The Non-Malleability Lectures

Cynthia Dwork*

Tentative Outline

1. One-way functions, trapdoor functions, hard bits: negligible and overwhelming probabilities, feasible computations and BPP, uniform and non-uniform polynomial time, collections of one-way functions/permutations/trapdoor permutations, one-time signature schemes, hard bits

2. Pseudo-Randomness A. sequence generators unpredictability indistinguishability from random equivalence of the two definitions Statement: pseduo-randomness from any 1-way function

   B. functions Goldreich, Goldwasser, and Micali construction * synthesizers and Naor-Reingold construction

3. Public-Key encryption I A. Definitions of attack B. Definitions of break (why "secrecy" is not enough); include coin-flipping and bidding examples C. Goldwasser-Micali construction based on quadratic residuousity D. Semantic Security with Respect to Relations E. Definition of nm-cpa and DDN Lite

4. hash functions A. universal one-way vs. collision intractable; NY1 B. Left-over Hash Lemma C. Impagliazzo-Naor construction based on subset-sum D. average-case vs. worst-case: explain the issue and state implications of Ajtai's work to IN; GGH1.

5. Commitment I A. Definition B. Applications C. Naor's construction

6. Zero-Knowledge I A. Basic Defs (ordinary and BB) B. NP $\in$ ZK; C. NP $\in$ ZK(5)

7. Zero-Knowledge II A. Trapdoor bit commitment B. Proofs of Knowledge C. Witness Indistinguishability and Witness Hiding D. NP $\in$ ZK(4) E. NIZKs

8. Public-Key encryption II A. Naor-Yung chosen-ciphertext-preprocessing B. Dolev, Dwork, and Naor chosen-ciphertext-postprocessing * C. Cramer-Shoup

9. Commitment II A. non-malleable commitments (DDN; DIO)

*IBM Almaden Research Center, 650 Harry Rd., San Jose, CA. 95120, dwork@almaden.ibm.com.
Lecture notes prepared for cs359 (Foundations of Cryptography), Stanford University, Spring Quarter, 1999.

1 Basics

1.1 BPP and Feasible Computations

Throughout these notes we assume a binary alphabet, unless otherwise indicated. The *length* of x, denoted $|x|$, is the number of bits in the binary encoding of x (if x is a string then $|x|$ is just the number of bits in x).

A *probabilistic polynomial time* Turing machine is a Turing machine that runs in a fixed polynomial in the length of its input and that may make *random moves*. We can think of the random moves as being chosen by coin-flipping or we can think of the machine as having an auxiliary read-only input tape containing random bits. The two formulations are equivalent. If M is a probabilistic Turing machine, then $M(x)$ denotes a probability distribution, where the probability space is over the random choices made by M, describing the behavior of M on input x. Note that the randomization is *not* over the input x; it is only over the coin flips of M.

The class BPP is the set of languages L that are recognized in probabilistic polynomial time; that is there exists a probabilistic polynomial time Turing machine M such that

$$\text{for all } x \in L,\ \Pr[M(x) = 1] \geq \tfrac{2}{3}$$
$$\text{for all } x \notin L,\ \Pr[M(x) = 0] \geq \tfrac{2}{3}$$

where the probability space is over the coin flips of M. The constant $2/3$ is almost irrelevant, what is "bounded" here is the distance away from an acceptance/rejection probability of $\frac{1}{2}$.

Exercise 1.1 *Replace $\frac{2}{3}$ with $\frac{1}{2} + \varepsilon$, where ε is any constant in $(0, \frac{1}{2})$.*

Exercise 1.2 *Replace $\frac{2}{3}$ with $1 - \varepsilon(n)$, (1) for $\varepsilon(n) = n^{-c}$ and (2) for $\varepsilon(n) = 2^{-n}$.*

Throughout these notes, a "feasible" computation is one that can be carried out in probabilistic polynomial time with *negligible* error probability (error probability that decreases faster than the inverse of any polynomial in n). We will use the notation $\nu(n)$ to denote any function for which, for every $c > 0$ there exists n_0 such that for all $n \geq n_0$

$$\nu(n) < \frac{1}{n^c}$$

When an event happens with probability at least $1 - \nu(n)$ we say it occurs with *overwhelming* probability.

1.2 Uniformity

By the term *uniform* probabilistic polynomial time algorithm we just mean a randomized algorithm that runs in polynomial time in the length of its input. The term "uniform," which is usually omitted, refers to the fact that there is a *single* algorithm, or Turing Machine, that takes inputs of any size.

Intuitively, a *non-uniform* polynomial time algorithm is actually a collection of deterministic polynomial time bounded algorithms, one for each input size. However, we cannot let the description of each algorithm be too large as a function of the input size. The concept is equivalent to a *polynomial-size family of circuits*, formally defined in Definition 2.2. Alternatively, we can think of a single algorithm that receives a certain piece of "advice," which depends only on the size of the input and not on the value of the input. Moreover, the amount of advice, or its *length*, is polynomially bounded in the length of the input.

The use of "(non-)uniformity" is over instance size. People also discuss uniformity over *instances* or *inputs* of a given size. The meaning is different. We assume we have a (uniform or non-uniform) algorithm A, and we talk about its behavior on inputs that are chosen *not* by an omniscient being, who may be able to find the very worst inputs to the algorithm, but instead according to a polynomial-time sampleable distribution (defined below). For example, if we choose public keys for a public-key cryptosystem, it suffices that every polynomial-time bounded adversary should fail with overwhelming probability (over the choice of key) to break the system. It is not necessary (and not to possible!) to require that every adversary fail on every possible key choice.

Finally, the term "uniformity" (over inputs) should not be confused with the *uniform probability distribution* on the elements of a finite set S. The uniform probability distribution assigns to element in S the probability $1/|S|$.

1.3 One-Way Functions and Hard Core Predicates

As discussed above, informally, a one-way function is easy to compute and hard to invert. "Easy to compute" is easy to formalize, for example, $f(x)$ can be computed in polynomial time for all x. "Hard to invert" requires some care, and defining this concept can be viewed as a "warm-up" exercise for defining security of a cryptographic primitive.

We restrict our attention to one-way functions that are *length-preserving*: $|f(x)| = |x|$. Note that in any case if a function f is too length-reducing, in the sense that $f(x)$ is of length subpolynomial in $|x|$, then it trivially cannot be inverted in polynomial time, since the length of $f^{-1}(y)$ would not be polynomial in the length of y.

It makes no sense to say that no polynomial time algorithm can invert *any* y or even infinitely many y. To see this, consider a machine M that on input y computes $z = f(1^{|y|})$ and checks to see if $z = y$. If so, then $M(y)$ outputs $1^{|y|}$. Similarly, it is insufficient to say that inverting f is not in BPP, for this only means at best that every probabilistic polynomial time algorithm A fails to invert on some x of length n, for infinitely many values of n.

