Non-Malleable Cryptography*

Danny Dolev[†] Cynthia Dwork[‡] Moni Naor[§]

March 24, 2000

Abstract

The notion of *non-malleable* cryptography, an extension of semantically secure cryptography, is defined. Informally, in the context of encryption 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.

Our cryptosystem is the first proven to be secure against a strong type of chosen ciphertext attack proposed by Rackoff and Simon, in which the attacker knows the ciphertext she wishes to break and can query the decryption oracle on any ciphertext other than the target.

Keywords: cryptography, cryptanalysis, randomized algorithms, non-malleability, chosen-ciphertext security, auction protocols, commitment schemes, zero-knowledge

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

---

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

[†]Dept. of Computer Science, Hebrew University Jerusalem, Israel.

[‡]IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120. Research supported by BSF Grant 32-00032-1. E-mail: dwork@almaden.ibm.com.

[§]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 BSF Grant 32-00032-1. E-mail: naor@wisdom.weizmann.ac.il.

1 Introduction

The notion of *non-malleable* cryptography, is an extension of semantically secure cryptography. Informally, in the context of encryption the additional requirement is that given the ciphertext it is impossible to generate a *different* ciphertext so that the respective plaintexts are related. For example, 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 $E(15,000,000)$ to the published number over an insecure line. Intuitively, the public-key cryptosystem is *malleable* if, having access to $E(15,000,000)$, Company B is more likely to generate a bid $E(\beta)$ such that $\beta \leq 15,000,000$ 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. It also does not require the municipality M to know of A or B before the companies bid, nor does it require A or B to have any kind of public key. The system remains non-malleable even under a very strong type of chosen ciphertext attack in which the attacker knows the ciphertext she wishes to break (or maul) and can query the decryption oracle on any ciphertext other than the target.

A well-established, albeit implicit, notion of non-malleability is existential unforgeability of signature schemes [45]. Informally, a signature scheme is existentially unforgeable if, given access to $((m_1, S(m_1)), \ldots, (m_k, S(m_k))$, where $S(m_i)$ denotes a signature on message m_i, the adversary cannot construct a single valid $(m, S(m))$ pair for any *new* message m – even a nonsense message or a function of $m_1, \ldots, m_k$. Thus, existential unforgeability for signature schemes is the "moral equivalent" of non-malleability for cryptography. We do not construct signature schemes in this paper. However, we introduce the related notion of *public-key authentication* and present a simple method of constructing a provably secure public-key authentication scheme based on any non-malleable public-key cryptosystem[1].

Non-malleability is also important in private-key cryptography. Many common protocols, such as Kerberos or the Andrew Secure Handshake, use private key encryption as a sort of authentication mechanism: parties A and B share a key K_{AB}. A sends to B the encryption of a nonce N under K_{AB}, and the protocol requires B to respond with the encryption under K_{AB} of $f(N)$, where f is some simple function such as $f(x) = x - 1$. The *unproved and unstated* assumption (see, *e.g.* [16]) is that seeing $K_{AB}(N)$ doesn't help an imposter falsely claiming to be B to compute $K_{AB}(f(N))$. As we shall see, this is *precisely* the guarantee provided by non-malleability.

Non-malleability is a desirable property in many cryptographic primitives other than encryption. For example, suppose Researcher A has obtained a proof that $P \neq NP$ and wishes to communicate this fact to Professor B. Suppose that, to protect herself, A proves her claim to B in a zero-knowledge fashion. Is zero-knowledge sufficient protection? 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

[1] For more on existentially unforgeable signature schemes see [27, 45, 60].

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 (initially) unaware of Researcher A.

Our work on non-malleability was inspired by early attempts to solve the distributed coin flipping problem. Although $t+1$ rounds are necessary for solving Byzantine agreement in the presence of t faulty processors [33], in the presence of a global source of randomness the problem can be solved in constant expected time [62]. Thus, in the mid-1980's several attempts were made to construct a global coin by combining the individual sources of randomness available to each of the participants in the system. At a very high level, the original attempts involved commitment to coins by all processors, followed by a revelation of the committed values. The idea was that the global coin would be the exclusive-or (or some other function) of the individual committed values. Disregarding the question of how to force faulty processors to reveal their committed values, the original attempts erred because *secrecy was confused with independence.* In other words, the issue was malleability: even though the faulty processors could not know the committed values of the non-faulty processors, they could potentially force a desired outcome by arranging to commit to a specific function of these (unknown) values.

As the examples show, *secrecy* does not imply *independence.* The goal of non-malleable cryptography is to *force* this implication.

1.1 Description of Principal Results

Non-Malleable Public Key Cryptography

Goldwasser and Micali define a cryptosystem to be *semantically secure* if anything computable about the cleartext from the ciphertext is computable without the ciphertext [43]. This powerful type of security may be insufficient in the context of a distributed system, in which the mutual independence of messages sent by distinct parties often plays a critical role. For example, 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 $E(\alpha)$[2]. We present public-key cryptosystem that is non-malleable even against what we call a chosen ciphertext attack in the post-processing mode (defined informally in Section 2.1 and formally in Section 3). Since non-malleability is an extension of semantic security, this yields the first public-key cryptosystem that is semantically secure against this strong type of chosen ciphertext attack[3].

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

[2] 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 Section 2.

[3] For this type of attack it turns out that semantic and non-malleable security are equivalent, which is not the case for weaker attacks. See Section 3.4.2.

research touching on this problem of which we are aware requires at least one of these assumptions (*e.g.*, [20, 21, 63]).

Non-Malleable String Commitment

A second important scenario for non-malleability is 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 knows 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 polynomial-time computable 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 to require that 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 the (A, B) 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* (perhaps A) knows β, but not necessarily that C does.

Non-Malleable Zero-Knowledge Protocols

Using non-malleable string commitment as a building block, we can convert any zero-knowledge interaction into a non-malleable one. In particular we obtain non-malleable zero-knowledge proofs of possession of knowledge, in the sense of Feige, Fiat, and Shamir [31]. Zero-knowledge protocols [44, 40] may compose in an unexpectedly malleable fashion. A classic example is the so-called "man-in-the-middle" attack (also known as the "intruder-in-the-middle," "Mafia scam," and "chess-masters problem") [24] 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. The construction in Section 5 yields a non-malleable scheme for zero-knowledge proof of possession of knowledge.

1.2 Some Technical Remarks

Non-Malleability in Context

In the scenarios 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, say, 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.

Identities

One delicate issue is the question of identities. Let α and β be as above. If the players have names, then our commitment and zero-knowledge interaction protocols guarantee 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 $\alpha = A \circ s$. In this case the only dependent statement provable by Professor B is α, which contains the name A. Note that we do not assume any type of authenticated channels.

Computational complexity assumptions

We assume the existence of trapdoor functions in constructing our public-key cryptosystems. The string commitment protocols and the compiler for zero-knowledge interactions 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 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. We start in Section 2.1 by providing definitions for the primitives, as well as for some of the tools we use. Our presentation of the notion of security is non-standard and we call it semantic security with respect to relations. In Theorem 2.2 we show that our version is equivalent to the "traditional" definition We prefer this version for several reasons:

- It provides a uniform way of treating the security of all the primitives, *i.e.*, the definition of zero-knowledge and semantic security do not *seem* different.
- It generalizes to the non-malleable case in a natural way, whereas the usual notion of semantic security (provably) does not.

In Section 2.2 we provide the definition of non-malleable security. In Section 2.3 we define the system model which is most relevant to those primitives which involve a lot of interaction.

The following definitions and notation are common to all the sections. We use $X \in_R B$ to mean that X is chosen from B at random. If B is a set then X is simply chosen uniformly at random from the elements of B. If B is a distribution, then $X \in_R B$ means that X is chosen according to B from the support of B.

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. Thus A is the protocol, and $\psi(A)$ is the player.

2.1 Definitions of Primitives

In this section we review the definitions from the literature of probabilistic public key cryptosystems, string commitment, zero-knowledge interaction and non-interactive zero-knowledge proof systems, all of which are used as primitives in our constructions. As mentioned above, we provide a unifying treatment of the security of all the primitives.

Probabilistic Public Key Encryption

A *probabilistic public key encryption* scheme (see [43]) consists of:

- GP, the *key generator*. A probabilistic machine that on unary input 1^n, where n is 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

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

- The system has the property of *indistinguishability*: for all polynomial time machines M, for all $c > 0$ $\exists n_c$ s.t. for $n > n_c$

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

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

This definition is for *bit* encryption and the existence of such a method suffices for our constructions. To encrypt longer messages one can concatenate several bit encryptions or use some other method. The definition of indistinguishability in this case becomes that with overwhelming probability over choice of encryption keys e, M cannot find two messages (m_0, m_1) for which it can distinguish with polynomial advantage between encryptions of m_0 and m_1. Formally:

Definition 2.1 *Let (GP, E, D) be a probabilistic public-key cryptosystem. We say the that system has the property of indistinguishability of encryptions if for all pairs of probabilistic polynomial time machines $(\mathcal{F}, \mathcal{T})$, for all $c > 0$ $\exists n_c$ s.t. for $n > n_c$*

$$Pr[|Pr[\mathcal{T}(e, m_0, m_1, E_e(m_0, r)) = 1] - Pr[\mathcal{T}(e, m_0, m_1, E_e(m_1, r)) = 1]| \geq \frac{1}{n^c}] < \frac{1}{n^c}$$

where the external probability is over the choice of e and the coin flips of $\mathcal{F}$ (which gets e as input), and each internal probability is taken the coin flips of $\mathcal{T}$ and the choice of r.

For implementations of probabilistic encryption see [2, 14, 39, 52, 66]. In particular, such schemes can be constructed from *any* trapdoor permutation.

When describing the security of a cryptosystem, one must define what the attack is and what it means to break the system. The traditional notion of breaking (since [43]) has been a violation of semantic security or, equivalently, a violation of indistinguishability. This work introduces the notion of non-malleable security, and a break will be a violation of non-malleability. We return to this in Section 2.2. We consider three types of attacks against a cryptosystem:

- Chosen plaintext. This is the weakest form of attack that makes any sense against a public-key cryptosystem. The attacker can (trivially) see a ciphertext of any plaintext message (because she can use the public encryption key to encrypt).

- Chosen ciphertext in the sense of [61], sometimes called lunch-break or lunch-time attacks in the literature; we prefer the term chosen ciphertext attack in the *pre-processing* mode, abbreviated *CCA-pre*. Here, the adversary may access a decryption oracle any polynomial (in the security parameter) number of times. Then the oracle is removed and a "challenge" ciphertext is given to the attacker.

- Chosen ciphertext in the sense of Rackoff and Simon [63]; we prefer the term chosen ciphertext attack in the *post-processing* mode, abbreviated *CCA-post*. This is defined formally in Section 3. The key point is that the attacker sees the challenge ciphertext *before* the oracle is removed, and can ask the oracle to decrypt any (possibly invalid) ciphertext *except the challenge*.

Our version of semantic security under chosen plaintext attack is the following: Let R be a relation computable in probabilistic polynomial time. 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 (including a specific time bound) 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 $m \in_R \mathcal{M}$ and $E_e(m)$ denotes the set $\{E_e(m, r)$ *s.t.* $|r| = p(n)\}$. In addition, $\mathcal{A}$

receives a "hint" (or history) about m in the form of hist(m), where hist is a polynomially computable function. $\mathcal{A}$ then produces a string β. *We assume that the prefix of β is the description of $\mathcal{M}$.* $\mathcal{A}$ is considered to have succeeded with respect to R if $R(m, \beta)$. Since β contains a description of $\mathcal{M}$, R is *aware* of $\mathcal{M}$ and may decide to accept or reject based on its description. This rules out achieving "success" by choosing a trivial distribution. 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 coin-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. 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.

Remark 2.1 *1. In their seminal paper on probabilistic encryption, Goldwasser and Micali separate the power of the adversary into two parts: a message finder that, intuitively, tries to find a pair of messages on which the cryptosystem is weak, and the line tapper, that tries to guess which of the two chosen messages is encrypted by a given ciphertext [43]. Accordingly, we have let $\mathcal{A}$ choose the message space $\mathcal{M}$, on which it will be tested. By letting $\mathcal{A}'$ choose $\mathcal{M}'$ (rather than "inheriting" $\mathcal{M}$ from $\mathcal{A}$), we are letting the simulator completely simulate the behavior of the adversary, so in this sense our definition is natural. A second reason for this choice is discussed in Section 3.4.3.*

2. As noted above, the fact that the description of $\mathcal{M}$ or $\mathcal{M}'$ is given explicitly to R prevents $\mathcal{A}'$ from choosing a trivial distribution, e.g. a singleton, since R can "rule out" such $\mathcal{M}$'s.

Definition 2.2 *A scheme $\mathcal{S}$ for public-key cryptosystems is* semantically secure with respect to relations *under chosen plaintext attack if for every probabilistic polynomial time adversary $\mathcal{A}$ as above there exists a probabilistic polynomial time adversary simulator $\mathcal{A}'$ such that for every relation $R(m, \beta)$ and function hist(m), both computable in probabilistic polynomial time, $|\,\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,|$ is subpolynomial.*

In this definition, the chosen plaintext attack is implicit in the definition of $\mathcal{A}$. This is a convention that will be followed throughout the paper.

Note the differences between our definition of semantic security with respect to relations and the original definition of semantic security [43]: in the original definition the challenge was to compute $f(x)$ given $E(x)$, where the function f is not necessarily even recursive. In contrast, here R is a relation and it is probabilistic polynomial time computable. Nevertheless, the two definitions are equivalent, as we prove in Theorem 2.2[4].

We prove the following theorem for the case of chosen plaintext attacks; the proof carries over to chosen ciphertext attacks in both the pre- and post-processing modes.

[4]The literature shows for 3 versions of semantic security and 3 corresponding versions of indistinguishability that each version of semantic security is equivalent to the corresponding version of indistinguishability [54, 36]. We are using a fourth version of indistinguishability – a uniform version of the non-uniform one-pass version in [54]. Equivalence of this definition to a corresponding version of semantic security has not been proved in the literature, but we conjecture it holds.

Theorem 2.2 *A public key cryptosystem is semantically secure with respect to relations under chosen plaintext attack if and only if it has the indistinguishability property.*

Proof. We first show that if the cryptosystem has the indistinguishability property, then it is semantically secure with respect to relations. 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 $\alpha_1, \alpha_2 \in_R \mathcal{M}$.

