of the zero-knowledge proof of consistency is that it allows the extractor to know whether any trial is successful. Therefore if any trial is successful the extractor succeeds.

This completes the description of the extractor. We now show that its probability of success is high.

At each node v of the tree we can define the probability of success, $\rho(v)$, i.e., the probability that the communication between A and D leads to a good leaf. Let ρ_0 be the probability of success at the root. Notice that by definition, the expected value of ρ_0 is ρ.

Lemma 4.3 *In each run of the above experiment β is successfully extracted with probability $\rho_0^2/4 - 1/2^{2n}$.*

Proof. Consider a random root-leaf path w in the tree (the randomness is over the coin flips of A and D). At each node v let $\rho(v)$ denote the probability, taken over choices of A and of D, of successfully completing the execution from v. Let $\rho(w)$ be the minimum along the execution path w. Note that $\rho(w)$ is a random variable.

Claim 4.4 *With probability at least $\rho_0/2$ we have $\rho(w) > \rho_0/2$.*

Proof. The probability of failure is $1-\rho_0$. Let $\mathcal{V}$ be the set of nodes v such that $\rho(v) < \rho_0/2$ and for no parent u of v is $\rho(u) < \rho_0/2$ (i.e. the $\mathcal{V}$ consists of the "first" nodes such that $\rho(v) < \rho_0/2$). We know that $\Pr[\rho(w) \leq \rho_0/2] \leq \sum_{v \in \mathcal{V}} \Pr[v \text{ is reached}]$. On the other hand, the probability of failure, $1 - \rho_0$, is

$$\sum_{v \in \mathcal{V}} \Pr[v \text{ is reached}](1 - \rho(v) \geq (1 - \rho_0/2) \sum_{v s.t. \rho(v) \leq \rho_0/2} \Pr[v \text{ is reached}].$$

Therefore $\Pr[\rho(w) \leq \rho_0/2] \leq \frac{1-\rho_0}{1-\rho_0/2} = 1 - \frac{\rho_0/2}{1-\rho_0/2} < 1 - \rho_0/2$. □

Thus, with probability $\rho_0/2$ the main path we take succeeds. The experiment branches at a point with probability of success $\rho_0/2$. The probability of success of each branch is independent. Therefore, the new branch succeeds with probability $\rho_0/2$. Excluding a small probability $1/2^{2n}$ that both branches choose identical strings, the experiment succeeds with probability $\rho_0^2/4 - 1/2^{2n}$. □

With probability $\rho - \epsilon/2$ the probability of success at the root, ρ_0, is at least $\epsilon/2$. The extractor makes ℓ independent experiments. Because of the proof of consistency, extraction fails only if all experiments fail. This occurs with probability at most $(1 - \frac{\rho_0^2}{4})^\ell$. The choice of ℓ implies that the probability that the extractor succeeds, given that $\rho_0 > \epsilon/2$, is at least

$$1 - (1 - \frac{\rho_0^2}{4})^\ell \geq 1 - (1 - \frac{\epsilon^2}{4})^\ell \geq 1 - \epsilon/2.$$

Therefore, with probability at least $\rho - \epsilon$ the string β is extracted in at least one of the ℓ experiments. Thus we can conclude that,

Lemma 4.4 *For any adversarially coordinated system $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ there is a knowledge extraction procedure that succeeds with probability arbitrarily close to ρ.* □

The next claim says, in essence, that the values β obtained by the extractor are "correctly" distributed.

Claim 4.5 *Let $\alpha \in_R \mathcal{D}$ and let β be obtained by the extractor. Then for every relation approximator R, either (1) the probability that $R(\alpha, \beta)$ outputs 1, where the probability space is over the choice of α and the internal coin flips of the machines involved, is larger than $\pi(\mathcal{A}, R)$ or (2) these two probabilities are arbitrarily close.*

4.3 Extraction Implies Non-Malleability

In this subsection we reduce the non-malleability of $\mathcal{S}$ to the security of $\mathcal{S}'$. Let R be a relation approximator and let $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$ be an adversarially controlled system, where (A,B) and (C,D) are both instances of $\mathcal{S}$.