Intuitively, we want A to fail almost all the time: for all sufficiently large n A should fail on essentially all inputs of length n. One way of saying this is to assume a probability space on the *inputs* to A (values to be inverted) and to require subpolynomial probability of success, where the probability space is over *both* the inputs and the coin flips of A.

Definition 1.1 (One-Way Function) *The function* $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ *is* one-way *if for all probabilistic polynomial time bounded Turing machines* A*,*

$$\Pr[A(f(x)) \in f^{-1}(f(x))] \leq \nu(n)$$

where for every n *the probability space is over the choice of* x *and the coin flips of* A*.*

Note that in this definition, the probability space on the inputs to A is implicitly defined by the probability space on the input to f. We need not assume that x is chosen with uniform probability over strings of length n (that is, that all strings of length n are equally likely). However, we do assume that the distribution is polynomial-time sampleable, defined as follows:

Definition 1.2 (Polynomial-Time Sampleable Distribution) *A family of probability distributions* $\{\pi_i\}_{i=1,2,\ldots}$ *is polynomial-time sampleable if there exists a probabilistic polynomial time machine* M *such that for all* n *and all* $x \in \{0,1\}^n$*,*

$$\Pr[M(1^n) = x] = \pi_n(x)$$

The above definition for "hard to invert" requires that every probabilistic polynomial-time A will fail to invert f with high probability. An alternative is to require that A will fail with *non-negligible* probability. These definitions are "existentially" equivalent: from any function for which all polynomial-time inverters fail with non-negligible (that is, polynomial) probability, we can construct a new function for which all polynomial-time inverters fail with overwhelming probability.

Exercise 1.3 *Formalize and prove the last setence.*

An immediate application of one-way functions is to a *one-time signature scheme*. In such a scheme the signer publishes a one-time signature verification key. Later, the signer may sign exactly one message, which anyone can verify using the pubic key.

The public key consists of a description of a one-way function f together with k pairs of values:

$$\begin{array}{cccc} f(x_1^0) & f(x_2^0) & \dots & f(x_k^0) \\ f(x_1^1) & f(x_2^1) & \dots & f(x_k^1) \end{array}$$

To sign the k-bit message $m = m_1 \dots m_k$, the signer releases the pre-images of the values selected by the bits of m:

$$x_1^{m_1}, x_2^{m_2}, \dots, x_k^{m_k}$$

Exercise 1.4 *Define security for a one-time signature scheme and prove that this construction is secure under your definition.*

Until now, we have defined one-way functions so that asymptotically these functions are hard to invert; thus, in particular, for every probabilistic polynomial-time bounded algorithm "attempting" to invert $f(x)$, the probability (over choice of x and coin flips of A) that A succeeds drops superpolynomially as k grows. So for every integer d there exists a k_d such that for all $k \geq k_d$, A will succeed in inverting f_k with probability strictly less than $\frac{1}{k^d}$. We can go one step further and require *non-uniform* one-wayness.

Definition 1.3 (Non-Uniform One-Way Functions) *A polynomial-time computable function f is* non-uniform one-way *if for every non-uniform polynomial-time algorithm A and every constant $c > 0$ there exists n_d such that for all $n \geq n_d$:*

$$\Pr[A(f(x)) \in f^{-1}(f(x))] < \frac{1}{n^c}$$

where the probability space is over the input $x \in_R \{0,1\}^n$.

Exercise 1.5 *Why do we not consider* randomized *non-uniform algorithms?*

Definition 1.4 (Collection of One-Way Functions) *A collection of functions $\{f_i : D_i \mapsto \{0,1\}^*\}_{\{i \in S\}}$ is one-way if there exist probabilistic polynomial time algorithms I, D, and F, satisfying the following conditions.*

1. - *Algorithm $I(1^n)$ computes an index $i \in S$ chosen (not necessarily uniformly) from $S \cap \{0,1\}^n$.*
   - *Algorithm $D(i)$ samples an element (not necessarily uniformly) from D_i.*
   - *Algorithm $F(i, x)$ outputs $f_i(x)$.*
2. *For every probabilistic polynomial time algorithm A and every polynomial $p(\cdot)$, if n is sufficiently large then*

$$\Pr[A(i, f_i(x)) \in f_i^{-1}(f_i(x))] < \frac{1}{p(n)}$$

   where the probability space is over $i \in I(1^n)$, $x \in D_i$.

Note that the definition says nothing about $|S \cap \{0,1\}^n|$: it might contain just a single element or exponentially many in n. The definition is usually relaxed so that $I(1^n)$ produces indices of length $p(n)$ for some polynomial $p(\cdot)$, and $I(1^n)$ may occasionally (subpolynomially often) produce indices not in S.

Theorem 1.1 *There exists a collection of one-way functions if and only if there exists a one-way function.*

Definition 1.5 (Collection of Trapdoor Permutations) *Let I be a probabilistic algorithm and let $I(1^n)$ be written as $I_1(1^n)I_2(1^n)$, where $|I_1(1^n)| = |I_2(1^n)|$. The triple (I, D, F) of probabilistic polynomial time algorithms forms a* collection of trapdoor permutations *if:*

1. *(I_1, D, F) is a collection of one-way permuations (one-way 1-1 length-preserving functions);*
2. *Let $i_1 i_2$ be chosen according to $I(1^n)$. Then i_2 is a* trapdoor *with which inverting $F(i_1, x)$ is easy. More formally, there exists a deterministic polynomial time algorithm F^{-1} so that, for every (i, t) in the range of I, and for every $x \in D_i$,*

$$F^{-1}(t, F(i, x)) = x$$

The definition is typically relaxed so that the requirements hold with overwhelming probability.

Definition 1.4 is uniform. It can also be modified so as to require uninvertability even by non-uniform polynomial-time bounded algorithms. Definition 1.5 *cannot* (reasonably) be made non-uniform, since for any trapdoor function in the collection there is some inverting circuit with the trapdoor wired into the circuit.

Hard-Core Predicates A *predicate* is a binary-valued function $B : \{0,1\}^* \longrightarrow \{0,1\}$.

Let f be a one-way function. As mentioned earlier, even though f is hard to invert it does not follow that *all* of x is hidden by $f(x)$, just that, intuitively, f hides enough of x to make finding x infeasible. The concept of a hard-core predicate allows us to formally capture and exploit the fact that *something* about x remains "hidden" by $f(x)$.