In the first experiment, $\mathcal{A}$ is given $\mathrm{hist}(\alpha_1)$ and $E_e(\alpha_1)$ and produces β_1. By definition, for any relation R

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

In the second experiment, $\mathcal{A}$ is given $\mathrm{hist}(\alpha_1)$ and $E_e(\alpha_2)$ and produces β_2. Let

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

Note that if R is probabilistic polynomial time computable and $\Pr[R(\alpha_1, \beta_1) \text{ holds}]$ differs polynomially from $\Pr[R(\alpha_1, \beta_2) \text{ holds}]$, where the probabilities are taken over the coin flips by R, and the random bits used in generating the encryptions (but not over the choice of e, $\mathcal{M}$, and α_1 and α_2), then we can create a distinguisher for encryptions of α_1 and α_2 under encryption key e, so in particular, given e, we have found a pair of messages whose encryptions are easy to distinguish. Thus, with overwhelming probability over choice of e, $\mathcal{M}$, and α_1, α_2, the individual probabilities (with fixed e) are close. It follows that the probabilities $\pi(\mathcal{A}, R)$ and χ (which are aggregated over choice of e) are also close.

For the third experiment, consider an $\mathcal{A}'$ that, on input e, 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 $\mathrm{hist}(\alpha)$ for an $\alpha \in_R \mathcal{M}$. $\mathcal{A}'$ generates $\alpha' \in_R \mathcal{M}$ and gives to the simulated $\mathcal{A}$ the hint $\mathrm{hist}(\alpha)$ and the encryption $E_e(\alpha')$. The simulated $\mathcal{A}$ 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 given the public-key can find two message (m_0, m_1) for which it can distinguish encryptions of m_0 from encryptions of m_1. The specification of $\mathcal{A}$ is as follows: Given a key e, $\mathcal{A}$ runs M to obtain (m_0, m_1). Let $\mathcal{M} = \{m_0, m_1\}$, where m_0 and m_1 each has probability $1/2$, be the message distribution on which $\mathcal{A}$ is to be tested. The function hist is the trivial $\mathrm{hist}(x) = 1$ 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 plus the description of $\mathcal{M}$. The relation R that witnesses the fact that the cryptosystem is not semantically secure with respect to relations is equality plus a test of consistency with $\mathcal{M}$. Recall that the description of $\mathcal{M}$ is provided explicitly and hence R can also check that $\mathcal{M}$ is of the right form. Since M is by assumption a distinguisher, having access to the ciphertext γ gives $\mathcal{A}$ a polynomial advantage at succeeding with respect to R over any $\mathcal{A}'$ that does not have access to the ciphertext (which has probability $1/2$). $\square$

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 [36, 43, 54] that the notions of of (traditional) semantic security, indistinguishability and semantically secure with respect to relations are all equivalent.

String Commitment

The literature discusses two types of bit or string commitment: *computational* and *information* theoretic. These terms describe the type of secrecy of the committed values offered by the scheme. In computational bit commitment there is only one possible way of opening the commitment. Such a scheme is designed to be secure against a probabilistic polynomial time receiver and an arbitrarily powerful sender. In information theoretic commitment it is possible to open the commitment in two ways, but the assumed computational boundedness of the sender prevents him from finding the second way. Such a scheme is designed to be secure against an arbitrarily powerful receiver and a probabilistic polynomial time prover. We restrict our attention to computational 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 this stage B has some information that represents α, but B should gain no information on the value of α from the messages exchanged during this stage.

- The *reveal* stage: at the end of this stage B knows α. There should be only one string that A can reveal.

The two requirements of a string commitment protocol are *binding* and *secrecy*. Binding means that following the commit stage the A can reveal at most one string. In our scenario we require the binding to be unconditional, but probabilistic: with high probability over B's coin-flips, following the commit stage there is at most one string that B accepts (as the value committed) in the reveal stage.

The type of secrecy we require is semantic security. We specify what this means, using the notions of security with respect to relations (however, as above, it is equivalent to the "traditional" way of defining semantic security). Let $\mathcal{A}$ be an adversary that produces a distribution $\mathcal{M}$ on strings of length $\ell(n)$ computable in probabilistic polynomial time. A string $\alpha \in_R \mathcal{M}$ is chosen and $\mathcal{A}$ receives $\text{hist}(\alpha)$, where hist is a probabilistic polynomial time computable function. The commitment protocol is executed where $\psi(A)$ follows the protocol and $\psi(B)$ is controlled by $\mathcal{A}$. The adversary $\mathcal{A}$ then produces a string β. We assume that the prefix of β is the description of $\mathcal{M}$.

$\mathcal{A}$ is considered to have succeeded with respect to R if $R(\alpha, \beta)$. Let $\pi(\mathcal{A}, R)$ be the probability that $\mathcal{A}$ succeeds with respect to R. The probability is over the coin-flips of $\mathcal{A}$, and the choice of α.

For the second probability, we have an adversary simulator $\mathcal{A}'$ who will not have access to the $(\psi(A), \psi(B))$ execution of the string commitment protocol. $\mathcal{A}'$ chooses a distribution $\mathcal{M}'$. An $\alpha \in_R \mathcal{M}'$ and $\text{hist}(\alpha)$ is given to $\mathcal{A}'$. $\mathcal{A}'$ produces β. As above, $\mathcal{A}'$ is considered to have succeeded with respect to R if $R(\alpha, \beta)$.

Definition 2.3 *A commitment scheme is* semantically secure with respect to relations *if for every probabilistic polynomial time adversary $\mathcal{A}$ as above there exists a probabilistic polynomial time adversary simulator $\mathcal{A}'$ such that for every probabilistic polynomial time computable relation $R(\alpha, \beta)$ and function hist(m) computable in probabilistic polynomial time $|\, \pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,|$ is subpolynomial.*

Zero-Knowledge Interaction We next present a generalization of a (uniform) zero-knowledge interactive proof of language membership.

Let $(A, B)[a, b]$ be an interactive protocol, where (a, b) belongs to a set Π of legal input pairs to A and B. (In the special case of zero-knowledge proofs of language membership, the valid pairs (a, b) have the property that the prefixes of a and b are the common input $x \in L$.) Roughly speaking, we say that (A, B) is *zero-knowledge* with respect to B if for every polynomial time bounded B', there exists a simulator that can produce conversations between (A, B') which are indistinguishable from the actual (A, B') conversation. More accurately, and pursuing the terminology of this section, let $\mathcal{A}$ be an adversary that controls $\psi(B)$. $\mathcal{A}$ chooses a joint distribution $\mathcal{D}$, consistent with Π, on $[a, b]$, and then a pair $[a, b]$ is drawn according to $\mathcal{D}$, $\psi(A)$ gets a, $\psi(B)$ gets b, and the interaction proceeds by $\psi(A)$ following the protocol (while $\psi(B)$'s actions are controlled by $\mathcal{A}$). The result is a transcript T of the conversation between $\psi(A)$ and $\psi(B)$. $\mathcal{A}$ also produces a string σ which contains as a prefix the description of $\mathcal{D}$ (and may contain such information as the state of $\psi(B)$ at the end of the protocol).

Let R be a ternary relation. $\mathcal{A}$ is considered to have succeeded with respect to R if $R([a, b], T, \sigma)$. Let $\pi(\mathcal{A}, R)$ be the probability that $\mathcal{A}$ succeeds with respect to R. The probability is over the coin-flips of $\mathcal{A}$, the coin-flips of $\psi(A)$ and the choice of $[a, b]$.

On the other hand, we have $\mathcal{A}'$ that selects $\mathcal{D}'$ consistent with Π. A pair $[a, b]$ is then drawn according to $\mathcal{D}'$ and $\mathcal{A}'$ receives b. $\mathcal{A}'$ produces a transcript T' and a string σ'. $\mathcal{A}'$ is considered to have succeeded with respect to R if $R([a, b], T', \sigma')$. Let $\pi(\mathcal{A}', R)$ be the probability that $\mathcal{A}$ succeeds with respect to R. The probability is over the coin-flips of $\mathcal{A}'$ and the choice of $[a, b]$.

Definition 2.4 *A protocol (A, B) is zero-knowledge with respect B if for all probabilistic polynomial time adversaries $\mathcal{A}$ as above there exists a probabilistic polynomial time adversary simulator $\mathcal{A}'$ such that for every relation R computable in probabilistic polynomial time $|\, \pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,|$ is subpolynomial.*

(If $(a, b) \notin \Pi$ then zero-knowledge is not ensured, but other requirements may hold, depending on the protocol.)

Two interesting examples of zero-knowledge interaction are proof of language membership [44, 40] and proofs of knowledge [31]. Both of these can be based on the existence of string commitment protocols.

Non-Interactive Zero-Knowledge Proof Systems

An important tool in the construction of our public-key cryptosystem are non-interactive zero-knowledge proof systems. The following explanation is taken almost verbatim from [61]: 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 [12, 13]. A non-interactive zero-knowledge scheme for proving membership in any language in NP which may be based on *any* trapdoor permutation is described in [32]. Recently, Kilian and Petrank [49, 50] found more efficient implementations of such schemes. Their scheme is for the circuit satisfiability problem. Let k be a security parameter. 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 on strings of length polynomial in n and k, where the polynomial depends on the particular protocol, 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 [13, 32] 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

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

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 [12], modified to incorporate the tractability of $\mathcal{P}$. The uniformity conditions of the system are adopted from Goldreich [35].

Definition 2.5 *A triple $(\mathcal{P},\mathcal{V},\mathcal{U})$, where $\mathcal{P}$ is a probabilistic polynomial time 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.

In the construction of the non-malleable cryptosystem in Section 3, non-interactive zero-knowledge proof systems are used to prove that encryptions generated under *independent* keys correspond to the *same* plaintext. This is similar to their application in [61].

2.2 Definitions Specific to Non-Malleability

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. The intended value is a generalization of the notion of an input. Indeed, when $\psi(A)$ is non-faulty we may refer to the intended value as an *input* to A. However, we do not know how to define the input to a faulty processor that can, for example, refuse to commit to it. In this case we may need to substitute in a default value. The term *intended value* covers cases like this.

We 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 $\text{hist}(\alpha)$, 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 all 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 $\psi(A), \psi(B), \psi(C)$ and $\psi(D)$ 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 – *e.g.*, fails to respond to a query, is caught cheating, or produces invalid ciphertexts – we take β to be all zeros.

We treat "copying" slightly differently in the context of encryption, which is non-interactive, and in the commitment and zero-knowledge settings, which are interactive. In particular, our results are stronger for encryption, since our construction rules out anything but exact copying of the ciphertext. Thus, seeing the ciphertext does not help the adversary to construct a different encryption of the same message. In the interactive setting we only ensure that if $\alpha \neq \beta$, then the two values are unrelated. We use identities (chosen by the users and not enforced provided by any authentication mechanism) to force α and β to be different. In particular, if the adversary wishes to be a transparent intermediary, then we do not bother to rule out the case in which the adversary commits to or proves exactly the same string as A does, even if it gives a different commitment (to the same value) or a different proof (of the same theorem).

We now formally define the non-malleability guarantee in the interactive setting. A *relation approximator* R is a probabilistic polynomial time Turing machine taking two inputs[5] 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. In the interactive settings, we restrict our attention to the special class of relation approximators which on input pairs of the form (x, x) always output zero. The intuition here is that we cannot rule out copying, but intuitively this is not the cases in which the adversary "succeeds."

When we discuss composition (or parallel execution) 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 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 $\psi(C)$ sends the same value as one of the non-faulty players (in the case of encryption this is ruled out by the definition of the adversary).

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 $\mathrm{hist}(\alpha)$).

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, $\mathrm{hist}(\alpha)$, and $\mathcal{D}$, produces an intended value γ.

[5]Sometimes we will need R to take three inputs, the third being in plaintext.

Consider 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 Π is the set of legal input pairs to the two parties executing $\mathcal{S}$. $\mathcal{A}$ may choose any probabilistic polynomial time sampleable distribution $\mathcal{D}$ on the joint distribution to all four players,

$$\psi(A), \psi(B), \psi(C), \psi(D),$$

respectively, where the inputs to $\psi(A)$ and $\psi(B)$ are consistent with Π. Let $(\alpha, x, y, \delta) \in_R \mathcal{D}$. For any 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 $\mathcal{A}$, given x, y, $\text{hist}(\alpha)$, and participation in the (A,B) execution in which $\psi(A)$ sends α, causes $\psi(C)$ to send β in the (C,D) execution, such that $R(\alpha,\beta)$ outputs 1, *under some specified form of attack* (Since $\psi(C)$ is under control of the adversary there is no reason that β should equal y.)

Similarly, for an adversary simulator $\mathcal{A}'$ choosing a joint distribution $\mathcal{D}'$ for all four players where the inputs to $\psi(A)$ and $\psi(B)$ are consistent with Π, for $(\alpha, x, y, \delta) \in_R \mathcal{D}'$, let $\mathcal{A}'$ have access to x, y, and $\text{hist}(\alpha)$, and let $\mathcal{A}'$ send γ. Let $\pi'(\mathcal{A}', R)$ denote the probability, taken over the the choices made of $\mathcal{A}'$, and the choices of R, that $R(\alpha, \gamma) = 1$.

Definition 2.6 *A scheme $\mathcal{S}$ for a primitive $\mathcal{P}$ is non-malleable with respect to itself under a given type of attack G, if for all adversarially coordinated systems $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$ where $(A,B) = (C,D) = \mathcal{S}$, where $\mathcal{A}$ mounts an attack of type G, there exists an adversary simulator $\mathcal{A}'$ such that for all relation approximators R, $|\,\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,|$ is subpolynomial*[6]*.*

This definition is applicable to *all* three primitives. As stated above, the precise attack against the system is crucial to the definition of $\mathcal{A}$ and hence of $\pi(\mathcal{A}, R)$. In particular, when we discuss encryption in Section 3, we will specify the nature of the adversary precisely. The definition makes sense for all types of attack, with the appropriate choices of $\pi(\mathcal{A}, R)$. Finally, we must specify the "unit" which we are trying to protect, *i.e.*, is it a single encryption or several.