Consider the following procedure for an adversary simulator $\mathcal{A}'$ with access to the probability distribution $\mathcal{D}$.

Procedure for $\mathcal{A}'$:

1. Generate $\delta \in_R \mathcal{D}$.

2. Emulate the system $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$ where $\psi(A)$ is running $\mathcal{S}'$ with private input δ and $\mathcal{A}$ has access to hist(δ), and extract γ, the value committed to by $\psi(C)$ in the emulation.

3. Output γ.

The structure of the proof is as follows. Let $\alpha \in_R \mathcal{D}$. We define three random variables:

1. Let β be the value committed to by C in an execution of $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$ in which A has input α and $\mathcal{A}$ has input hist(α). By definition, $\Pr[R(\alpha,\beta)] = \pi(\mathcal{A},R)$.

2. Let β' be obtained by extraction from $\mathcal{A}'$ in a run of the $\mathcal{S}'$-adaptor-$\mathcal{S}$ system in which P has input α but $\mathcal{A}'$ has input hist(α). By definition, $\Pr[R(\alpha,\beta')] = \pi'(\mathcal{A}',R)$.

3. Let β'' be obtained by extraction from $\mathcal{A}'$ in a run of the $\mathcal{S}'$-adaptor-S system in which P has input $\delta \in_R \mathcal{D}$ but $\mathcal{A}'$ has input hist(α). Let $\tilde{\pi}'(\mathcal{A}',R)$ denote $\Pr[R(\alpha,\beta'')]$.

We will first show that if $|\Pr[R(\alpha,\beta')] - \Pr[R(\alpha,\beta'')]|$ is polynomial, then there is a distinguisher for $\mathcal{S}'$. By the semantic security of $\mathcal{S}'$, this means that $\pi'(\mathcal{A}',R) = \Pr[R(\alpha,\beta')]$ is very close to $\tilde{\pi}'(\mathcal{A}',R) = \Pr[R(\alpha,\beta'')]$. In other words, on seeing the history hist(α), $\mathcal{A}'$, interacting with P having input α, is essentially no more successful at committing to a value related by R to α than $\mathcal{A}'$ can be when it again has history hist(α) but is actually interacting with P having input δ (unrelated to α). This means that, for $\mathcal{A}'$, having the interaction with P doesn't help in committing to a value related to P's input.

Let us say that $\mathcal{A}'$ *succeeds* in an execution of the $\mathcal{S}'$-adaptor-$\mathcal{S}$ system, if $\psi(C)$ commits to a value related by R to P's input (the value to which P commits). Similarly, we say that $\mathcal{A}$ succeeds in an execution of $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$ if $\psi(C)$ it commits to a value related by R to A's input. Recall that, by Claim 4.5, either $\mathcal{A}$ is essentially equally likely to succeed as $\mathcal{A}'$, or $\mathcal{A}$ is less likely to succeed than $\mathcal{A}'$ is. So the probability that $\mathcal{A}$ succeeds, $\pi(\mathcal{A},R)$ is essentially less than or equal to $\pi'(\mathcal{A}',R)$, which we show in the first step of the proof to be close to $\tilde{\pi}'(\mathcal{A}',R)$. ¿From this we conclude the non-malleability of $\mathcal{S}$.

Lemma 4.5 *If $|\tilde{\pi}'(\mathcal{A}',R) - \pi'(\mathcal{A}',R)|$ is polynomial, then there is a distinguisher for $\mathcal{S}'$ that violates the indistinguishability of encryptions (commitment) property (equivalent to semantic security [13, 17, 23]).*

Proof. Assume $|\tilde{\pi}'(\mathcal{A}', R) - \pi'(\mathcal{A}', R)|$ is polynomial. The distinguisher is as follows.
Distinguisher for $\mathcal{S}'$:

1. Create a random challenge $(\alpha_1 \in_R \mathcal{D}, \alpha_2 \in_R \mathcal{D})$;
2. Choose $i \in_R \{1,2\}$. Emulate the system $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$, where $\psi(A)$ is running $\mathcal{S}'$ with private input α_i and $\mathcal{A}$ has access to $\mathrm{hist}(\alpha_1)$, and extract ζ, the value committed to by $\psi(C)$ in the emulation.
3. Output $R(\alpha_1, \zeta)$.