Definition 1.6 (Hard-Core Predicate) *$B : \{0,1\}^* \longrightarrow \{0,1\}$ is a* hard-core predicate *of the function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ if*

1. *B is computable in polynomial time; and*

2. for every probabilistic polynomial time algorithm A,

$$\Pr[A(f(x)) = B(x)] \leq \frac{1}{2} + \nu(n)$$

where the probability space is over the choice of x and the coin flips of A.

This is a uniform definition of hardness (we consider probabilistic polynomial-time algorithms A). Sometimes the definition of hard-core predicates is strengthened by requiring strenght against non-uniform polynomial-time algorithms (families of circuits).

Theorem 1.2 ([47]) *Let f be a length-preserving one-way function. For all x and r with $|x| = |r|$, let $g(x,r) = (f(x), r)$. Let $b(x,r) = x \cdot r$, where the notation $y \cdot z$ means: view the strings y and r as $|y|$-bit vectors and compute their inner product modulo 2. Then b is a hard-core predicate for g.*

The proof is long, and will not be presented here.

Intuitively, the theorem says that for any polynomial-time bounded "adversary" $\mathcal{A}$, the probability that $A(f, f(x), r)$ outputs $b(x,r)$ is subpolynomially close to 1/2. The probability space is over choice of x, r and the coin flips of A. The value $b(x,r)$ is sometimes called a *Goldreich-Levin bit* or a *Goldreich-Levin hard bit.* Note that it is "hard" even when r is known.

2 Pseudo-Randomness

A *pseudo-random* object "looks" random, in the sense that it cannot be distinguished from a truly random object in polynmial time. Informally, a *pseudo-random sequence* generator takes a small truly random *seed* and stretches it into a longer pseudo-random string. Still informally, a *pseudo-random function* generator stretches a seed into a function, with the property that an adversary, obtaining the value of the function a an adaptively chosen polynomial-sized set of points, cannot predict the value of the function at a new point, or even distinguish the value of the function at that point from a truly random value.

Pseudo-random objects play an important role in many constructions of cryptographic primitives, such as encryption schemes and string commitment (pseudo-random sequences) and identification schemes (pseudo-random functions).

2.1 Pseudo-Random Sequence Generators: Definitions

The basic theory of cryptographically strong pseudo-random generators is due to Blum and Micali [12] and to Yao [78]. There are two definitions, based, respectively, on *indistinguishability* from truly random strings and on *unpredictability*: given k bits in a pseudo-random sequence it is hard to guess the $(k+1)$st bit with probability polynomially better than 1/2. We will present the two definitions and prove their equivalence. The notes in this section follow the presentations in [16] and [76].

Definition 2.1 (ensemble) *An* ensemble S over $\{0,1\}$ *is a sequence* $\{S_k\}$ *such that each* S_k *is a probability distribution on* $\{0,1\}^k$*. The random ensemble* $R = \{R_k\}$ *is the sequence of uniform distributions, that is, for all* $x \in \{0,1\}^k$*,*

$$R_k(x) = 2^{-k}$$

Definition 2.2 (polynomial-size family of circuits) *A* polynomial-size family of circuits *is a sequence of circuits* $C = \{C_k\}$ *such that for some positive integer* d*,* C_k *has at most* k *inputs and at most* k^d *gates.*

Definition 2.3 (predicting collection) *Let* S *be an ensemble. A* predicting collection *is a polynomial-size family of circuits* C *such that each* C_k *has* strictly *less than* k *inputs and one output bit.*

Fix an ensemble S and a predicting collection C. For $k \geq 1$ let π_k denote the probability that, on input the first i bits of $s \in_R S_k$, the circuit C_k outputs the $(i+1)$st bit of s. The probability space is over the choice of s. We say that the ensemble S is *unpredictable by* C if for all positive integer constants d and all sufficiently large k (i.e., $\exists k_d \forall k \geq k_d$):

$$\pi_k < \frac{1}{2} + \frac{1}{k^d}$$

Note that the inequality is strict. So in other words, C fails to predict S if C_k's probability of success of guessing the next bit, given any prefix of at most $k-1$ bits of a randomly chosen element in S_k is subpolynomially (in k) close to $\frac{1}{2}$.

Unpredictability gives us our first definition of pseudo-randomness of an ensemble.

Definition 2.4 (next bit test) *The ensemble* S passes the next bit test *if* S *is unpredictable by all predicting collections.*

Clearly, the truly random ensemble R passes the next bit test.

Definition 2.5 *If* C_k *is a circuit with* k *inputs and one output bit, and* S *is any ensemble, define*

$$C_k(S) = \Pr[C_k \textit{ outputs 1 on } s \in_R S_k]$$

where the probability space is over the choice of s*.*

Generally speaking, a statistical test is an algorithm that on any input produces a single output bit. We give an analagous definition for circuits.

Definition 2.6 (polynomail-size statistical test) *A* polynomail-size statistical test *is a polynomial-size family of single-output circuits.*

Definition 2.7 *An ensemble S* passes *test $T = \{T_k\}$ if $\forall d\, \exists k_d\, \forall k \geq k_d$:*

$$|T_k(S) - T_k(R)| < \frac{1}{k^d}$$

The concept of passing a statistical test is always relative to the random ensemble R: S passes the particular test T if T cannot distinguish S from R in a "reasonable" (i.e., polynomial) amount of time. In other words, it is impossible to distinguish S from R by observing the behavior of T on any polynomial number of samples of the two input distributions.

Definition 2.8 (polynomial indistinguishability of distributions) *Ensembles S and S' are* polynomially indistinguishable *if for all polynomial-size statistical tests T, $\forall d\, \exists k_d\, \forall k \geq k_d$:*

$$|T_k(S) - T_k(S')| < \frac{1}{k^d}$$

S is *pseudo-random* if S is polynomially indistinguishable from R, i.e., S passes all polynomial-size statistical tests T.

The presentation of the following theorem follows that in [16].

Theorem 2.1 *[78] An ensemble S passes the next bit test if and only if S is pseudo-random.*

Proof: If x is a string of length k, then we let x_i denote the ith bit of x and $x_{[i \ldots j]}$ denotes $x_i x_{i+1} \ldots x_j$, $1 \leq i \leq j \leq k$.

Suppose an ensemble S fails the next bit test. Then there exists a predicitng collection C and an integer d such that for infinitely many values of k there exists $i < k$ such that C_k predicts the $(i+1)$st bit of $s \in_R S_k$ with probability at least $\frac{1}{2} + \frac{1}{k^d}$. We will use C to build a statistical test $T = \{T_k\}$ that distinguishes S from R.

We define T_k as follows. Given $x = x_1 \ldots x_k$, T_k first runs C_k on input $x_{[1 \ldots i]}$; let y be the output bit of C_k. If $y = x_{i+1}$ then T_k outputs 1, otherwise T_k outputs 0.