Remark 2.3 *There are three possible interpretations of Definition 2.6, according to the running time of $\mathcal{A}'$:*

1. *$\mathcal{A}'$ runs in fixed polynomial time; this is* strict *non-malleability (we usually drop the appellation "strict").*

2. *$\mathcal{A}'$ runs in* expected *polynomial time; in accordance with Goldreich's taxonomy for zero-knowledge, we call this* liberal *[35] non-malleability.*

3. *For every ε there exists $\mathcal{A}'$ running in time polynomial in n and ε^{-1} such that $|\,\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,| < \varepsilon$; this is* ε-malleability*, this time in analogy to ε-knowledge ([42]; see also [29]).*

Our public-key cryptosystem is strictly non-malleable. M. Fischlin and R. Fischlin have pointed out that we do not prove strict non-malleability in our commitment scheme; however, we prove both liberal non-malleability and ε-malleability.

[6] In the previous version of this paper the order of quantifiers was $\forall R \forall \mathcal{A} \exists \mathcal{A}'$, yielding a possibly weaker definition. However, all the constructions in our work satisfy the stronger order of quantifiers given here. Now all our definitions share a common order of quantifiers.

2.3 System Model

We assume a completely asynchronous model of computing. For simplicity, we assume FIFO communication links between processors (if the links are not FIFO then this can be simulated using sequence numbers). We *do not* assume authenticated channels.

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 [63] propose a model in which each sender possesses a secret associated with a *publicly known* identifying key issued by a trusted center.

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 $\mathcal{S}$ is non-malleable and $\alpha \neq \beta$, then β's dependence on α is limited to dependence on $\text{hist}(\alpha)$. 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 private 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. As was done by [45] in 1984 in the context of digital signatures, 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.

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 [63] and which we call *chosen ciphertext in the post-processing mode (CCA-post)*: 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. Indeed, (plain) RSA [64] and the implementation of probabilistic encryption based on quadratic residuousity [43] are insecure against a chosen ciphertext postprocessing attack.

Malleability, as defined in Section 2.2 specifies what it means to "break" the cryptosystem. Informally, given a relation R 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 under a given attack G if for every $\mathcal{A}$ mounting an attack of type G, 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 $R(\alpha,\gamma)=1$. Given the notion of semantic security with respect to relations and Theorem 2.2, non-malleability is clearly an extension of semantic security. See Section 3.4.2 for the relationship between non-malleability and the type of attack.

We now define precisely the power of the CCA-post 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 $\mathcal{A}$ receives 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 $c \in_R E_e(m)$ 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 $c \neq c'_1, c'_2, \ldots$. At the end of the process, $\mathcal{A}$ produces a polynomially bounded length vector of ciphertexts $(f_1, f_2, \ldots)$ not containing the challenge ciphertext c, with each $f_i \in E_e(\beta_i)$, and a *cleartext* string σ which we assume contains a description of $\mathcal{M}$[7]. Let $\beta = (\beta_1, \beta_2, \ldots)$. $\mathcal{A}$ is considered to have succeeded with respect to R if $R(m, \beta, \sigma)$. (We separate β from σ because the goal of the adversary is to produce encryptions of the elements in β.) Let $\pi(\mathcal{A}, R)$ be the probability that $\mathcal{A}$ succeeds where the probability is over the coin-flips of the key generator, $\mathcal{A}, \mathcal{M}$ and the encryption of m.

Let $\mathcal{A}'$ be an adversary simulator that does not have access to the encryptions or to the decryptions, but can pick the distribution $\mathcal{M}'$. On input e, $\mathcal{A}'$ produces $\mathcal{M}'$ and then $m \in_R \mathcal{M}'$ is chosen. $\mathcal{A}'$ receives $\text{hist}(m)$ and without the benefit of the chosen ciphertext attack should produce a vector of ciphertexts $(f_1, f_2, \ldots)$, where each $f_i \in E_e(\beta_i)$, and a string σ containing $\mathcal{M}'$. 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 where the probability is over the coin-flips of the key generator, $\mathcal{A}'$ and $\mathcal{M}'$.

[7]In the public key context σ serves no purpose other than providing the description of $\mathcal{M}$ as an input to R, since in this situation from any plaintexts $p \in \mathcal{M}$ that are part of σ it is always possible to compute an encryption of σ, so we could always add an additional $f_i \in E_e(p)$ to our vector of ciphertexts. However, we introduce the possibility of including plaintexts p in σ so that the definition can apply to symmetric, or private key, encryption.

Note that $\mathcal{A}'$ has a lot less power than $\mathcal{A}$: not only does it not have access to the ciphertext encrypting α, but it cannot perform *any* type of chosen ciphertext attack, even in choosing the distribution $\mathcal{M}'$. Note also that as in the definition of semantically secure with respect to relations, the fact that $\mathcal{M}$ is given to R prevents $\mathcal{A}'$ from choosing trivial distributions.

Definition 3.1 *A scheme $\mathcal{S}$ for public-key cryptosystems is non-malleable with respect to chosen ciphertext attacks in the post-processing mode, if for all probabilistic polynomial time adversaries $\mathcal{A}$ as above there exists a probabilistic polynomial time adversary simulator $\mathcal{A}'$ such that for all relations $R(\alpha, \beta, \sigma)$ computable in probabilistic polynomial time, $|\,\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)\,|$ is subpolynomial.*

Note that the definition does not require R to be restricted (to a relation approximator) as described in Section 2.2.

An illustration of the power of non-malleability under CCA-post attacks is presented in Section 3.5, where 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 (and perhaps shouldn'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.

Simple Ideas That Do Not Work

A number of simple candidates for non-malleable cryptosystems come to mind. Let E be a cryptosystem semantically secure against a chosen ciphertext attack. Assume for concreteness that A wishes to send the message m and B wishes to send "1 + the value sent by A". That is, B, without knowing m, wishes to send $m + 1$.

One "solution" would be to append to $E(m)$ a non-interactive zero-knowledge proof of knowledge of the encrypted value m. The problem with this approach is that the proof of knowledge may itself be malleable: conceivably, given $E(m)$ and a proof of knowledge of m, it may be possible to generate $E(m + 1)$ and a proof of knowledge of $m + 1$.

Another frequently suggested approach is to sign each message. Thus, to send a message m, party A sends $(E(m), S_A(E(m)))$, where S_A is a private signing algorithm for which a public verification key is known. There are two problems with this: first, it assumes that *senders* as well as *receivers* have public keys; second, it misses the point: if E is malleable then B, seeing $(E(m), S_A(E(m)))$, simply ignores the second component, generates $E(m + 1)$, say, based on $E(m)$, and sends $(E(m + 1), S_B(E(m + 1)))$.

Yet another suggestion is to put the signature *inside* the ciphertext: A sends $E(m \circ S_A(m))$. This still suffers from the assumption that A has a public verification key corresponding to S_A, and it again misses the point: B is not trying to produce $E(m+1, S_A(m+1))$, but only $E(m + 1 \circ S_B(m + 1))$. The unforgeability properties of S_A say absolutely nothing about B's ability to produce an encryption of $S_B(m + 1)$.

One more suggestion is to append an ID to each message and send, for example, $E(A \circ m)$. Again, we do not know how to show that, based only on the semantic security of E against chosen ciphertext attack, seeing $E(A \circ m)$ does not help B to produce $E(B \circ m)$ or $E(B \circ m + 1)$.

Overview of the scheme

The public key consists of 3 parts: a collection of n pairs of keys $\langle e_i^0,\ e_i^1\rangle$, $i = 1, \ldots, n$, a random string U for providing zero-knowledge proofs of consistency in a non-interactive proof system and a universal one-way hash function. U is uniformly distributed because it is to the advantage of its creator (the verifier in the non-interactive zero-knowledge proof) that it should be so.

The process of encryption consists of 4 parts.

1. An "identity" is chosen for the message by creating a public signature verification key; the corresponding signing key is kept private. The signing key is only used to sign a single message, so a one-time signature scheme may be used here.

2. The message is encrypted under several encryption keys chosen from $\langle e_i^0,\ e_i^1\rangle$, $i = 1, \ldots, n$, as a function of the public signature verification key chosen in the first step. The selection is made by hashing the public signature verification key using the universal one-way hash function that is part of the public key for the cryptosystem.

3. A (non-interactive zero-knowledge) proof of consistency is provided, showing that the value encrypted under all the selected keys is the same one.

4. The encryptions and the proof are signed using the private signing key chosen in the first step.

When a message is decrypted, the signature verification key comprising the identity is used to verify that the signature is valid; the proof of consistency of encryptions is also checked. Only then is the (now well defined) plaintext extracted.

The hash function is used only for efficiency; without it we would have to increase n, the number of encryption key pairs $\langle e_i^0,\ e_i^1\rangle$ in the public key for the cryptosystem. Thus, intuitively, the hash function plays the a role analogous to the usual role of a hash function in an implementation of a signature scheme; however, we use it to hash the public verification key of the (freshly chosen) signature scheme, rather than the text of a message to be signed. As we will see, the critical point is that every identity chosen yields a distinct set of keys under which consistent encryptions must be created.

The idea of encrypting under several keys and proving consistency appeared in [61]. However, in [61] every plaintext bit is encrypted under *every* public key (there are only two), while here each identity for a message yields a distinct set of keys. Thus, the main changes here to the scheme in [61] are

1. the addition of an "identity" for each message to select a distinct set of keys;

2. using a (hash of the) freshly-chosen public signature verification key as the identity;

3. signing the encryptions under the selected keys and the proof of consistency with the (secret) signing key that corresponds to the identity.

To develop some intuition for how the identities are used, consider a hypothetical situation in which all they keys in the pairs $\langle e_i^0,\ e_i^1\rangle$, $i = 1, \ldots, n$, are completely malleable, and suppose further that given a NIZK that one set of encryptions is consistent, it is easy to generate a proof of the true theorem that a set of related encryptions is also consistent.

If (as is the case in [61], where only two keys are used) we were *not* to use signatures and we were *not* to select a new set of keys for each message (so that an encryption $E(m)$ would be a consistent set of encryptions under *all* the e_i^0 and e_i^1, $i = 1, \ldots, n$, and a proof of consistency), then given an encryption $E(m)$, creating an encryption, say of $2m$, would be easy: use the assumed malleability of all the $e_i^b(m)$ to create encryptions $e_i^b(2m)$, and use the assumed malleability of the NIZK to prove (the true theorem) that the resulting set of encryptions is consistent. We combat this hypothetical attack (which we cannot rule out!) using the identities, as we now describe.

Consider an attacker that has an encryption $\alpha \in_R E(m)$ under our scheme, and wishes to create from it an encryption $\beta \in E(2m)$. Suppose, as above, that the encryption functions e_i^b are completely malleable and that the NIZKs are malleable in the sense previously described. In our case, the attacker must create an identity for the message. Remember that an identity is the public verification key for a signature scheme. The attacker can choose to use the same identity (signature verification key) as in α or a different one. If the identity is preserved, this means that the attacker is using the public signature verification key appearing in α, for which he does not know the corresponding signature key. In this case, while the attacker can exploit the malleability of the e_i^b and the NIZK, in the last step of the encryption process he must forge a signature on the *new* encryptions and *new* proof of consistency – which he cannot do because he does not know the private signing key. On the other hand, if the attacker selects a new identity for β, different than the one used in α, then, since the identity selects the keys e_i^b under which the message is encrypted, for some i the attacker will have in α only $e_i^b(m)$ (and he will *not* have $e_i^{1-b}(m)$), but he will need to create $e_i^{1-b}(2m)$, so there will be no way to exploit the malleability of the encryption schemes e_i^0 and e_i^1. To summarize, non-malleability comes from the fact that the choice of the subsets and the signature each authenticate the other.

As in [61], anyone can decide whether a ciphertext is legitimate, *i.e.,* decrypts to some meaningful message, by verifying the NIZK proof of consistency and checking, using the signature verification key that comprises its identity, that the message is correctly signed. Thus, no information is ever gained during an attack when the decrypting mechanism rejects an invalid ciphertext.

Intuitively, given $E(\alpha)$, an attacker with access to a decryption mechanism can generate a legal ciphertext $E(\beta)$ and learn β, but non-malleability implies that an adversary simulator can generate $E(\gamma)$ without access to $E(\alpha)$, where γ is distributed essentially as β is distributed. Thus β is unrelated to α (non-malleability), and learning β yields no information about α (semantic security).

3.1 The Tools

We require a probabilistic public key cryptosystem that is semantically secure (see Section 2.1). 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 $L = \{e_1, e_2, \ldots e_n, w | w$ 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 2.1. Recall that the system consists of a prover, a verifier, and a common random string U known to both the prover and the verifier and that such a scheme can be based on any trapdoor permutation. Note that the length of U depends only on the security parameter and not on the number of messages to be encrypted over the lifetime of this public key.

The cryptosystem uses a *universal family of one-way hash functions* as defined in [60]. 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 [65] 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 [5, 45, 60].

3.2 The Non-Malleable Public-Key Encryption Scheme

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

Key generation.

1. Run $GP(1^n)$, 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 reference string U.
3. Generate $h \in_R H$.

The public encryption key is
$$\langle h, e_1^0, e_1^1, e_2^0, e_2^1, \ldots, e_n^0, e_n^1, U \rangle$$
and 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 $GS(1^n)$, 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 the n-bit string $v_1 v_2 \ldots v_n$.
3. For each $1 \leq i \leq k$

(a) For $1 \leq j \leq n$

   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 $1 \leq i \leq k$ 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 $1 \leq i \leq k$ 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 Under CCA-Post Attack

We now prove the non-malleability of the public key encryption scheme $\mathcal{S}$ under a chosen ciphertext post-processing attack. We define a related scheme $\mathcal{S}'$ whose (malleable) semantic security with respect to relations under chosen *plaintext* attack is straightforward. We then argue that the semantic security of $\mathcal{S}'$ under chosen plaintext attack implies the non-malleability of $\mathcal{S}$ under chosen ciphertext post-processing attack.

The Cryptosystem $\mathcal{S}'$:

1. Run $GP(1^n)$, 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 $m = b_1, b_2, \ldots b_k$

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

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

5. To decrypt an encryption $\langle \alpha_1, \ldots, \alpha_n \rangle$, compute $m_j = D_{d_j}(\alpha_j)$ for $1 \leq j \leq n$. If $m_1 = m_2 = \ldots = m_n$ 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 $\mathcal{B}$ that, on being given an encryption under $\mathcal{S}'$, generates an encryption under $\mathcal{S}$. As above, 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').

Notation. In the sequel, adversaries $\mathcal{A}$ and $\mathcal{A}'$ are adversaries against the scheme $\mathcal{S}$. Adversaries $\mathcal{B}$ and $\mathcal{B}'$ are adversaries against the system $\mathcal{S}'$.