If, in the emulation, the input to $\psi(P)$ is α_1, then the distinguisher outputs 1 with probability $\pi'(\mathcal{A}', R)$. Similarly, if in the emulation the input to $\psi(P)$ is α_2, then the distinguisher outputs 1 with probability $\tilde{\pi}'(\mathcal{A}', R)$. Since by assumption these two quantities differ polynomially, we have a polynomial distinguisher for commitments in $\mathcal{S}'$. $\square$

Corollary 4.6 *$|\tilde{\pi}'(\mathcal{A}', R) - \pi'(\mathcal{A}', R)|$ is subpolynomial.* $\square$

Theorem 4.7 *The string commitment scheme $\mathcal{S}'$ is non-malleable.*

Proof. By Claim 4.5, $\pi(\mathcal{A}, R) < \pi'(\mathcal{A}', R)$ or the two are subpolynomially close. Thus $\mathcal{A}$ is not polynomially more likely to successfully commit to a value related by R to the value committed to by $\psi(A)$ than $\mathcal{A}'$ is able to commit to a value related by R to the value committed to by $\psi(P)$. However, by Lemma 4.5, $\pi'(\mathcal{A}', R)$ is subpolynomially close to $\tilde{\pi}(\mathcal{A}', R)$; that is, interacting with P does not help $\mathcal{A}'$ to commit to a value related to the value committed to by $\psi(P)$. $\square$

Remark 4.8 *The number of rounds in the above protocol is proportional to the length of $|I|$. However, the number of rounds may be reduced to $\log|I|$ using the following: Let $n = |I|$. To commit to string α, choose random $\alpha_1, \alpha_2, \ldots \alpha_n$ satisfying $\bigoplus_{i=1}^{n} \alpha_1 = \alpha$. For each α_i (in parallel) commit to α_i with identity (i, I_i) (i concatenated with the ith bit of the original identity).*

To see why this is secure, consider an adversary with identity $I' \neq I$ who commits to α'. For $I' \neq I$ there must be at least one i such that $I'_i \neq I_i$ (we assume some prefix free encoding). This i implies the non-malleability of the resulting scheme: Make α_j for $j \neq i$ public. Since all the identities of the form (j, I'_j) are different than (i, I_i) we can extract all the α'_j's and hence α'.

Remark 4.9 *As we have seen, the proofs of consistency aid in the extraction procedure. Interestingly, they also ensure that if there are many concurrent invocations of (A,B), call them*

$$(A_1, B_1), \ldots, (A_k, B_k),$$

such that the adversary controls all the $\psi(B_i)$ and $\psi(C)$, then if C commits to a value β to D then β is essentially unrelated to all the α_i committed to by the A_i in their interactions with the B_i.

5 General Non-Malleable Zero-Knowledge Interactions

For the results in this section we assume the players have unique identities. Let (A, B) be an interactive protocol with *joint* distribution $\mathcal{D}$ on the inputs to A and B. Roughly speaking, (A, B) is *zero-knowledge* with respect to B if for every $\psi(B)$ there exists a simulator Sim such that the following two ensembles of conversations are indistinguishable. In the first ensemble, a pair α, β is drawn according to $\mathcal{D}$, $\psi(A)$ gets α, $\psi(B)$ gets β, and the interaction proceeds and produces a conversation. In the second ensemble, $\alpha, \beta \in_R \mathcal{D}$ is selected, Sim is given β, and produces a conversation.

We have constructed a compiler, which, given any zero-knowledge interaction (A, B) produces a zero-knowledge protocol which is non-malleable in the sense described next.