If $x \in S_k$, then T_k accepts with probability at least $\frac{1}{2} + \frac{1}{k^d}$ (by the assumption that S fails the next bit test T). On the other hand, if $x \in R_k$ then T_k accepts with probability exactly $\frac{1}{2}$. Hence,

$$T_k(S) - T_k(R) \geq \frac{1}{k^d}$$

for infinitely many k; that is, S fails the statistical test T.

Conversely, suppose S fails a statistical test T. We wish to show that S is predictable (fails the next bit test).

If S fails the test T then, by definition, there exists d such that for infinitely many k, $|T_k(S) - T_k(R)| \geq \frac{1}{k^d}$. Let K be the set of integers k satisfying the above inequality. Fix any $k \in K$ and, without loss of generality, assume $T_k(S) > T_k(R)$.

We will use a *hybrid* argument to show the existence of a randomized algorithm that, for some $i < k$ that, given the first i bits of $x \in_R S_k$, predicts x_{i+1}. We then convert the randomized algorithm into a polynomial-size family of predicting circuits for S.

For $0 \leq j \leq k$, let p_j denote the probability that $T_k(x_{[1\ldots j]}y) = 1$, given that $x \in_R S_k$ and y is a string of $k - j$ truly random bits ($y \in_R R_{k-j}$). Then $p_0 = T_k(R_k)$ and $p_k = T_k(S_k)$. By assumption,

$$p_k - p_0 \geq \frac{1}{k^d}$$

Thus, by a pigeonhole argument $\exists i$, $0 \leq i < k$, such that

$$p_{i+1} - p_i \geq \frac{1}{k \cdot k^d} = \frac{1}{k^{d+1}}$$

We define a randomized algorithm A as follows.

1. On input $(1^k, x_{[1\ldots i]})$, A flips $k - i$ coins to obtain $y = y_{1\ldots k-i} \in_R R_{k-i}$.
2. A runs T_k on input $x_{[1\ldots i]}y$. If $T_k(x_{[1\ldots i]}y) = 1$, then A predicts $x_{i+1} = y_1$. If $T_k(x_{[1\ldots i]}y) = 0$, then A predicts $x_{i+1} = \overline{y_1}$.

We now analyze the probability that A is correct. Let

$$q = \Pr[T_k(x_{[1\ldots i]}\overline{x_{i+1}}r) = 1 \mid x \in_R S_k \text{ and } r \in_R R_{k-i-1}]$$

Note that

$$p_i = \frac{p_{i+1} + q}{2}$$

To see this, recall that

$$p_i = \Pr[T_k(x_{[1\ldots i]}y) = 1 \mid x \in_R S_k \text{ and } y \in_R R_{k-i}]$$

Since the bit y_1 is random, half the time we have $y_1 = x_{i+1}$, so this case contributes $\frac{1}{2}p_{i+1}$ to p_i. Half the time $y_1 = \overline{x_{i+1}}$, so this case contributes $\frac{1}{2}q$ to p_i.

Claim 2.1 *The probability that $A(1^k, x_{[1\ldots i]})$ correctly predicts x_{i+1} is at least $\frac{1}{2}+\frac{1}{k^{d+1}}$, where the probability space is over the choice of $x \in_R S_k$* and *the coin flips of A (choice of $y \in_R R_{k-i}$).*

Proof: If $y_1 = x_{i+1}$, which occurs with probability $\frac{1}{2}$, then T_k accepts with probability p_{i+1} and A outputs y_1. If $y_1 = \overline{x_{i+1}}$, which occurs with probability $\frac{1}{2}$, then T_k rejects with probability $1-q$ and A guesses $\overline{y_1} = x_{i+1}$. Thus, the probability, taken over choice of y and x, that A correctly predicts x_{i+1} is

$$\begin{aligned}
& \frac{1}{2}p_{i+1} + \frac{1}{2}(1-q) && (1)\\
= \; & \frac{1}{2}[p_{i+1} + 1 - (2p_i - p_{i+1})] && (2)\\
= \; & \frac{1}{2}p_{i+1} + \frac{1}{2} - p_i + \frac{1}{2}p_{i+1} && (3)\\
= \; & \frac{1}{2} + p_{i+1} - p_i && (4)\\
\geq \; & \frac{1}{2} + \frac{1}{k^{d+1}} && (5)
\end{aligned}$$

■

Thus, the randomized algorithm A has a polynomial advantage in predicting S. We now convert A into a family of (deterministic) predicting circuits.

The probability over all i-bit prefixes and $(k-i)$-bit extensions that A answers correctly is the average over all 2^{k-i} extensions z of the probability (over the prefixes) that A is correct when A extends its input with z. Moreover, there exists an extension z for which the probability (over $x_{[1\ldots i]}$) that A answers correctly with extension z is at least average. Fix such an extension z (note that z may be hard to find deterministically). Since the algorithm A runs in polynomial time (in k), it can be converted into a circuit A_k with at most k^d gates, for some $d \in O(1)$. Hard-wire z, i, and which of p_k, p_0 is larger, into A_k.

In a similar way, we can build A_k for all $k \in K$. For all $k \notin K$, A_k can be defined arbitrarily. By construction, S fails the statistical test $A = \{A_k\}$. ■

Remark 1 *We have proved a* non-uniform *formulation of Theorem 2.1, where a statistical test and a predicting collection are defined as a families of circuits. Alternatively, we could define a statistical test and a predictor to be probabilistic polynomial time algorithms. The uniform version of Theorem 2.1 says that the two uniform notions (passing all statistical tests and unpredictability) are equivalent.*

Exercise 2.1 *State and prove the uniform version of Theorem 2.1.*

Definition 2.9 (Pseudo-Random Sequence Generator) *Let Q be a polynomial, I a set of inputs, and $I_k \subseteq I$ the set of inputs of length k. Let G be a deterministic algorithm that, on input a seed $x \in I_k$, outputs a $Q(k)$-bit sequence s_x. Let*

$$S_k = \{s_x \mid x \in_R I_k\}.$$

G is a cryptographically strong pseudo-random sequence generator *if the ensemble $S = \{S_k\}$ is pseudo-random.*

Henceforth "pseudo-random" will mean "cryptographically strong pseudo-random." The particular polynomial Q in the definition above is in some sense irrelevant:

Theorem 2.2 *[48] Let $1 < c_1 < c_2$ be integer constants. If there exists a pseudo-random sequence generator that stretches a k-bit seed to a k^{c_1}-bit output, then there exists a pseudo-random sequence generator that stretches a k-bit seed to a k^{c_2}-bit output.*

Caveat. A pseudo-random sequence $g(s)$ is only pseudo-random if the seed s is secret. In particular, it is clear that if s is known then $g(s)$ is completely predictable.