Procedure for $\mathcal{B}$: Given a public key $E' = \langle e_1, \ldots, e_n \rangle$ 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 a proof of consistency of n encryptions that will be given to it later on).

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

4. Run $GS(1^n)$ 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 $x_1, x_2, \ldots$. For each x_i produced by $\mathcal{A}$, $\mathcal{B}$ verifies the signatures and the proofs of consistency. If these verifications succeed, $\mathcal{B}$ decrypts x_i by using one of the new decryption keys generated in Preprocessing 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. $\mathcal{B}$ outputs $\mathcal{M}$. We will show that the semantic security of $\mathcal{S}'$ with respect to relations under chosen plaintext attack implies the non-malleability of $\mathcal{S}$ under chosen ciphertext post-processing attack.

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

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

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

Lemma 3.2 *Let $\mathcal{A}$ be an adversary attacking the original scheme $\mathcal{S}$. On input E' and $c' \in_R E'(m)$, let E be generated by $\mathcal{B}$ as above, and let c be the encryption of m under E created by $\mathcal{B}$ in Simulation Step 3. 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 verification key in ζ), then $\mathcal{B}$ can decrypt ζ.*

Proof. Let F' be the public signature verification 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 Preprocessing Step 1 of the procedure for $\mathcal{B}$ will be used in ζ. Since $\mathcal{B}$ generated this encryption key and its corresponding decryption key, $\mathcal{B}$ can decrypt.

We now argue that $F' \neq F$ (that is, that we must be in the previous case). Since $\mathcal{A}$ has not seen F or anything depending on F during its chosen ciphertext attack in Step 1 of the Simulation Phase, the probability that $\mathcal{A}$ uses $F' = F$ in a ciphertext during this step is negligible. Suppose for the sake of contradiction that after it has seen F in the target ciphertext c, $\mathcal{A}$ uses $F' = F$ in ζ. Then by the security of the signature scheme, only the original ciphertexts and proofs of consistency $(c_1, p_1) \ldots (c_n, p_n)$ from Preprocessing Step 2 and Simulation Step 3 can be signed; otherwise $\mathcal{A}$ could be used to break the signature scheme. This forces $\zeta = c$, contradicting the fact that $\zeta \neq c$. $\square$

Note that in Step 3 of the Simulation Phase the vector c' is a legitimate encryption under E' and therefore is a vector of consistent encryptions, so the simulated non-interactive proof of consistency is a proof of a true theorem. Note also that this is the only place in which a proof is simulated by $\mathcal{B}$. Thus, even though the shared random string is used to generate many proofs of consistency during the lifetime of the public key, the zero-knowledge property we will need for the proof is only for a single theorem, since the only simulated proof will be on the target ciphertext.

The next lemma says that, as in the Naor-Yung scheme, $\mathcal{A}$ cannot distinguish the "instance" of $\mathcal{S}$ concocted by $\mathcal{B}$ from a real instance of $\mathcal{S}$, so $\mathcal{A}$ is just as likely to break the concocted instance.

For any probabilistic polynomial time relation R, let $\pi(\mathcal{B}, R)$ denote the probability that $\mathcal{B}$, using $\mathcal{A}$ as described in the Simulation Phase, generates a vector of plaintexts $(\beta_1, \beta_2, \ldots)$ and a string σ such that $R(m, \beta, \sigma)$ holds, where $\beta = (\beta_1, \beta_2, \ldots)$. By choice of $\mathcal{B}$, $\pi(\mathcal{B}, R)$ is exactly the probability that $\mathcal{A}$ breaks $\mathcal{S}$ with respect to R in the Simulation phase: $\mathcal{A}$ (interacting with $\mathcal{B}$), generates σ and a vector of encryptions $(E(\beta_1), E(\beta_2), \ldots)$; by Lemma 3.2, $\mathcal{B}$ can decrypt these values, and so outputs β_i, $i = 1, 2, \ldots$, together with σ.

Lemma 3.3 *For any probabilistic polynomial time relation R, let $\pi(\mathcal{B}, R)$ denote the probability that $\mathcal{B}$, using $\mathcal{A}$ as described in the Simulation Phase, generates a vector of plaintexts $(\beta_1, \beta_2, \ldots)$ and a string σ such that $R(m, \beta, \sigma)$ holds, where $\beta = (\beta_1, \beta_2, \ldots)$. Let $\pi(\mathcal{A}, R)$ denote the probability that $\mathcal{A}$ breaks a random instance of $\mathcal{S}$ with respect to R. Then $\pi(\mathcal{B}, R)$ and $\pi(\mathcal{A}, R)$ are subpolynomially close.*

Proof. As noted above, $\pi(\mathcal{B}, R)$ is exactly the probability that $\mathcal{A}$ breaks $\mathcal{S}$ with respect to R in the Simulation phase. The only difference between the instance of $\mathcal{S}$ generated by $\mathcal{B}$ and an instance of $\mathcal{S}$ generated at random is in the reference string U and the proof of consistency for the target ciphertext: in the former case these are produced by the simulator (Steps 2 and 3 of the Simulation Phase) and in the latter case they are authentic. The lemma therefore follows immediately from the definition of non-interactive zero knowledge (Definition 2.5): any difference between the two probabilities can be translated into an ability to distinguish a simulated proof from a true proof. $\square$

Theorem 3.4 *The public-key encryption scheme $\mathcal{S}$ is non-malleable against chosen ciphertexts attacks in the post-processing mode.*

Proof. Let $\mathcal{A}$ be any polynomially bounded adversary and assume for the sake of contradiction that $\mathcal{A}$ and the probabilistic polynomial time computable relation R witness the malleability of $\mathcal{S}$ under a chosen ciphertext post-processing attack. We will use the semantic security of $\mathcal{S}'$ with respect to relations to derive a contradiction by exhibiting an adversary simulator $\mathcal{A}'$ that, without access to the target ciphertext and without mounting any kind of chosen ciphertext attack against $\mathcal{S}$, does (negligibly close to) as well as $\mathcal{A}$ at breaking $\mathcal{S}$ (in the malleability sense).

Let E' be an encryption key in $\mathcal{S}'$. Let $\mathcal{B}$ be as described above. $\mathcal{B}$ generates an encryption key E in $\mathcal{S}$, invokes $\mathcal{A}$ on E to obtain a message distribution $\mathcal{M}$, and outputs $\mathcal{M}$. $\mathcal{B}$ is then given a ciphertext $c' = E'(m)$, for $m \in_R \mathcal{M}$, generates 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, $\mathcal{B}$ can extract the β_i. Let $\beta = (\beta_1, \beta_2, \ldots)$. Let R be any probabilistic polynomial time computable relation. By Lemma 3.3, the probability that $R(m, \beta, \sigma)$ holds is subpolynomially close to $\pi(\mathcal{A}, R)$.

Recall the definition of semantically secure with respect to relations: There exists a procedure $\mathcal{B}'$ that "does as well" as $\mathcal{B}$ at producing messages related to m, in the following sense. On input $(E', 1^n)$, $\mathcal{B}'$ outputs a message distribution $\mathcal{M}'$; $m' \in_R \mathcal{M}'$ is selected, but $\mathcal{B}'$ is not given access to the a ciphertext for m', just the hint $\text{hist}(m')$. $\mathcal{B}'$ generates β', σ'.

The definition of semantic security with respect to relations guarantees that for every $\mathcal{B}$ there exists $\mathcal{B}'$ such that $|\Pr[R(m, \beta, \sigma)] - \Pr[R(m', \beta', \sigma')]| \le \nu(n)$, where the probabilities are over the choice of public key, the coin flips of $\mathcal{B}$ and $\mathcal{B}'$ respectively, the choices of m and m', and the coin flips in creating the encryption of m. Note that we are *re-randomizing* the public key: it is chosen afresh for each probability.

We use $\mathcal{B}'$ to define the adversary simulator $\mathcal{A}'$ whose existence is mandated by the definition of non-malleability under chosen ciphertext post-processing. On input $(E, 1^n)$, $\mathcal{A}'$ ignores E and selects a public key for an instance E' of $\mathcal{S}'$ (with security parameter n). It then runs $\mathcal{B}'$ on $(E', 1^n)$ to select a message space $\mathcal{M}'$. $\mathcal{A}'$ outputs $\mathcal{M}'$. A message $m' \in_R \mathcal{M}'$ is chosen, and $\mathcal{A}'$ is given hist(m'), which it forwards to $\mathcal{B}'$. $\mathcal{B}'$ outputs (β', σ'), where $\beta' = (\beta'_1, \beta'_2, \ldots)$ is a vector of *plaintexts*. $\mathcal{A}'$ outputs the vector of *encryptions* $E(\beta) = (E(\beta'_1), E(\beta'_2), \ldots)$, together with σ'.

Clearly $\pi'(\mathcal{A}', R) = \pi'(\mathcal{B}', R)$, where the first term is the probability that $\mathcal{A}'$ succeeds at producing $E(\beta', \sigma')$ such that $R(m', \beta', \sigma')$ and the second term is the probability that $\mathcal{B}'$ succeeds at the same task.

By choice of $\mathcal{B}'$ $|\pi'(\mathcal{B}', R) - \pi(\mathcal{B}, R)|$ is negligible. This, together with Lemma 3.3 and the fact that $\pi'(\mathcal{A}', R) = \pi'(\mathcal{B}', R)$, implies that $|\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)| \le \nu(n)$. Therefore $(\mathcal{A}, R)$ cannot witness the malleability of $\mathcal{S}$. $\square$

Corollary 3.5 *If there exists a public-key cryptosystem semantically secure against chosen plaintext attack and if non-interactive zero-knowledge satisfying the requirements of Definition 2.5 is possible, then there exists a non-malleable public-key cryptosystem secure against chosen ciphertexts attacks in the post-processing mode. In particular, if trapdoor permutations exist, then such cryptosystems exist.*

An interesting open problem is whether one can rely on the existence of a public-key cryptosystem semantically secure against chosen plaintext attacks alone to argue that non-malleable public-key cryptosystems secure against chosen ciphertexts attacks in the postprocessing mode exist. Two assumptions that are known to be sufficient for semantically secure public-key cryptosystems secure against plaintext attacks, but where the existence of the stronger kind of cryptosystems is not clear are the hardness of the Diffie-Hellman (search) problem and the unique shortest vector problem (used in the Ajtai-Dwork cryptosystem [1]).

3.4 Remarks

3.4.1 On Vectors of Encryptions

1. We have defined non-malleable public key encryptions to cover the case in which $\mathcal{A}$ produces a vector of encryptions $(E(\beta_1), \ldots, E(\beta_n))$, having been given access to only a *single* $E(\alpha)$. It is natural to ask, what happens if $\mathcal{A}$ is given access to to encryptions of *multiple* α's, $(E(\alpha_1), \ldots, E(\alpha_n))$. Security under this type of composition is, intuitively, a *sine qua non* of encryption. A simple "hybrid" argument shows that *any* non-malleable public key cryptosystem is secure in this sense: seeing the encryptions of multiple α's does not help the adversary to generate an encryption of even one related β.

2) The computational difficulty of generating a single $E(\beta)$ for a related β does not imply the computational difficulty of generating a vector $(E(\beta_1), \ldots, E(\beta_n))$ such that

$R(\alpha, \beta_1, \ldots, \beta_n)$ holds. We next describe a counter-example in the case of a chosen ciphertext pre-processing attack. Let E' be a non-malleable cryptosystem under chosen ciphertext pre-processing attack. Let $E(m)$ be constructed as $(E'_0(m_0), E'_1(m_1))$, where $m = m_0 \oplus m_1$. Given a ciphertext of this form, the adversary can construct two ciphertexts: $(E'_0(m_0), E'_1(0))$ and $(E'_0(0), E'_1(m_1))$. The parity of the two decrypted values is: $(m_0 \oplus 0) \oplus (0 \oplus m_1) = m_0 \oplus m_1 = m$. On the other hand, it can be shown from the non-malleability of the E'_i that seeing $E(m)$ is of no assistance in generating *a single* encryption $E(m')$ such that $R(m, m')$.

3.4.2 Security Taxonomy and Comparisons

We have discussed two notions of breaking a cryptosystem, semantic security and non-malleability, and three types of attacks:

- Chosen plaintext.
- Chosen ciphertext attack in the pre-processing mode (CCA-pre).
- Chosen ciphertext attack in the post-processing mode (CCA-post).

This yields six types of security and the question is whether they are all distinct and which implications exist. Two immediate implications are (i) non-malleable security implies semantic security under the same type of attack and (ii) security against chosen ciphertext post-processing attacks implies security against chosen ciphertext attacks in the preprocessing mode which in turn implies security against chosen plaintext attacks, using the same notion of breaking the cryptosystem. We now explore other possibilities - the discussion is summarized in summarized in Figure 1.

The first observation is that if a cryptosystem is *semantically secure* against chosen ciphertext post-processing attacks, then it is also *non-malleable* against chosen ciphertext post-processing attacks, since the power of the adversary allows it to decrypt whatever ciphertext it generated. On the other hand, it is not difficult to start with a cryptosystem that is secure against chosen ciphertext attack in the preprocessing mode and make it only secure against a chosen plaintext attack (under any notion of breaking), as we now explain. For the case of semantic security, simply add to the decryption mechanism the instruction that on input all 0's outputs the private-key. The case of non-malleable security is more subtle. Choose a fixed random ciphertext c_0, and instruct the decryption mechanism to output the decryption key when presented with input c_0. In addition, instruct the decryption mechanism to output c_0 on input all 0's.

There is a simple method for "removing" non-malleability without hurting semantic security: starting with a cryptosystem that is non-malleable against chosen ciphertext pre-processing attacks, one can construct a cryptosystem that is only semantically secure against chosen ciphertext pre-processing attacks - add to each ciphertext a cleartext bit whose value is Xor-ed with the first bit of the plaintext. Thus, given a ciphertext of a message m it is easy to create a ciphertext of a message where the last bit is flipped, so the scheme is malleable. However, the semantic security remains, as long as the adversary does not have access to the challenge ciphertext while it can access the decryption box.