Consider an adversarially coordinated system $\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ in which (A, B) *need not* be the same protocol as (C, D) (in particular, (C, D) needn't be zero-knowledge). Note that, if (A, B) were to be run in isolation, then given the inputs α, β and the random tapes of $\psi(A)$ and $\psi(B)$, the conversation between these agents is completely determined. A similar statement applies to (C, D). For every polynomial time relation approximator R and for every adversarially coordinated system of the compiled versions with adversary $\mathcal{A}$ there exists an adversary simulator $\mathcal{A}'$ satisfying the following requirement.

Let $\mathcal{D}$ now denote the distribution for inputs to all four players. Let $(\alpha, \beta, \gamma, \delta) \in_R \mathcal{D}$ and run the compiled versions of the two protocols. Let $p_{\mathcal{A}}$ denote the probability that $R(\alpha, \beta, \gamma, \delta, \mathcal{C}(C, D)) = 1$, where $\mathcal{C}(C, D)$ denotes the conversation between C and D. As above, R rejects if a conversation is syntactically incorrect.

Again, let $(\alpha, \beta, \gamma, \delta) \in_R \mathcal{D}$. $\mathcal{A}'$ gets inputs β, γ. Run an execution of (C, D) in which $\mathcal{A}'$ controls $\psi(C)$ and let $\mathcal{C}'(C, D)$ denote the resulting conversation. Let $p_{\mathcal{A}'}$ denote the probability that $R(\alpha, \beta, \gamma, \delta, \mathcal{C}'(C, D)) = 1$.

The security requirement is that for any $\mathcal{A}$ there exists an $\mathcal{A}'$ (of comparable complexity) such that $|p_{\mathcal{A}} - p_{\mathcal{A}'}|$ is subpolynomial.

Our compiler is extremely simple. Each player commits to its input and the seed to a cryptographically strong pseudo-random bit generator, using the non-malleable scheme for string commitment described in Section 4. The pseudo-random sequence is used instead of a truly random sequence whenever the original protocol calls for a random bit. The parties then execute the original protocol (with the pseudo-random bits), with each player proving at each step that the message it sends at that step is the one it should have sent in the unique conversation determined by its committed inputs and messages of the original protocol received so far. Since proving the consistency of the new message with the conversation so far can be done effectively (given the random tape and the input), this proof has a (malleable) zero-knowledge protocol [16].

Informally, we prove non-malleability of the compiled programs by constructing an extractor for the committed values in a fashion similar to the one constructed in Section 4. We apply Lemma 4.4 to show that the probability of extraction is similar to the probability of success (in the (C, D) protocol). The rest of the proof then follows from the zero-knowledge property of the proof of consistency.

The issue of preserving the non-malleability of compiled programs under concurrent composition is delicate, as in general zero-knowledge proofs do not compose (e.g. [14]). As in Remark 4.9, the proofs of consistency in the steps of our compiled programs play a key role in yielding non-malleability under composition.

Acknowledgments

Discussions with Moti Yung on achieving independence started this research. Advice and criticism from Russell Impagliazzo, Charlie Rackoff and Dan Simon were critical in the formation of the paper. We thank Uri Feige, Hugo Krawczyk, Oded Goldreich and Adi Shamir for valuable discussions.

References

[1] W. Alexi, B. Chor, O. Goldreich and C. Schnorr, *RSA/Rabin Bits are* $1/2 + 1/poly$ *Secure*, Siam Journal on Computing, 17(2) (1988), pp.194-209.

[2] M. Bellare and S. Micali, *How to Sign Given Any Trapdoor Function*, Proc. 20th Annual Symposium on the Theory of Computing, Chicago, 1988, pp.32-42.

[3] 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.

[4] M. Blum, A. De Santis, S. Micali and , G. Persiano, *Non-Interactive Zero-Knowledge*, SIAM J. Computing, 1991, pp. 1084–1118.