2.2 Pseudo-Random Sequence Generators: Constructions

As mentioned earlier, pseudo-random sequence generators can be constructed from any one-way function [56]. The proof is difficult. We will show a weaker result: a pseudo-random generator can be constructed from any one-way 1-1 length-preserving function. This is essentially the result and proof in [12].

Let $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ be non-uniform one-way, 1-1, and length preserving. Let B be a non-uniform hard-core predicate for f. Let $G(x)$ be defined as follows. Let $n = |x|$ and let $p(n) > n$ be a fixed polynomial.

$$G(x) = B(f(x))B(f^{(2)}(x))\ldots B(f^{(p(n))}(x))$$

where $f^{(i)}$ denotes the ith iterate of f.

Theorem 2.3 *G is a pseudo-random sequence generator.*

Proof: Let S_n be the probability distribution on $p(n)$-bit strings output by G on inputs in $x \in \{0,1\}^n$, where the probability space is over choice of x. If G is not pseudo-random (under the non-uniform definition), then G fails some statistical test $T = \{T_k\}$. That is, there exists an integer d, an infinite family K of indices, so that

$$|\Pr[T_{p(k)}(R_{p(k)})] - \Pr[T_{p(k)}(S_k)]| > \frac{1}{k^d}$$

Note that the distribution on elements in S_k need not be not uniform; for example, it may be the case that several different choices of s yield the same sequence. Let S_k^R denote the probability distribution on *reversed* strings in S_k, where a string s appears in S_k^R with exactly the same probability that its reverse appears in S_k. Clearly, there exists a statistical test T^R so that $S^R = \{S_k^R\}$ fails T^R.

By Theorem 2.1, S^R fails the next bit test. Thus, there exists an integer d, an infinite family K of indices, and a predicting collection C such that for all $k \in K \, \exists j < p(k)$, the probability that $C_{p(k)}$, on input $s_{[1\ldots j]}$, outputs s_{j+1} is at least $\frac{1}{2} + \frac{1}{k^d}$, where the probability space is over choice of $s \in S_k^R$.

Consider the following deterministic algorithm A for guessing $B(x)$ from $f(x)$, assuming j is known:

1. On input $f(x)$, compute $f^{(2)}(x), f^{(3)}(x), \ldots, f^{(j)}(x)$ and output the sequence $s_{[1\ldots j]} = B(f^{(j)}(x)), B(f^{(j-1)}(x)), \ldots, B(f^{(2)}(x)), B(f(x))$.
2. Let $y = C(s_{[1\ldots j]})$. Output y as the guess of $B(x)$.

By assumption, the probability over $s \in S_k^R$ that A will correctly guess s_{j+1} is at least $\frac{1}{2} + \frac{1}{k^d}$.

We are almost done. However, so far we have only a statement about probabilities over strings in S_k^R, and to contradict the assumption that B is a hard-core predicate for $f(x)$ we need a statement about probabilities over $x \in \{0,1\}^k$. For any event E,

$$
\begin{aligned}
\Pr_s[E] &= \sum_{s \in S_k^R} \Pr[E \mid s] \cdot \Pr[s] && (6)\\
&= \sum_{s \in S_k^R} \Pr[E \mid s] \cdot \Pr_{x \in \{0,1\}^k}[s \mid x] \cdot \Pr[x] && (7)\\
&= \sum_{s} \sum_{x} \Pr[E \mid s] \Pr[s \mid x] \cdot \Pr[x] && (8)\\
&= \sum_{x} \left(\sum_{x} \Pr[E \mid s] \Pr[s \mid x] \right) \Pr[x] && (9)\\
&= \sum_{x} \Pr[E \mid x] \cdot \Pr[x] && (10)\\
&= \Pr_x[E] && (11)
\end{aligned}
$$

Thus, the probability over x that A succeeds in computing $B(x)$ from $f(x)$ is at least $\frac{1}{2} + \frac{1}{k^d}$, contradicting the asumption that B is a hard-core predicate for f. Finally, $C_{p(k)}$ can have j wired in. ∎

Exercise 2.2 *Let $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ be non-uniform one-way, 1-1, and length preserving. Let B be a non-uniform hard-core predicate for f. Let $n = |x|$ and let*

$p(n) > n$ be a fixed polynomial. For $i = 1, 2, \ldots$, define $b_i = B(f^{(i)}(x))$. Let $G(x)$ be defined as above:

$$\begin{aligned} G(x) &= B(f(x))B(f^{(2)}(x)) \ldots B(f^{(p(n))}(x)) && (12) \\ &= b_1 b_2 \ldots b_{p(n)} && (13) \end{aligned}$$

Let $G'(X) = G(x) \circ f^{(p(n)+1)}(x)$, (where $\circ$ denotes concatenation); that is,

$$G'(x) = b_1 b_2 \ldots b_{p(n)} f^{(p(n)+1)}(x)$$

Prove that $G'(x)$ is a pseudo-random sequence generator.

Hint: First prove that the sequence $G'(x) = B(x)f(x)$ is pseudo-random. Note that $G''(x) = B(f^{(k-1)})(x)f^{(k)}(x)$ is distributed exactly as is $G'(x)$. Then construct a hybrid argument in which one extreme is the sequence our candidate pseudo-random sequence, $b_1 \ldots b_{k-1} f^{(k)}(x)$, and the other extreme is $r_1 r_{k-2} \ldots b_1 f^{(k)}(x)$, where each $r_i \in_R \{0, 1\}$. The proof of the heart of the hybrid argument is essentially the same as the proof of Theorem 2.3.

2.2.1 Aside: Application to Encryption

As remarked above, we cannot sensibly speak of *non-uniform* trapdoor functions or collections of trapdoor functions. However, these can be defined in the uniform sense, and the corresponding generator constructed as in Exercise 2.2 (using a collection of *uniform* trapdoor functions) immediately yields the following public-key cryptosystem.

Let (I, D, F) be a collection of trapdoor permutations, as in Definition 1.5, and let $B = \{B_i\}$ be a collection of hard-core predicates, where for each $i \in I$, B_i is hard-core for f_i. Let t_i denote the trapdoor information for f_i. Recall that algorithm $I(1^n)$ computes an index $i \in S$ chosen (not necessarily uniformly) from $S \cap \{0, 1\}^n$. For simplicity, we let $f = f_i$ and $B = B_i$. We may assume f and B have short descriptions (such as a programs for computing $f(x)$ and $B(x)$, respectively). The public key will consist of the descriptions of f and B, and the security parameter 1^n. To encrypt a $(k-1)$-bit message m, the sender chooses a random $x \in \{0, 1\}^n \cap Dom(f)$ and computes

$$G(x) = B(f(x)) \ldots B(f^{(k-1)}(x))f^{(k)}(x)$$

which we write as $b1 \ldots b_{k-1} f^{(k)}(x)$ and sends the pair

$$(m \oplus b_1 \ldots b_{k-1}, f^{(k)}(x))$$

where the first component is the bitwise exclusive-or of m and the pseudo-random pad $b_1 \ldots b_{k-1}$.