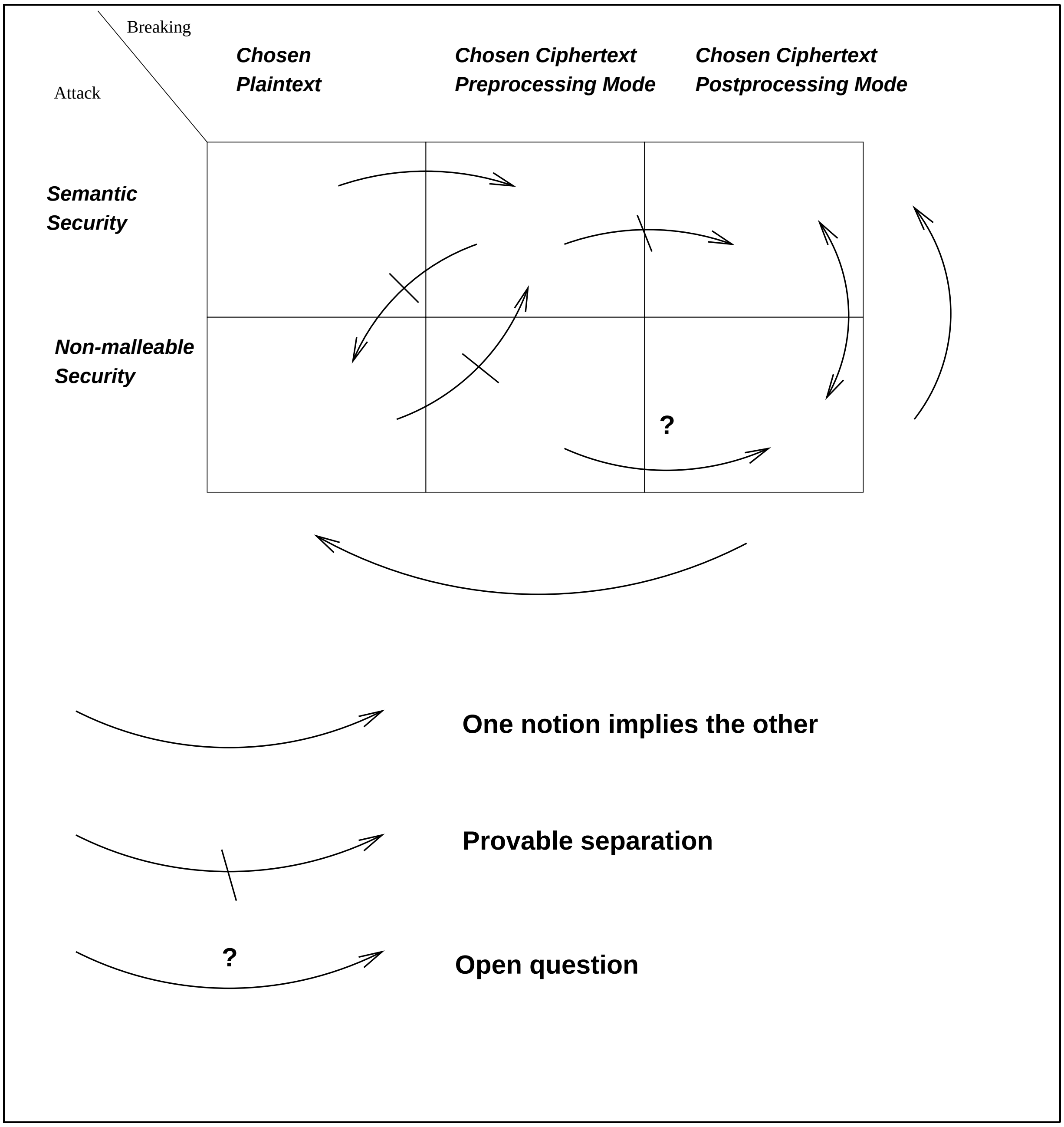

Figure 1: Relationship between security notions.

We do not know whether a scheme that is non-malleable against chosen ciphertext pre-processingis also non-malleable against chosen ciphertext post-processing attack. We conjecture that whenever deciding whether or not a string represents a legitimate ciphertext (that could have been generated by any user) is easy (to someone *not* holding the private key), non-malleability implies semantic security against a chosen ciphertext post-processing attack. From the above discussion (summarized in Figure 1), we conclude that of the six possibilities for security of a cryptosystem (combinations of the type of attack and notion of breaking) we have that either four or five are distinct[8].

Note that the type of combination to be used depends on the application. For instance, for the bidding example given in the introduction, if the public-key is *not* going to be used for bidding on more than a single contract, and assuming the bids are not secret after the bids are opened, then the type of security needed is *non-malleability against chosen plaintext attacks.* If the same public key is to be used for bidding on several contracts successively, but the secrecy of non-winning bids need not be preserved, then non-malleability under chosen ciphertext in the pre-processing mode is required. On the other hand, if the same public key is to be used for bidding on several contracts, and the secrecy of non-winning bids must be preserved, one should use a non-malleable cryptosystem secure against chosen ciphertext attacks in the post-processing mode.

Finally one may wonder what is the "correct" description of the notion of breaking a cryptosystem secure against chosen ciphertext post-processing attacks: semantic security or non-malleable security, given their equivalence under this attack. We think it is more helpful to think in terms of non-malleability, since the way to think about trying to break a candidate system is to think of trying to maul the target ciphertext(s). This was done (without the vocabulary of non-malleability) in the recent work Bleichenbacher [11] (see Section 6).

3.4.3 On Allowing $\mathcal{A}'$ to Choose $\mathcal{M}'$

Having $\mathcal{A}'$ choose $\mathcal{M}'$, rather than inheriting $\mathcal{M}' = \mathcal{M}$ from $\mathcal{A}$, makes the adversary simulator *weaker*: the real adversary $\mathcal{A}$ is allowed to mount a chosen-ciphertext attack before choosing its target distribution $\mathcal{M}$, while the adversary simulator $\mathcal{A}'$ must choose $\mathcal{M}'$ without the benefit of such an attack. Since the adversary simulator is weaker, $\forall \mathcal{A} \exists \mathcal{A}' \ldots$ becomes a stronger requirement on the cryptosystem. Our cryptosystem satisfies this strong requirement.

3.5 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 "real" 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

[8] For a very recent discussion of the relationship between these notions see Bellare et al. [3], where they show that there are indeed five distinct possibilities.

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 [45], 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 $\mathcal{P}=(A,B)$ for A to Authenticate Message m to B:

A's public key is E, chosen according to $\mathcal{S}$, a non-malleable public key cryptosystem secure against chosen ciphertext attacks in the postprocessing mode (*e.g.*, the one from Section 3.2).

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" $\gamma\in_R E(m\circ r)$ 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 3.6 *Given an adversary $\mathcal{B}$ that can break the authentication protocol $\mathcal{P}$ with probability ρ, one can construct an adversary $\mathcal{A}$ for breaking the (presumed non-malleable) encryption scheme E 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 $\langle (A,B),(C,D),\ \mathcal{B}:\psi(B)\leftrightarrow\psi(C)\rangle$, where $(A,B)=(C,D)=\mathcal{P}$. Note that since this is a simulation, $\mathcal{A}$ can control the messages sent by $\psi(D)$ in the simulation. Run the system $\langle (A,B),(C,D),\ \mathcal{B}:\psi(B)\leftrightarrow\psi(C)\rangle$ until $\psi(C)$, under control of $\mathcal{B}$, 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; whence 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 the query γ in the simulation. Let r' be $\psi(C)$'s reply. $\mathcal{A}$ outputs $\theta\in_R E(m\circ r')$.

The distribution that $\mathcal{B}$ sees in the simulation of the adversarially coordinated system $\langle (A,B),(C,D),\ \mathcal{B}:\psi(B)\leftrightarrow\psi(C)\rangle$ is exactly as usual. Therefore by assumption 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 without access to $E(m\circ r)$ is 2^{-n}. $\square$

This solution will be of practical use as soon as the current constructions of non-malleable cryptosystems are improved to be more practical. The very recent construction of Cramer and Shoup (see Section 6) makes this scheme very attractive.

Remark 3.7 *If the cryptosystem $\mathcal{S}$ is malleable, and in particular if given an encryption of a message $\lambda \circ r$ it is easy (possibly after mounting a CCA-post or other type of attack) to generate an encryption of a message $\lambda' \circ r$, where $\lambda' \neq \lambda$ (many cryptosystems have this property), then there is a simple attack on the protocol proposed: as before $\psi(C)$ is pretending to be $\psi(A)$. To forge an authentication of a message m, when D sends challenge $\gamma = m \circ r$, $\psi(B)$ asks A to authenticate a message m' by sending the challenge $\gamma' = m' \circ r$. When A replies with r, $\psi(C)$ sends r to D, who will accept.*

Remark 3.8 *As mentioned above, this protocol provides a weaker form of authentication than digital signatures (no third party verification). However, this can be viewed as a feature: there may be situations in which a user does not wish to leave a 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 (although it is almost surely not black-box zero knowledge [38]). By adding a (malleable) proof of knowledge to the string r this can be ensured in the sequential case. We do not know if the resulting zero-knowledge authentication protocol remains zero-knowledge if many executions, with the same authenticator, execute* concurrently. *The straightforward simulation fails. (See [51] for impossibility results for 4-round black-box concurrent zero-knowledge protocols.) Very recently, an approach for achieving deniable authentication in the concurrent setting based on* timing constraints *was suggested by Dwork, Naor and Sahai, who also present several efficient protocols in the standard model (no timing) for the sequential case.*

3.6 Non-Malleable Encryption in Other Settings

In this section we briefly mention non-malleable encryption in two additional settings: private key cryptography and interactive public key cryptography. In both cases we begin with a known semantically secure system and add authentication to achieve non-malleability.

Private-key Encryption

As mentioned in the beginning of Section 3, the definition of non-malleable security is applicable for private (or shared) key cryptography as well. For example, in their celebrated paper on a logic of authentication [16], Burrows, Abadi, and Needham give the following analysis of a scenario (the Needham-Schroeder authentication protocol) in which A and B share a key K_{AB}. Party B chooses a nonce N_b, and sends an encryption of N_b under K_{AB} to A. A then responds with an encryption of $N_b - 1$ under K_{AB} in order for B

> "... to be assured that A is present currently ... Almost any function of N_b would do as long as B can distinguish his message from A's – thus, subtraction is used to indicate that the message is from A, rather than from B."

The unproved and unstated assumption here is that K_{AB} provides non-malleable encryption; malleability completely destroys their reasoning and their proof of security, *even if there adversary's access to the system is very limited* (i.e. an attack weaker than chosen ciphertext in the pre-processing mode).

Achieving non-malleability in the private-key setting is much simpler and more efficient than in the public-key setting. Let K_{AB} be a private key shared by A and B. We first describe a system that is semantically secure against a chosen ciphertext attack in the pre-processing mode: Treat K_{AB} as (K_1, K_2) which will be used as seeds to a pseudo-random function f (see [37] for definition of pseudo-random functions, [56, 57] for recent constructions and [58] for a recent discussion on using pseudo-random functions for encryption and authentication). In order to encrypt messages which are n bits long we need a pseudo-random function $f_K : \{0,1\}^\ell \mapsto \{0,1\}^n$, i.e. it maps inputs of length ℓ to outputs of length n where ℓ should be large enough so as to prevent "birthdays", i.e. collision of randomly chosen elements. For A to send B a message m, A chooses a random string $r \in \{0,1\}^\ell$ and sends the pair $(r, m \oplus f_{K_1}(r))$. Semantic security of the system against chosen ciphertext attack in the pre-processing mode follows from the fact that the pseudo-random function is secure against *adaptive* attacks. However, this scheme is *malleable* and not secure against a chosen ciphertext attack in the post-processing mode: given a ciphertext (r, c) one can create a ciphertext (r, c') where c' is obtained from c by flipping the last bit. This implies that the corresponding plaintext also has its last bit flipped. In order to thwart such an attack we employ another pseudo-random function $g_k : \{0,1\}^{n+\ell} \mapsto \{0,1\}^\ell$ and add a third component to the message:

$$g_{K_2}(r \circ (m \oplus f_{K_1}(r))).$$

When decrypting a message (r, c, a) one should first verify that the third component, a, is indeed proper, i.e. $a = g_{K_2}(r \circ c)$. This acts as an *authentication tag* for the original encryption and prevents an adversary from creating any other legitimate ciphertext, except the ones he was given explicitly. (Recall that by definition of pseudo-random function, seeing any number of pairs $(r, f_{K_2}(r))$ does not yield any information about $(r', f_{K_{AB}}(r'))$ for any new r' and in particular they are unpredictable.)

Since it is known that the existence of one-way functions implies the existence of pseudo-random functions [37, 48] we have

Theorem 3.9 *If one-way functions exist, then there are non-malleable private-key encryption schemes secure against chosen ciphertext attacks in the post-processing mode.*

Since it is known that in order to have private key cryptography we must have one-way functions [47] we conclude:

Corollary 3.10 *If any kind of private-key encryption is possible, then non-malleable private-key encryption secure against chosen ciphertext attacks in the post-processing mode is possible.*

Note that the property of "self-validation" enjoyed by the above construction is stronger than needed for non-malleability, *i.e.* there are non-malleable cryptosystems that do not have this property: one can start with a non-malleable private-key cryptosystem and add to it the possibility of encryption using a pseudo-random permutation; this possibility is never (or rarely) used by the legitimate encrypter, but may be used by the adversary. The resulting cryptosystem is still non-malleable but not self-validating, since the adversary can create ciphertexts of random messages.

For a recent application of the above construction to the security of remotely-keyed encryption see Blaze et al [10].

Interactive Encryption

The second setting resembles the one studied by Goldwasser, Micali, and Tong [46], in which they constructed an *interactive* public key cryptosystem secure against chosen ciphertext attack (see also [34, 67]). An "interactive public key cryptosystem" requires a public file storing information for each message recipient, but this information alone is not sufficient for encrypting messages. The additional information needed is chosen interactively by the sender and receiver. To the best of our knowledge, their paper was the first to try to cope with an oracle for distinguishing valid from invalid ciphertexts in any setting (interactive or not). An interactive system is clearly less desirable than what has now come to be called "public key cryptography," in which the public key *is* sufficient for sending an encrypted message, without other rounds of interaction.

The definitions of non-malleable security can be easily adapted to this case, but when discussing the attack there is more freedom for the adversary, due to the interactive nature of the communication. In general, we assume that the adversary has complete control over the communication lines and can intercept and insert any message it wishes. A precise definition is outside the scope of this paper.