[5] M. Blum and S. Goldwasser, *An Efficient Probabilistic Public-key Encryption that Hides All Partial Information*, Advances in Cryptology - Crypto 84, Lecture Notes in Computer Science No. 196, 1985 Springer Verlag, pp. 289-299.

[6] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, *Verifiable Secret Sharing in the Presence of Faults*, Proc. 26th IEEE Symp. on Foundations of Computer Science, 1985, pp. 383-395.

[7] B. Chor and M. Rabin, *Achieving Independence in Logarithmic Number of Rounds*, Proc. 6th ACM Symp. on Principles of Distributed Computing, 1987, pp. 260-268.

[8] Y. Desmet, C. Goutier and S. Bengio, *Special uses and abuses of the Fiat Shamir passport protocol* Advances in Cryptology - Crypto 87, Lecture Notes in Computer Science No. 293, Springer Verlag, 1988, pp. 21-39.

[9] A. De Santis and G. Persiano, *Non-Interactive Zero-Knowledge Proof of Knowledge* Proc. 25th ACM Symposium on the Theory of Computing, Chicago, 1993.

[10] U. Feige and A. Shamir, *Witness Hiding and Witness Indistinguishability*, Proc. 22nd Annual Symposium on the Theory of Computing, Baltimore, 1990, pp. 416–426.

[11] 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).

[12] 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.

[13] O. Goldreich, *A Uniform Complexity Encryption and Zero-knowledge*, Technion CS-TR 570, June 1989.

[14] O. Goldreich and H. Krawczyk, *On the Composition of Zero-knowledge Proof Systems*, Technion CS-TR 568, June 1989. ICALP '90.

[15] O. Goldreich and L. Levin, *A Hard Predicate for All One-way Functions* , Proc. 21st Annual Symposium on the Theory of Computing, Seattle, 1989, pp. 25-32.

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

[17] S. Goldwasser and S. Micali, *Probabilistic Encryption*, J. Com. Sys. Sci. 28 (1984), pp 270-299.

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

[19] S. Goldwasser, S. Micali and R. Rivest, *A Secure Digital Signature Scheme* , Siam Journal on Computing, Vol. 17, 2 (1988), pp. 281-308.

[20] J. Kilian, *On the complexity of bounded-interaction and non-interactive zero-knowledge proofs*, Proceedings of the 35th IEEE Symposium on the Foundation of Computer Science, 1994.

[21] J. Kilian and E. Petrank, *An efficient non-interactive zero-knowledge proof system for NP with general assumptions*, Electronic Colloquium on Computational Complexity TR95-038. Available: http://www.eccc.uni-trier.de/eccc/info/ECCC.

[22] Luby M., **Pseudo-randomness and applications**, Princeton University Press, To appear.

[23] S. Micali and C. Rackoff and R. Sloan, *Notions of Security of Public-Key Cryptosystems*, SIAM J. on Computing 17(2) 1988, pp. 412-426.

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

[25] M. Naor and M. Yung, *Universal One-way Hash Functions and their Cryptographic Applications*, Proc. 21st Annual Symposium on the Theory of Computing, Seattle, 1989, pp. 33-43.

[26] M. Naor and M. Yung, *Public-key Cryptosystems provably secure against chosen ciphertext attacks* Proc. 22nd Annual Symposium on the Theory of Computing, Baltimore, 1990, pp. 33-43.

[27] C. Rackoff and D. Simon, Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack, Advances in Cryptology - Crypto'91, Lecture Notes in Computer Science No. 576, Springer Verlag, 1992, pp. 433–444.

[28] R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signature and Public Key Cryptosystems*, Comm. of ACM, 21 (1978), pp 120-126.

[29] J. Rompel, *One-way Function are Necessary and Sufficient for Signatures*, Proc. 22nd Annual Symposium on the Theory of Computing, Baltimore, 1990, pp. 387–394.

[30] A. C. Yao, *Theory and Applications of Trapdoor functions*, Proceedings of the 23th Symposium on the Foundation of Computer Science, 1982, pp. 80-91.