To decrypt a pair (z, y), the owner of the public key uses the trapdoor information to obtain $f^{(k-1)}(x) \ldots f(x)$ by repeatedly inverting $f(y)$. From there, it is easy to compute the pad $B(f^{(k-1)}(x)) \dots B(f(x))$ and to exclusive-or the pad with z to obtain m.

We will not now formally prove correctness of this scheme, since we have not yet presented a definition of security for a cryptosystem. However, we make the following remarks.

1. The cryptosystem is highly *malleable*: given an encryption of a message m it is trivial to construct an encryption of a message related to m, even without knowing m. For example, an adversary can flip the last bit of the ciphertext. Malleability is *not* just an issue of integrity of ciphertext. Rather, it is a basic security issue for encryption, even under chosen plaintext attack. This is discussed in Sections 3 and refencryption II.

2. This scheme is in some sense analgous to encryption by bitwise exclusive-or with a *one-time pad*, which is secure *information-theoretically* (hence, even against eavesdroppers with arbitrary computational power). However, here the pad is pseudo-random rather than truly random (hence, the adversary must be polynomial-time bounded in the current scheme). In the traditional (unbounded adversary) scenario, the sender and receiver must agree on the pad (in person) before the message is sent. The scheme described here requires no previous agreement on the pad or even on a seed to the pseudo-random generator. Moreover, no explicit encryption of the seed is transmitted.

Return to Pseudo-Random Sequence Generators The generator G of Theorem 2.3 can be generalized to be based on a *collection of one-way permutations*. Let (I, D, F) be a collection of one-way permutations. For each $i \in I$, let B_i be a hard-core predicate for f_i. We define a generator $G(r,t)$ as follows: Let $I(1^n, r)$ denote the outcome of the sampling algorithm I in input 1^n when the coin toss sequence is r. Let $t_0 = D(i,t)$ denote the "starting point for f_i" obtained by sampling from D_i using coin toss sequence t. For $1 \leq j \leq p(|t|)$, let $\tau_j = B(t_{j-1})$ and $t_j = f_i(s_{j-1})$. That is, t_j is the jth iterate of f applied to t_0. Then

$$G(r,t) = \tau_1 \dots \tau_{p(n)}$$

The following theorem appears in [43] (see also [12]):

Theorem 2.4 *If for all i in the range of $I(1^n)$ the distribution $D(i)$ is uniform over D_i, then for $p(n) > |s| + |r|$ G is a cryptographically strong pseudo-random sequence generator.*

Concrete Suggestions One possibility is to use the RSA function. Specifically, let $N = pq$ be the product of two large primes, and let e not divide the order of Z_{N^*}. Let $f : Z_{N^*} \longrightarrow Z_{N^*}$ be defined by $f(x) = x^e \bmod N$. Then under the *RSA Assumption* f is one-way. (Strictly speaking, we need a collection of N's and then $f = \{f_N\}$.) The least significant bit of x is a hard-core predicate for $f(x)$.

A generator based on the discrete logarithm problem appears in [12]; another, based on the difficulty of computing square roots modulo N, is described in [10].

2.3 Pseudo-Random Function Generators

A truly random function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ has no short (polynomial in n-sized) representation. Intuitively, a pseudo-random function $g = g_s : \{0,1\}^n \longrightarrow \{0,1\}^n$, specified by a short (say, $O(n)$-bit) seed, is polynomial-time indistinguishable from a truly random function, in that a polynomial-time bounded adversary, querying a function h at adaptively chosen points $x \in \{0,1\}^n$, cannot determine whether h is pseudo-random or truly random.

A beautiful application is to "Identification: Friend or Foe?" schemes (IFF). See also [**?**].

We will describe a simple construction of a pseudo-random function from any any cryptographically strong pseudo-random sequence generator [45]. This construction is sequential: it requires n applications of a sequence generator that stretches n-bit strings to $2n$-bit strings. The presentation of this material closely follows [66]. There is also a *parallelizable* pseudo-random function generator, based on a so-called *pseudo-random synthesizer* [66].

In order to formally capture the intuition for what should be achieved by a pseudo-random function generator, we must build an appropriate vocabulary. That is the goal of the next set of definitions.

Notation. Let $\mathbb{N}$ denote the set of all natural numbers. Let I^n denote the set of all n-bit strings, $\{0,1\}^n$. Let U_n denote the random variable uniformly distributed over I^n.

The following definitions are taken from [66]. See also [45, 63].

Informally, a pseudo-random function ensemble is an efficient distribution of functions that cannot be efficiently distinguished from the uniform distribution. That is, an efficient algorithm that gets a function as a black box cannot tell (with non-negligible success probability) from which of the distributions it was sampled. To formalize this, we first define function ensembles and efficient function ensembles:

Definition 2.10 (function ensemble) *Let ℓ and k be any two $\mathbb{N} \mapsto \mathbb{N}$ functions.*

An $I^\ell \mapsto I^k$ function ensemble *is a sequence* $F = \{F_n\}_{n\in\mathbb{N}}$ *of random variables, such that the random variable* F_n *assumes values in the set of* $I^{\ell(n)} \mapsto I^{k(n)}$ *functions. The* uniform $I^\ell \mapsto I^k$ function ensemble, $R = \{R_n\}_{n\in\mathbb{N}}$, *has* R_n *uniformly distributed over the set of* $I^{\ell(n)} \mapsto I^{k(n)}$ *functions.*

Definition 2.11 (efficiently computable function ensemble)
A function ensemble, $F = \{F_n\}_{n\in\mathbb{N}}$, *is* efficiently computable *if there exist probabilistic polynomial-time algorithms,* $\mathcal{I}$ *and* $\mathcal{V}$, *and a mapping from strings to functions,* ϕ, *such that* $\phi(\mathcal{I}(1^n))$ *and* F_n *are identically distributed and* $\mathcal{V}(i,x) = (\phi(i))(x)$.

We denote by f_i *the function assigned to* i *(i.e.* $f_i \stackrel{\text{def}}{=} \phi(i)$*). We refer to* i *as the key of* f_i *and to* $\mathcal{I}$ *as the key-generating algorithm of* F.