Our non-malleable interactive public key cryptosystem requires a digital signature scheme that is *existentially unforgeable against a chosen message attack* (see the Introduction for an informal definition of existential unforgeability). Let (S_i, P_i) denote the private/public signature keys of player i (the model assumes that there is a public directory containing P_i for each player i that is to *receive* messages, but the sender is *not* required to have a key in the public directory). The system will also use a public-key cryptosystem semantically secure against chosen plaintext attacks.

The idea for the system is straightforward: for each interaction the receiver chooses a fresh public-key private-key pair that is used only for one message. However, this is not sufficient, since an active adversary may intercept the keys and substitute its own keys. We prevent this behavior by using signatures. A sender j wishing to send a message m to receiver i performs the following:

1. Sender j chooses a *fresh* private/public pair of signature keys (s_j, p_j) and sends the public part, p_j, to i (lower case is used to distinguish p_j from what is in the directory);

2. Receiver i chooses a *fresh* private/public pair of *encryption and decryption* keys (E_{ij}, D_{ij}), where E_{ij} is *semantically secure against chosen plaintext attack*, and sends E_{ij} together with $S_i(E_{ij} \circ p_j)$ (*i.e.* a signature on the fresh public-key E_{ij} concatenated with the public signature key j chose) to j; j verifies the signature and that p_j is indeed the public key it sent in Step 1.

3. Sender j encrypts m using E_{ij} and sends $E_{ij}(m)$ together with $s_j(E_{ij}(m))$ to i. Receiver i verifies that the message encrypted with E_{ij} is indeed signed with the corresponding p_j.

Note that the sender may use a one-time signature scheme for (s_j, p_j) and if the receiver uses a signature scheme such as in [27, 22], then the approach is relatively efficient.

4 A Non-Malleable Scheme for String Commitment

We present below a scheme $\mathcal{S}$ for string commitment that is non-malleable with respect to itself (Definition 2.6). 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 $\mathcal{S}$: knowledge extraction for $\mathcal{S}'$ implies non-malleability of $\mathcal{S}$. Thus, in this section, the new $\mathcal{S}'$ 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 [55] (although any computational scheme will do). 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 [55], we consider only the second stage of that protocol.

We also require *zero-knowledge proofs* satisfying the security requirements in [35]. These can be constructed from any bit commitment protocol [40].

Before we continue it would be instructive to consider the protocol of Chor and Rabin [21]. 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 ensures 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 Non-Malleable String Commitment Scheme $\mathcal{S}$

Protocol $\mathcal{S}$ consists of two general stages. The first is a string commitment as in [55]. The second stage, 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 used repeatedly during each execution of **BCK**, which we now describe. In the following, n is a security parameter.

Protocol BCK(α) (assumes the committer has already committed to α):

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

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

performed at once.

- **BCK1** (Commit): Committer selects random $x_0, x_1 \in \{0,1\}^k$, where $k = |\,\alpha\,|$, and commits to both of them using the protocol in [55].
- **BCK2** (Challenge): Receiver sends Committer a random bit $r \in \{0,1\}$.
- **BCK3** (Response): 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 $\mathbf{BCK}1$. The proof of consistency with the initial commitment is done for all n instances together as a single statement.

The interactive proof in $\mathbf{BCK}3$ is a proof of consistency; it need not be proof of knowledge in the sense of [31].

Remark 4.1 *From $\alpha \oplus x_{1-r}$, x_{1-r}, and the proof of consistency, one can obtain α. This is why we call the protocol* Basic Commit with *Knowledge* (of α).

Note also that the interactive proof is of consistency; it is not a proof of knowledge in the sense of [31].

In the rest of the section we consider each $\mathbf{BCK}i$ as single operation, thus it can be viewed as an operation on an n-dimensional vector or array. Note that $\mathbf{BCK}1$ and $\mathbf{BCK}2$ are indeed "instantaneous," in that each requires a single send, while $\mathbf{BCK}3$, due to its interactive nature, requires more time to carry out. We frequently refer to an instance of **BCK** as a *triple.*

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 2 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 for which no other triple is useful. This is analogous to Chor and Rabin ensuring that for every pair of participants A, C there exists a round in which C is proving knowledge while A is not. We say such a triple is *exposed* (defined precisely below). This is the key idea in the construction.

The next two *sixplet* protocols perform a pair of distinct instances of $\mathbf{BCK}(\alpha)$ in two different interleaved orders. To distinguish between the two instances of **BCK** 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 **BCK**. These Sixplet protocols will be used to ensure the existence of an exposed triple in the Basic Commit with Knowledge. The intention of the spacing of the presentation is 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 is exactly the same fact that was exploited by Chor and Rabin in the synchronous "everyone-knows-everyone" model to enforce the condition that for every pair of provers $A \neq C$, there is a round in which C is proving but A is not [21]. The same fact is used in both cases for the same purpose, but we do it without any assumption of synchrony and without any assumption that each processor knows of all other processors in the system.

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

- Commit to α (*e.g.*, using the protocol in [55]).
- 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.

4.2 Properties of $\mathcal{S}$

We now show some properties of $\mathcal{S}$ that allow us to prove its non-malleability. Suppose that $(A, B) = (C, D) = \mathcal{S}$, and suppose further that adversary $\mathcal{A}$ controls $\psi(B)$ and $\psi(C)$. 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 copying.) Note also that, given the proofs of consistency, both sender and receiver know at the end of the commitment protocol whether or not the sender has succeeded in committing to a well-defined value. Thus, the event of *successful commitment to some value* by $\psi(C)$ is independent of the value committed to by $\psi(A)$.

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**2 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 do not assume that these values are known to the parties involved – there is no "common clock."

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) $\delta_3^j > \tau_2^i$ (see Figure 2).

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 $j_1 < j_2$. □

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 $\mid \Gamma^{(2k-1)} \mid \leq 1$, and $\mid \Gamma^{(2k)} \mid \leq 1$. 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. □

Notice that the reverse claim does not hold.

Lemma 4.2 *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. □

4.3 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 $\psi(C)$ succeeds in committing to a string β, then 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 the $\mathcal{S}'$-adaptor-$\mathcal{S}$ system 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. (As we will see, $m = |I| \frac{16}{\epsilon^2 \log \epsilon^{-1}}$, where ϵ is how close we would like the extraction probability to be to the probability of successful completion of the protocol by $\mathcal{A}$.)

Protocol $\mathcal{S}'$: P Commits to a string α:

- Commit to α (*e.g.*, using the protocol in [55]).
- Repeat m times:

1. Q chooses a bit b and requests a b-sixplet; according to additional inputs, Q requests that the b-sixplet be augmented by an *additional* proof of consistency in step **BCK**3 of either triple in the b-sixplet;
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 simulation argument yields the following lemma:

Lemma 4.3 *For any strategy of choosing the sixplets and for any receiver Q', the string commitment protocol $\mathcal{S}'$ is semantically secure.* □

We provide an adaptor 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 adaptor.

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 adaptor that operates between P and B: whenever (A, B) calls for a b-sixplet the adaptor "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 adaptor 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 $\rho - \epsilon$[9].

Fix $\epsilon > 0$. The knowledge extractor begins to run $\mathcal{S}' = (P, Q)$ and $\mathcal{S} = (C, D)$ with the adaptor 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 3) in which $(A, B) = (C, D) = \mathcal{S}$.

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 messages sent by 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. In order to exploit Remark 4.1 we will be interested in D-nodes corresponding to a **BCK**2 step. The branches correspond to the different possible challenge vectors that D can send in this step. In the sequel, these are the only types of D-node that we will consider.

[9] The extraction procedure runs in a fixed polynomial time (in n and ϵ^{-1}) and succeeds only with probability p. This leads to an ϵ-malleable, which we suspect is sufficient "for all practical purposes." A modification of the procedure runs in *expected* polynomial time, and succeeds with probability ρ (the best possible), yielding liberal non-malleability. See Remark 4.4.

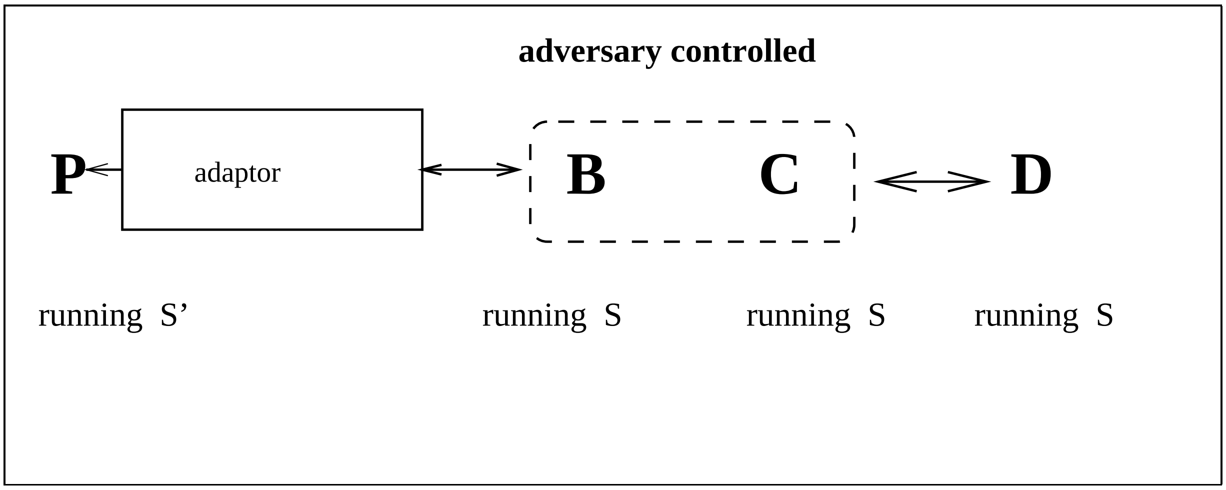

Figure 3: The $\mathcal{S}'$-adaptor-$\mathcal{S}$ system used in constructing the Extractor

To enable us to follow more than a single path (that is, to *fork*) in the tree, we keep at each such 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 in Remark 4.1, given two such nodes we can extract β.

To identify such a pair of nodes, choose $\ell = \frac{16}{\epsilon^2 \log \epsilon^{-1}}$, and run the following extraction procedure ℓ times, each time starting again at the root of tree. (Recall that the root of the tree corresponds to the point at which the tape was frozen; we do not re-start (C, D) each time the extraction procedure is repeated.)

By Lemma 4.2 every path to a good leaf contains an exposed triple. Run the $\mathcal{S}'$-adaptor-$\mathcal{S}$ 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 according to the interleavings (the interleavings are shown pictorially after the types are formally defined):

- 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 $\tau_2^i \geq \delta_2^j$ 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:

$$
\begin{array}{ccccc}
\tau_1^i & & & \tau_1^i & \\
 & \delta_1^j & \text{or} & \tau_2^i & \\
\tau_2^i & & & & \delta_1^j \\
 & \delta_2^j & & & \delta_2^j
\end{array}
$$

Thus, in the first type of exposure, 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 ended, thus A doesn't play a part in $\mathcal{S}$ 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 challenges in $\mathbf{BCK}2$ of triple j. Note that with overwhelming probability any two such challenge vectors differ in some position. Since at the point t defined above the challenges sent to A in $\mathbf{BCK}2$ of triple i are already fixed, the two copies of $\mathbf{BCK}3$ of triple i will differ only in the proofs of consistency. The adaptor multiplexes to P the two proofs of consistency. This completes the treatment of the first type of exposed triple.

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 open triple is the ith triple; if there are two open triples then they are the ith and $(i+1)$st 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 $\delta_2^j < \tau_2^{i+1} < \delta_3^j$). We show the interleavings and mark the forking points with asterisks:

$$
\begin{array}{ccccc}
\textbf{Case (a)} & & & \textbf{Case (b)} & \\
\tau_1^i & & & \tau_1^i & \\
\tau_1^{i+1} & & & \tau_1^{i+1} & \\
 & \delta_1^j & & & \delta_1^j \\
\tau_2^{i+1} & & & & * \\
* & & \text{or} & & \delta_2^j \\
\tau_3^{i+1} & & & & \delta_3^j \\
 & \delta_2^j & & \tau_2^{i+1} & \\
 & \delta_2^j & & \tau_3^{i+1} & \\
\tau_2^i & & & \tau_2^i &
\end{array}
$$

In Case (a) we fork right after τ_2^{i+1}, running a copy of $\mathcal{A}$ until the conclusion of triple j in the copy. Although this means there will be two copies of $\mathbf{BCK}3(i+1)$, they will differ only in their interactive proofs of consistency: the challenges are fixed by time τ_2^{i+1}. (Note that we can assume that $\tau_3^{i+1} < \delta_2^j$ because the replies to the challenges and the statements to be proved by P in **BCK**3$(i+1)$ are completely determined by **BCK**1 and **BCK**2, and are therefore are completely determined by time τ_2^{i+1}. Moreover, the challenges sent in **BCK**2(j) by D are independent of **BCK**2$(i+1)$ because **BCK**2$(i+1)$ ends before **BCK**2(j) starts and D is non-faulty.) D makes independent challenges in the two runs. We will not run the original beyond δ_3^j. The communication with A is limited in the original execution to the replies to the challenges sent in **BCK**2$(i+1)$ and the zero knowledge proof of consistency in **BCK**3$(i+1)$. However, since the challenges in the two copies are the same, and since in $\mathcal{S}'$ the committer P is willing to repeat this proof, when running the copy we simply ask for a repeated proof of consistency and continue as before. We stop when the copy finishes **BCK**3 of the jth triple. Note that in the copy the jth triple need not be exposed (this depends on τ_2^i).

Case (b) is 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 **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 **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.

Remark 4.4 *To remove the dependency on ϵ, at the expense of running in expected polynomial time, i.e., to obtain liberal non-malleability (see Remark 2.3, pursue the following strategy. Choose a random path to a leaf in the tree defined above. If the leaf is not good, then abort. Otherwise "extract at all costs." That is, repeat until done:*

1. *Choose a random path to a leaf. If this leaf is good, then add to the set of previously chosen paths.*

2. *For all previously chosen paths (necessarily ending with a good leaf), attempt once (again) to extract as described above.*

We now show that the extractor succeeds with high probability. 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 4.5 *In each run of the above experiment the value of 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. $\mathcal{V}$ consists of the "first" nodes such that $\rho(v) < \rho_0/2$ and hence no member of $\mathcal{V}$ is an ancestor of another). 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 at least $\rho_0/2$. 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 $\rho_0^2/4 - 1/2^{2n}$. □

To obtain an analogous result for the liberal non-malleability extraction procedure outlined in Remark 4.4, consider a random path that yields a good leaf. By Claim 4.4, $\Pr[\rho(w) > \rho/2] \geq 1/2$, that is, with probability at least 1/2 a good leaf is also a good investment for extraction. Thus the "extract at all costs" procedure runs in expected time $O(1/\rho_0)$ and the probability this extraction is invoked is ρ_0, yielding expected polynomial time (taking the usual precautions against running forever).

Continuing with the proof of ϵ-malleability, 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 4.6 *For any adversarially coordinated system $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$ in which $(A,B) = (C,D) = \mathcal{S}$, there is a knowledge extraction procedure that succeeds in extracting from the $\mathcal{S}'$-adaptor-$\mathcal{S}$ system the value committed to by $\psi(C)$ with probability arbitrarily close to ρ.* □

We can therefore conclude that, in essence, the values β obtained by the extractor are "correctly" distributed. We would like to say that when β is obtained by the extractor,then for every relation approximator R, the probability that $R(\alpha, \beta)$ outputs 1, is subpolynomially close to $\pi(\mathcal{A}, R)$ (the probability that it holds under a "normal" execution). However, in the true execution $\psi(C)$ might make moves that force R to reject (for instance when the real player makes an illegal move or refuses to open with certain probability). This doesn't necessarily imply that the extraction would fail. However, such cases only help us and we can conclude:

Corollary 4.7 *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 can be made arbitrarily close.*

4.4 Extraction Implies Non-Malleability

In this section we reduce the non-malleability of $\mathcal{S}$ to the semantic 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 $\mathcal{S}$.

Recall that $R(x,x) = 0$ for all relation approximators. We view the goal of $\mathcal{A}$ (respectively, $\mathcal{A}'$) as trying to maximize $\pi(\mathcal{A}, R)$ (respectively, $\pi'(\mathcal{A}', R)$). Consider the following procedure for an adversary simulator $\mathcal{A}'$ with access to the probability distribution $\mathcal{D}$ chosen by $\mathcal{A}$, on inputs to $\psi(A)$.

Procedure for $\mathcal{A}'$ on input hist(α):

1. Set $\mathcal{D}' = \mathcal{D}$.
2. Generate $\delta \in_R \mathcal{D}' = \mathcal{D}$.
3. 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 $\mathrm{hist}(\alpha)$, and if $\psi(C)$ succeeds in committing to a value γ, extract γ.
4. Output γ (that is, give γ as input to $\psi(C)$).

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, if any, 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 $\mathrm{hist}(\alpha)$. By definition, for any probabilistic polynomial time relation approximator R, $\Pr[R(\alpha, \beta)] = \pi(\mathcal{A}, R)$.
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 and $\mathcal{A}'$ has input $\mathrm{hist}(\alpha)$. Let $\tilde{\pi}(\mathcal{A}, R) = \Pr[R(\alpha, \beta')]$ Intuitively, this is the probability that $\psi(C)$ commits to something related to α in an $\mathcal{S}'$-adaptor-$\mathcal{S}$ system in which all parties have the "right" inputs, and this value is successfully extracted.
3. Let β'' be obtained by extraction from $\mathcal{A}'$ in a run of the $\mathcal{S}'$-adaptor-S system in which $\psi(P)$ has input $\delta \in_R \mathcal{D}$ but $\mathcal{A}'$ has input $\mathrm{hist}(\alpha)$. Then $\pi'(\mathcal{A}', R) = \Pr[R(\alpha, \beta'')]$ since this is exactly the setting of the variables when $\mathcal{A}'$ receives as input $\mathrm{hist}(\alpha)$ (see the definition of $\mathcal{A}'$ above).

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 Corollary 4.7, 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 $\pi(\mathcal{A}, R)$, the probability that $\mathcal{A}$ succeeds, is essentially less than or equal to $\tilde{\pi}(\mathcal{A}, R)$, which we show in the first step of the proof to be close to $\pi'(\mathcal{A}', R)$. From this we conclude the non-malleability of $\mathcal{S}$.

Lemma 4.8 *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 committed values.*

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 hist(α_1), and extract ζ, the value committed to by $\psi(C)$ in the emulation.

3. Output $R(\alpha_1, \zeta)$.

If, in the emulation, $i = 1$, then the input to $\psi(P)$ is α_1 and so the distinguisher outputs 1 with probability $\tilde{\pi}(\mathcal{A}, R)$. Similarly, if in the emulation $i = 2$, then the input to $\psi(P)$ is α_2, and so the distinguisher outputs 1 with probability $\pi'(\mathcal{A}', R)$. Since by assumption these two quantities differ polynomially, we have a polynomial distinguisher for commitments in $\mathcal{S}'$. □

Corollary 4.9 $|\tilde{\pi}(\mathcal{A}, R) - \pi'(\mathcal{A}', R)|$ *is subpolynomial.* □

Theorem 4.10 *The string commitment scheme $\mathcal{S}$ is: (1) ε-malleable and (2) liberal non-malleable.*

Proof. (1) By Corollary 4.7, $\pi(\mathcal{A}, R) < \tilde{\pi}(\mathcal{A}, R)$ or the two can be made arbitrarily close. Thus $\mathcal{A}$ is at most ε 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 4.8, $\pi'(\mathcal{A}', R)$ 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 $\psi(P)$.

For (2), note that the expected polynomial time extraction procedure described in Remark 4.4 succeeds with probability ρ, so the ε difference disappears. □

Remark 4.11 *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). Let $\mathcal{F}$ (for fewer) denote this string commitment protocol.*

To see why $\mathcal{F}$ 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'.

Using this approach, the result of Chor and Rabin [21] can be improved to require $\log \log n$ *rounds of communication, (down from* $\log n$ *rounds). Recall that their model differs from ours in that they assume all n parties are aware of each other and that the system is completely synchronous.*

Remark 4.12 *1) 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. As in Section 3.4.1, this is shown by a hybrid argument.*

2) There is a lack of symmetry between our definitions of non-malleable encryption and non-malleable string commitment: the first requires that it should be computationally difficult, given $E(\alpha)$, to generate a vector *of encryptions $(E(\beta_1), \ldots E(\beta_n))$ such that $R(\alpha, \beta_1, \ldots, \beta_n)$ holds, while the second requires only that access to a commitment to a string α should not help in committing to a* single *related string β. It is possible to modify the definition to yield this stronger property. Roughly speaking, we add a fictitious step after the adversary attempts to commit to its values, in which the adversary specifies which successfully committed values will be the inputs to the relation approximator R. The extraction procedure is then modified by first running $\mathcal{S}'$ with a simulation of $\mathcal{A}$ to see which commitments succeed. Then we argue that with high probability the extraction procedure succeeds on all of these. This follows from the high probability of success during any single extraction (Lemma 4.6). We chose not to use the extended definition because it would complicate the proofs even beyond their current high level of complexity.*

3) The weaker definition does not imply the stronger one: the protocol $\mathcal{F}$ of Remark 4.11 is a counterexample. Let $(A, B) = \mathcal{F}$ and let $\psi(A)$, running $\mathcal{F}$, commit to α by splitting it into $\alpha_1, \ldots, \alpha_n$. Let $(C_1, D_1) = \ldots = (C_n, D_n) = \mathcal{F}$. If the $n+1$ parties $\psi(C_1), \ldots \psi(C_n)$ have identities such that for each i the ith bit of the identity of $\psi(C_i)$ equals the ith bit of the identity of $\psi(A)$, then the parties $\psi(B), \psi(C_1), \ldots \psi(C_n)$ can collude as follows. Each $\psi(C_i)$ commits to the string $\beta_i = \alpha_i$ by splitting it into $\beta_i = \beta_{i1} \oplus \ldots \oplus \beta_{in}$, where $\beta_{ii} = \alpha_i$ and $\beta_{ij} = 0^{|\alpha_i|}$. In this way the colluding parties can arrange to commit to $\beta_1, \ldots, \beta_n$ such that the exclusive-or of the β's equals α.

This counterexample also illustrates why the technique for reducing rounds described in Remark 4.11 cannot be iterated to obtain a constant round protocol.

5 Zero-Knowledge Proofs and General Non-Malleable Zero-Knowledge Interactions

For the results in this section we assume the players have unique identities. Let $(A, B)[a, b]$ be a zero-knowledge interactive protocol with valid set Π of input pairs. Recall from Section 2.1 that (A, B) is *zero-knowledge* with respect to B if for every $\psi(B)$ under control of a polynomial-time bounded adversary $\mathcal{A}$, there exists a simulator Sim such that the following two ensembles of conversations are indistinguishable. In the first ensemble, $\mathcal{A}$ chooses a distribution $\mathcal{D}$ consistent with Π, 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, and adversary simulator $\mathcal{A}'$ with the same computational power as $\mathcal{A}$ chooses a distribution $\mathcal{D}'$ consistent with Π, $(\alpha, \beta) \in_R \mathcal{D}'$ is selected, $\mathcal{A}'$ is given β, and produces a simulated conversation.

We construct a compiler $\mathcal{C}$, which, given *any* zero-knowledge interaction (A, B) produces a zero-knowledge protocol which is non-malleable in the sense described next.

Let (A', B') be any zero-knowledge protocol and let $(A, B) = \mathcal{C}(A', B')$. Let (C', D') be any (not necessarily zero knowledge) protocol, and let $(C, D) = \mathcal{C}(C', D')$. Consider the adversarially coordinated system $\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$. 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 a distribution for inputs to all four players chosen by $\mathcal{A}$ consistent with the valid inputs for (A, B). Let $(\alpha, \beta, \gamma, \delta) \in_R \mathcal{D}$, and run the compiled versions of the two protocols. Let $\pi(\mathcal{A}, R)$ denote the probability that $R(\alpha, \beta, \gamma, \delta, \mathcal{D}, \mathcal{K}(C, D)) = 1$, where $\mathcal{K}(C, D)$ denotes the conversation between $\psi(C)$ and $\psi(D)$. The probability is over the coin-flips of $\mathcal{A}, \psi(A)$ and $\psi(D)$ and the choice of $(\alpha, \beta, \gamma, \delta)$ in $\mathcal{D}$. As above, R rejects if a conversation is syntactically incorrect.

Let $\mathcal{D}'$ (consistent with the legal input pairs for (A, B)) be chosen by $\mathcal{A}'$, and 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 $\mathcal{K}'(C, D)$ denote the resulting conversation. Let $\pi(\mathcal{A}', R)$ denote the probability that $R(\alpha, \beta, \gamma, \delta, \mathcal{D}', \mathcal{K}'(C, D)) = 1$. The probability is over the coin-flips of $\mathcal{A}$ and $\psi(D)$ and the choice of $(\alpha, \beta, \gamma, \delta)$ in $\mathcal{D}'$.

The *non-malleable zero-knowledge* security requirement is that for every polynomial time-bounded $\mathcal{A}$, there exists a polynomial-time bounded $\mathcal{A}'$ such that for every polynomial-time computable relation approximator R $|\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)|$ is subpolynomial.

Theorem 5.1 *There exists a compiler $\mathcal{C}$ that takes as inputs a 2-party protocol and outputs a compiled protocol. Let (A', B') be any zero-knowledge protocol and let $(A, B) = \mathcal{C}(A', B')$. Let (C', D') be any (not necessarily zero knowledge) protocol, and let $(C, D) = \mathcal{C}(C', D')$. Then the adversarially coordinated system $\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ is non-malleable zero-knowledge secure.*

Proof. Our compiler is conceptually extremely simple: A and B commit to their inputs and random tapes and then execute the protocol (A', B'), at each step proving that the messages sent are consistent with the committed values. We have to make sure that these zero-knowledge proofs of consistency do not interfere with the original protocol. The goal of the preprocessing phases is to make all the players' actions in the rest of the protocol predetermined. We now describe the action of the compiler on (A', B') in more detail.

Preprocessing Phase I: Initially A and B choose a random string R_A as follows. A non-malleably commits to a string σ_A using a sequence of non-malleable bit commitments. B then sends a random string σ_B. The string R_A, not yet known to B, is the bitwise exclusive-or of σ_A and σ_B. A and B then choose a random string R_B in the same manner, but with the roles reversed, so that B knows R_B while A does not yet know it.

Preprocessing Phase II: Each player performs a sequence of pairs of non-malleable bit commitments. Each pair contains a commitment to zero and a commitment to one, in random order.

Preprocessing Phase III: Each player commits to its input and to the seed of 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. Note in particular that A and B *both* begin with a non-malleable commitment to their inputs and random tapes – this is critical.

Executing the Original Protocol The parties 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 input and random tape, and the messages of the original protocol received so far. The commitments performed as part of the proofs of consistency are selected from the list of pairs of commitments generated in Preprocessing Step II. 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 has a (malleable) zero-knowledge proof [40] in which the verifier only sends random bits. These random bits are taken from R_A and R_B. In particular, R_A is used as the random bits when B proves something to A: A, acting as verifier and knowing R_A, reveals the bits of R_A to B as they are needed by opening the necessary commitments from Preprocessing Phase I. The analogous steps are made when A proves consistency to B.

Before sketching the proof, we give some intuition for why we included Preprocessing Phases I and II. (While it is possible that these extra preprocessing steps are not needed, we do not see a complete proof without them.) First, note that the compiler uses a specific non-malleable string commitment scheme (the one from Section 4), rather than *any* such protocol. We used this protocol because of its extraction properties (which we use for proving non-malleability). However, as we saw in Section 4 in order to do the extraction in an adversarially coordinated system $\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ in which $(A, B) = (C, D) = \mathcal{S}$, we needed to define $\mathcal{S}'$, a relaxed version of $\mathcal{S}$, and construct an $\mathcal{S}'$-adaptor-$\mathcal{S}$ system. We do not know how to construct "relaxed versions" of *arbitrary* protocols (A', B'). Since the compiled protocol (A, B) has a very special form, the construction of its relaxation is straightforward.

We now sketch the proof that the compiled protocol satisfies the requirements of the Theorem. A's proofs of consistency are zero-knowledge since they use the random bits in R_B and in the simulation of this part of the interaction R_B can be extracted. A's proofs are sound since its bit commitments performed in Preprocessing Phase II are independent of R_B (since all the commitments are non-malleable, and in particular, involve proofs of knowledge).

Since A and B commit in Preprocessing Phase III to their random tapes and values, the parts of the compiled communication that correspond to messages in (A', B') are completely determined before the execution corresponding to the (A', B') interaction is carried out.