For simplicity, we concentrate in the definition of pseudo-random functions and in their construction on length-preserving functions. The distinguisher, in our setting, is defined to be an oracle machine that can make queries to a length-preserving function (which is either sampled from the pseudo-random function ensemble or from the uniform function ensemble). We assume that on input 1^n the oracle machine makes only n-bit queries. For any probabilistic oracle machine, $\mathcal{M}$, and any $I^n \mapsto I^n$ function, O, we denote by $\mathcal{M}^O(1^n)$ the distribution of $\mathcal{M}$'s output on input 1^n and with access to O.

Definition 2.12 (efficiently computable pseudo-random function ensemble)
An efficiently computable $I^n \mapsto I^n$ *function ensemble,* $F = \{F_n\}_{n\in\mathbb{N}}$, *is* pseudo-random *if for every probabilistic polynomial-time oracle machine* $\mathcal{M}$, *every polynomial* $p(\cdot)$, *and all sufficiently large* n*'s*

$$\left|\Pr\left[\mathcal{M}^{F_n}(1^n) = 1\right] - \Pr\left[\mathcal{M}^{R_n}(1^n) = 1\right]\right| < \frac{1}{p(n)}$$

where $R = \{R_n\}_{n\in\mathbb{N}}$ *is the uniform* $I^n \mapsto I^n$ *function ensemble.*

Henceforth, the term "pseudo-random functions" is used as an abbreviation for "efficiently computable pseudo-random function ensemble".

A pseudo-random ensemble can be constructed from any pseudo-random generator in a natural way. The following construction and its proof appear in [45]. Let $g : \{0,1\}^n \longrightarrow \{0,1\}^{2n}$ be a pseudo-random generator. On input $s \in \{0,1\}^n$, let $g_0(s)$, respectively $g_1(s)$ be the first, respectively last, n output bits of $g(s)$. The pseudo-random function ensemble is indexed by a seed s. For a fixed seed $s \in_R \{0,1\}^n$, for every $x \in \{0,1\}^n$, $f_s(x)$ is obtained as follows.

Computation of $f_s(x)$

Write $x = x_1 \ldots x_n$
$y := s$
For $i = 1$ to n do:
    $y := g_{x_i}(y)$
    end
Output y

In other words, if for $x = x_1 x_2 \ldots x_n$, we define $g_x(s) = g_{x_n}(\ldots g_{x_2}(g_{x_1}(s))\ldots)$, then $f_s(x) = g_x(s)$.

In general, we can think of a function $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ as a complete bineary tree of height n, with function values stored at the leaves. To find the value of the function at x, start at the tree and move from a parent to the left or right child according to the bits of x. The tree cannot be written down in its entirety in polynomial time, but any polynomial number of leaves may be examined by an oracle machine in polynomial time. In the case of the pseudo-random function f_s, we can think of the truly random seed s as being stored at the root of the tree (depth 0). In general, if the value v is stored at a vertext, then $g_0(v)$ and $g_1(v)$ are stored at the left and right children of v, respectively.

Theorem 2.5 *Let g be a pseudo-random generator mapping n-bit seeds to strings of length $2n$. The collection of functions $\{f_s\}$, $s \in_R \{0,1\}^n$ described above is an efficiently computable pseudo-random function ensemble.*

Proof: Assume for the sake of contradiction that there exists a probabilistic polynomial time bounded oracle machine T able to distinguish $f = f_s$, $s \in_R \{0,1\}^n$ from a truly random function R.

We will define a family of hybrid oracles $A_0, \ldots A_n$. On query $y = y_1 \ldots y_n$, A_i replies as follows.

> If y is the first query with prefix $y_1, \ldots y_i$, then A_i chooses $r \in_R \{0,1\}^n$, stores $(y_1 \ldots y_i, r)$, and outputs the reply $g_{y_{i+1} \ldots y_n}(r)$.
>
> If y is not the first query with prefix $y_1, \ldots y_i$, then A_i retrieves the pair $(y_1 \ldots y_i, r)$, and outputs the reply $g_{y_{i+1} \ldots y_n}(r)$.

The hybrid oracles implicitly define trees in the same way as does the pseudo-random function. The oracle A_0 is precisely the pseudo-random function (and hence is distributed exactly as f_s), while the oracle A_n replies exactly as does a truly random function queried at only a polynmial number of points. Indeed, we can think of A_i as having truly random values stored at the vertices of depth i. For every prefix $y_1 \ldots y_i$, let $r_{y_1 \ldots y_i}$ denote the random value conceptually stored at the vertex corresponding to $y_1 \ldots y_i$. Whenever the oracle is queried at the point $y_1 \ldots y_n$, A_i replies with $g_{y_{i+1} \ldots y_n}(r_{y_1 \ldots y_i})$.

Let p_n^i denote the probability that T outputs 1 ("accepts") on input 1^n when its oracle queries are answered by A_i, $0 \leq i \leq n$. Since by assumption the ensemble $\{f_s\}$ does not pass the test T, there is a polynomial gap between p_n^0 and p_n^n; more precisely, there exists a polynomial $q(\cdot)$ and infinitely many n such that $|p_n^0 - p_n^n| \geq \frac{1}{q(n)}$. Let K be the set of all n giving rise to such a gap.

We will use T to create an efficient algorithm B_T that distinguishes truly random *strings* from pseudo-random strings generated by g, deriving a contradiction to the assumption that g is a cryptographically strong pseudo-random sequence generator. Let $P_1(n)$ be the polynomial bounding the running time of T on input 1^n. Clearly, T can make at most $P_1(n)$ oracle queries during any execution. On input $n \in K$ and a set W_n of $P_1(n)$ $2n$-bit strings, A_T first picks $i \in_R 0 \ldots n-1$ and then gives to T the input 1^n and answers T's oracle queries as follows. On query $y = y_1 \ldots y_n$,

- if y is the first query with prefix $y_1 \ldots y_i$ then A_T writes the next unused string w in the set W_n as the concatenation of two n-bit strings $w = w_0 w_1$, stores the two pairs $(y_1 \ldots y_i 0, w_0)$ and $(y_1 \ldots y_i 1, w_1)$, and
  - replies $g_{y_{i+2} \ldots y_n}(w_0)$ if $y_{i+1} = 0$
  - replies $g_{y_{i+2} \ldots y_n}(w_1)$ if $y_{i+1} = 1$
- otherwise (if y is not the first query with prefix $y_1 \ldots y_i$) then A_T retrieves the pair $(y_1 \ldots y_{i+1}, v)$ and replies $g_{y_{i+2} \ldots y_n}(v)$.