Note that the three stage protocol described above remains zero-knowledge. This is true, since under the appropriate definition [41], the sequential composition of zero-knowledge protocols is itself zero-knowledge. So in particular, the (A, B) interaction is zero-knowledge.

Non-malleable zero-knowledge security is proved as follows. We first note that the commitment of its input and random tape that A makes to B in Preprocessing Phase III remains non-malleable despite the proofs of consistency during the execution of the original protocol. We then construct an extractor for the committed value in (C, D) in a fashion similar to the one constructed in Section 4. To do this, we construct a "relaxed" zero-knowledge protocol analogous to $\mathcal{S}'$, based on (A, B). We apply Lemma 4.6 to show that the probability of extraction is similar to the probability that $\mathcal{A}$ succeeds (in the compiled (C, D) protocol). The key point is that an exposed triple remains exposed despite the presence of the proofs of consistency because the queries in the proofs of consistency have been predetermined in Preprocessing Phase I.

As in Lemma 4.8, extraction violates the zero-knowledge nature of (the relaxed) (A, B). □

6 Concluding Remarks and Future Work

There are several interesting problems that remain to be addressed:

1. The issue of preserving the non-malleability of compiled programs (as in Section 5) under concurrent composition is challenging, as, unlike the cases of encryption and string commitment, in general zero-knowledge proofs are not known to remain zero-knowledge under concurrent composition (see, *e.g.*, [35, 38]). On the other hand, there are various techniques for changing zero-knowledge protocols so that they become parallelizable, such as witness indistinguishability [30] and perfect commitments (See Chapter 6.9 in [35]). These techniques do not necessarily yield protocols that can be executed concurrently while preserving zero-knowledge.

2. All our non-malleability results are for protocols that are in some sense zero-knowledge. Extend the definition of non-malleability to interactions that are not necessarily zero-knowledge, such as general multi-party computation, and construct non-malleable protocols for these problems.

3. Simplify the constructions in this paper. Bellare and Rogaway present simplified constructions using a random oracle [6, 7]. A challenging open problem is to (define and) construct a *publicly computable pseudo-random function*. Such a construction is essential if [6, 7] are to be made complexity-based. For a recent discussion on constructing such functions see [17, 18, 19]; note that none of the proposals there is sufficient to yield non-malleability.

Very recently Cramer and Shoup [23] suggested an efficient construction of a non-malleable cryptosystem secure against chosen ciphertext attacks in the postprocessing mode. The scheme is based on the Decisional Diffie-Hellman assumption (see [57] for a discussion of the assumption) and requires only a *few* modular exponentiations for encryption and decryption.

Recently, Di Crecsenzo *et al.* [26] showed that in a model in which there is a common random string shared by all parties, it it possible to obtain a non-interactive weaker variant of non-malleable commitments. Recall that, informally, in our definition of non-malleable commitment, the adversary succeeds if it commits to a "related" value. Our definition therefore does not require the adversary to actually *open* its commitment in order to succeed. In the [26] scheme, an adversary that commits to a related value, but never opens the commitment, is *not* to considered to have succeeded.

4. Another recent development related to malleability in encryption is the work of Bleichenbacher [11] who showed how the ability to mall ciphertexts in the PKCS # 1 standard allows for a chosen ciphertext post-processingattack. The interesting fact about this attack is that the only type of feedback the attacker requires is whether a given string represents a valid ciphertext. This demonstrates yet again the significance of using a *provable* non-malleable cryptosystem.

5. A recent result that utilizes non-malleability in an interesting way is [4] who explores the issue of reducing an adversary's success probability via parallel repetition. They give an example of a protocol where the fact that the upper bound on the adversary's probability of success is 1/2 is due to the *non-malleability* of a cryptosystem used, while the repeated protocol fails to reduce the error due to the *malleability* of the protocol itself.

6. The selective decryption problem: a type of chosen ciphertext attack *not* addressed in this paper is when the adversary is given the random bits used to generate the ciphertext (in addition to the plaintext). The following problem, phrased here in terms of a CD-ROM, is a concrete instance in which this kind of attack is relevant (the version presented here is due to [59], and is a variant of a problem posed by O. Goldreich): A CD-ROM is generated containing the encryptions of 100 images (generally, n images). A user, having a copy of the CD-ROM, chooses any subset, say of size 50, of the images, and purchases the decryption information for the selected images. Suggest an encryption scheme for this problem such that, *assuming the decryption information is significantly shorter* than the combined plaintexts of the purchased images, the remaining encryptions remain secure once the decryption information for the purchased images is known. Suppose we start with a semantically secure cryptosystem, and encrypt each image with its own key. Then, if the decryption information is the collection of keys for the selected images, it is easy to show that an adversary can't, for any given undecrypted image P_i produce an I related to P_i. The challenge is to show that no adversary can find an I related to, say, all the remaining P_i's. For example, show that the adversary can't find the bitwise logical-OR of the remaining pictures.

   This type of problem is simply ignored in papers on generating session keys (see, *e.g.*, [8, 9]). If session keys are to be used for encryption, then the selective decryption problem must be addressed.

7. Design a *completely malleable* cryptosystem in which, given $E(x)$ and $E(y)$ it is possible to compute $E(x+y)$, $E(xy)$, and $E(\bar{x})$, where $\bar{x}$ denotes the bitwise complement of x. Such a cryptosystem has application to secure 2-party computation. For example, to compute $f(x, y)$ player A generates a completely malleable E/D pair and sends $(E(x), E)$ to player B. Player B, knowing y and a circuit for f, can return $E(f(x, y))$.

   Alternatively, prove the *non-malleability conjecture*: if a cryptosystem is completely malleable then it is insecure. A related statement holds for discrete logarithms modulo p, and in general for the *black box field problem*. See the elegant papers of Maurer [53] and Boneh and Lipton [15].

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 Ran Canetti, Uri Feige, Marc Fischlin, Roger Fischlin, Oded Goldreich, Omer Reingold and Adi Shamir for valuable discussions.

References

[1] M. Ajtai and C. Dwork, *A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence* Proc. 29th Annual ACM Symposium on the Theory of Computing, 1997, pp. 284–293, and ECCC Report TR96-065.

[2] 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.

[3] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations among notions of security for public-key encryption schemes*, Advances in Cryptology - Crypto'98, Lecture Notes in Computer Science No. 1462, Springer-Verlag, 1998, pp. 26–45.

[4] M. Bellare, R. Impagliazzo, and M. Naor, *Does Parallel Repetition Lower the Error in Computationally Sound Protocols?*, Proc. of 38th Annual Symposium on Foundations of Computer Science, IEEE, 1997, pp. 374–383.

[5] M. Bellare and S. Micali, *How to Sign Given Any Trapdoor Function*, J. of the ACM **39**, 1992, pp. 214–233.

[6] M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, Proc. First ACM Conference on Computer and Communications Security, 1993, pp. 62–73.

[7] M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption – How to Encrypt with RSA*, Advances in Cryptology – Eurocrypt'94, Lecture Notes in Computer Science vol. 950, Springer-Verlag, 1994, pp. 92-111.

[8] M. Bellare and P. Rogaway, *Entity Authentication and key distribution*, Advances in Cryptology – Crypto'93, Lecture Notes in Computer Science No. 773, Springer-Verlag, 1994, pp. 232–249.

[9] M. Bellare and P. Rogaway, *Provably Secure Session Key Distribution: The Three Party Case*, Proc. 27th Annual ACM Symposium on the Theory of Computing, 1995, pp. 57–66.

[10] M. Blaze, J. Feigenbaum and M. Naor, *A Formal Treatment of Remotely Keyed Encryption*, Advances in Cryptology – Eurocrypt'98 Proceeding, Lecture Notes in Computer Science No. 1403, Springer-Verlag, 1998, pp. 251–265.

[11] D. Bleichenbacher, *Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS # 1*, Advances in Cryptology - Crypto'98 Lecture Notes in Computer Science No. 1462, Springer Verlag, 1998, pp. 1–12.

[12] M. Blum, 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.

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

[14] 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, Springer Verlag, 1985, pp. 289–299.

[15] D. Boneh and R. Lipton, *Algorithms for Black-Box fields and their application to cryptography*, Advances in Cryptology - Crypto'96, Lecture Notes in Computer Science No. 1109, Springer Verlag, 1996, pp. 283–297.

[16] M. Burrows, M. Abadi, and R. Needham, *A Logic of Authentication*, ACM Trans. on Computer Systems, *8*(1) 1990, pp. 18–36.

[17] R. Canetti, *Towards Realizing Random Oracles: Hash Functions that Hide all Partial Information*, Advances in Cryptology - Crypto 97, Lecture Notes in Computer Science vol. 1294, Springer Verlag, 1997, pp. 455–469.

[18] R. Canetti, O. Goldreich, and S. Halevi, *The Random Oracle Methodology*, Proc. 30th Annual ACM Symposium on the Theory of Computing, Dallas, 1998, pp. 209–218.

[19] R. Canetti, D. Miccianco and O. Reingold, *Perfectly One-way Probabilistic Hashing*, Proc. 30th Annual ACM Symposium on the Theory of Computing, Dallas, 1998, pp. 131–140.

[20] 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.

[21] 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.

[22] R. Cramer and I. Damgard, *New generation of secure and practical RSA-based signatures* Advances in Cryptology - Crypto '96, Lecture Notes in Computer Science 1109, Springer Verlag, 1996, pp. 173-185.

[23] R. Cramer and V. Shoup, *A practical Public Key Cryptosystem Provable Secure against Adaptive Chosen Ciphertext Attack*, Advances in Cryptology - Crypto'98 Lecture Notes in Computer Science No. 1462, Springer Verlag, 1998, pp. 13–25.

[24] 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.

[25] A. De Santis and G. Persiano, *Non-Interactive Zero-Knowledge Proof of Knowledge* Proc. of the 33th IEEE Symposium on the Foundation of Computer Science, 1992, pp. 427–436.

[26] G. Di Crescenzo, Y. Ishai and R. Ostrovsky, *Non-Interactive and Non-Malleable Commitment*, Proc. 30th Annual ACM Symposium on the Theory of Computing, Dallas, 1998, pp. 141–150.

[27] C. Dwork and M. Naor, *An Efficient Existentially Unforgeable Signature Scheme and its Applications*, Journal of Cryptology, vol 11, 1998, pp. 187-208. Preliminary version: Advances in Cryptology – Crypto'94 Proceeding, Springer-Verlag, Lecture Notes in Computer Science 839, 1994, pp. 234–246.

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

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

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

[31] 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).

[32] U. Feige, D. Lapidot and A. Shamir, *Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String*, Proc. of 31st IEEE Symposium on Foundations of Computer Science, 1990, pp. 308–317.

[33] M. J. Fischer, N. A. Lynch, *A Lower Bound for the Time to Assure Interactive Consistency*, IPL 14(4), 1982, pp. 183–186.

[34] Z. Galil, S. Haber and M. Yung, *Interactive Public-key Cryptosystems*, see *Symmetric Public-Key Encryption*, Advances in Cryptology – Crypto'85, Lecture Notes in Computer Science No. 218, Springer-Verlag, 1986, pp. 128–137.

[35] Goldreich, O., **Foundations of Cryptography** (Fragments of a Book) 1995. Electronic publication: http://www.eccc.uni-trier.de/eccc/info/ECCC-Books/eccc-books.html (Electronic Colloquium on Computational Complexity).

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

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

[38] O. Goldreich and H. Krawczyk, *On the Composition of Zero-knowledge Proof Systems*, Siam J. on Computing 25, 1996, pp. 169–192.

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

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

[41] O. Goldreich and Y. Oren, *Definitions and Properties of Zero-Knowledge Proof Systems*, J. Cryptology 6, 1993, pp. 1–32.

[42] O. Goldreich and E. Petrank, *Quantifying Knowledge Complexity*, Proc. of 32nd IEEE Symposium on Foundations of Computer Science, 1991, pp. 59–68.

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

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

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

[46] S. Goldwasser, S. Micali and P. Tong, *Why and How to Establish a Private Code on a Public Network*, Proc. of the 23rd IEEE Symposium on the Foundation of Computer Science, 1982, pp. 134-144.

[47] I. Impagliazzo and M. Luby, *One-way functions are essential to computational based cryptography*, Proc. 21st ACM Symposium on Theory of Computing, 1989.

[48] I. Impagliazzo, L. Levin and M. Luby, *Pseudo-random generation from one-way functions*, Proc. 21st ACM Symposium on Theory of Computing, 1989.

[49] J. Kilian, *On the complexity of bounded-interaction and non-interactive zero-knowledge proofs*, Proc. of the 35th IEEE Symposium on the Foundation of Computer Science, 1994, pp. 466–477.

[50] J. Kilian and E. Petrank, *An efficient non-interactive zero-knowledge proof system for NP with general assumptions*, Journal of Cryptology, vol 11, 1998, pp. 1–27.

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

[52] Luby M., **Pseudo-randomness and applications**, Princeton University Press, 1996.

[53] U. Maurer, *Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete algorithms*, Advances in Cryptology – Crypto'94, Lecture Notes in Computer Science No. 839, 1994, pp. 271–281.

[54] 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.

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

[56] M. Naor and O. Reingold, Synthesizers and their application to the parallel construction of pseudo-random functions, *Proc. 36th IEEE Symp. on Foundations of Computer Science*, 1995, pp. 170-181.

[57] M. Naor and O. Reingold, *Number-Theoretic Constructions of Efficient Pseudo-Random Functions*, Proc. of the 38th IEEE Symposium on the Foundation of Computer Science, 1982, pp. 80–91.

[58] M. Naor and O. Reingold, *From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs*, Advances in Cryptology – Crypto'98 Proceeding, Lecture Notes in Computer Science No. 1462, Springer-Verlag, 1998, pp. 267–282.

[59] M. Naor and A. Wool, *Access Control and Signatures via Quorum Secret Sharing* Proc. Third ACM Conference on Computer and Communications Security, 1996, pp. 157–168.

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

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

[62] M. O. Rabin, *Randomized Byzantine Generals*, Proc. of the 24th IEEE Symposium on the Foundation of Computer Science, 1983, pp. 403–409.

[63] 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.

[64] 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.

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

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

[67] M. Yung, *Cryptoprotocols: Subscription to a Public Key, the Secret Blocking and the Multi-Player Mental Poker Game*, Advances in Cryptology – Crypto'84, Lecture Notes in Computer Science No. 196, pp. 439–453, 1985.