Note that A_T stores data at depth $i+1$. If W_n consists of strings output by the generator g, then A_T stores *pseudo-random* strings at depth $i+1$; in particulary, since each $w = w_0 w_1$ is the output of the generator g on some random seed and w_0 and w_1 are stored at sibling nodes at depth $i+1$, A_T simulates a computation of T with oracle A_i. On the other hand, if W_n consists of truly random $2n$-bit strings, then A_T stores *truly random* strings at depth $i+1$ and hence simulates a computation of T with oracle A_{i+1}.

The probability that A_T outputs 1 when W_n consists of pseudo-random sequences is given by $\sum_{i=0}^{n-1} \frac{1}{n} p_n^i$, where the probability space is over the choice of i, the inputs to the pseudo-random generator, and the coin flips of T. Similarly, the probability that A_T outputs 1 when W_n consists of truly random sequences is given by $\sum_{i=0}^{n-1} \frac{p_n^{i+1}}{n}$. For $n \in N$ these probabilities differ by at least

$$\frac{1}{n}|p_n^0 - p_n^n| > \frac{1}{n \cdot q(n)}$$

contradicting the assumption that g is a pseudo-random generator. ∎

Remark 2 *As in the case of cryptographically strong pseudo-random sequences, the pseudo-random functions f_s described above are pseudo-random only if the seed s remains secret.*

3 Encryption I: Security

Encryption, or secret message transmission, is *the* cryptographic protocol; one on which many other cryptographic protocols are based. It is therefore crucial to carefully define the security properties afforded by a cryptosystem, and to check that these properties suffice for the more complicated construction.

In *public-key* cryptography a receiver publishes an encryption procedure, called a public key, which is used by senders to securely send information to the receiver. The sender and receiver do not have to meet or exchange information over a physically secure channel at any time, and the receiver does not need any information about the identity of the sender. As we shall see, a trapdoor function does not *by itself* constitute a public-key cryptosystem.

3.1 Probabilistic Public-Key Encryption

The first rigorous complexity-based notion of security appears in [51], where a cryptosystem is said to be *semantically secure* against a passive eavesdropper if anything computable (by an eavesdropper) about the cleartext from the ciphertext is computable without the ciphertext [51]. With this intuition in mind, it is clear that any semantically secure cryptosystem *must* use randomization when encrypting. First, of course, randomization must be used in choosing the (encryption, decryption) key pairs (otherwise on input 1^n everyone would generate the *same* keys). In addition, once the encryption key is fixed, the message encryptions should also be randomized:

- In the public-key case, if the encryption of each message is unique, then given a ciphertext α it is easy to verify whether α is an encryption of given message m: just encrypt m using the public key and check if the resulting ciphertext is equal to α. Since the adversary may have additional information that narrows the possible messages (*e.g.*, the adversary knows that the message will be in {attack, retreat}), this is a serious threat.
- In the private-key case, if the encryption of each message is unique, then given the encryptions of two messages it is possible to determine if the corresponding plaintexts are the same. This is important if the adversary has the chance to legitmately learn the decryption of one of the plaintexts.

Definition 3.1 (Probabilistic Public-Key Cryptosystem) *A probabilistic public-key encryption scheme 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 $\forall b \in \{0,1\}\ \forall r \in \{0,1\}^{p(n)}\quad D_d(E_e(b,r)) = b$.*

This definition is for *bit* encryption; to encrypt longer messages one can concatenate several bit encryptions or use some other method.

Specific implementations of probabilistic encryption appear in [3, 15, 47, 51, 63, 78] and in Section 3.2.2. In particular, such schemes can be constructed from *any* trapdoor permutation.

3.2 Semantic Security Against a Passive Eavesdropper

The first rigorous definition of complexity-based security of encryption against eavesdropping is due to Goldwasser and Micali [51]. We give a non-uniform version appearing in [64]. A *message space enemy* $\mathcal{F} = \{F_n(\cdot)\}$ is a family of polynomial-size probabilistic circuits that on input e produces a description of a pptm $\mathcal{M}$ that on 1^n produces elements of $\{0,1\}^n$ with some probability distribution. Let V be any set and let $\mathcal{G} = \{g_n^e : M_n \longrightarrow V \mid e \in GP(1^n)\}$ be any set of functions. Let p_n^e denote the probability of the most probable value for $g_n^e(m)$:

$$p_n^e = \max\{\sum_{m \in (g_n^e)^{-1}(v)} \Pr_n(m) \mid v \in V\}$$

Definition 3.2 (Semantic Security (non-uniform version)) *An encryption scheme (GP, E, D) is* semantically secure *if for every family of polynomial-size probabilistic circuits $\mathcal{F} = \{F_n(\cdot)\}$ that on input e produces a description of a pptm $\mathcal{M}$, all families of polynomial-size probabilistic circuits $\mathcal{A} = \{A_n\}(\cdot,\cdot,\cdot)$, and all $c > 0$, for all sufficiently large n*

$$\Pr[e \leftarrow GP(1^n);\ \mathcal{M} \leftarrow F_n(e);\ m \leftarrow \mathcal{M}(1^n);\ \alpha \leftarrow E_e(m) \tag{14}$$

$$A_n(e, \mathcal{M}, \alpha) = g_n^e(m)] < p_n^e + \frac{1}{n^c} \tag{15}$$

This is read as: The probability that $A_n(e,\mathcal{M},\alpha) = g_n^e(m) < p_n^e + \frac{1}{n^c}$ after the ordered execution of the statements $e \leftarrow GP(1^n)$, $\mathcal{M} \leftarrow F_n(e)$, and so on.

Intuitively, the definition says that the best way an adversary A_n seeing, e, a description of the message space $\mathcal{M}$, and α, can guess $g_n^e(D_d(\alpha))$, is by outputting the most probable value of $g_n^e(m)$.

Definition 3.2 is very powerful: the function g need not even be computable. Nonetheless, it is equivalent to (the non-uniform version of) *indistinguishability of encryptions* [64]. Intuitively, indistinguishability says that *it is hard to find* a pair of messages on which encryptions can be distinguished.

We present a uniform version of the definition of indistinguishability (there are several – see [42]). In the definition, $\mathcal{F}$ will be a probabilistic polynomial time Turing machine that, on input $(1^n, GP(1^n))$ outputs a pair of messages $\{m_0, m_1\}$.

Definition 3.3 (Indistinguishability of Encryptions) *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 probabilisitic 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.

3.2.1 Encryptions of Zeros and Ones

Many subtleties in the definitions disappear for the special case in which the message space is restricted to $\{0, 1\}$ (or any other fixed pair of messages $\{m_0, m_1\}$). In this case, Definition 3.3 becomes:

Definition 3.4 (Indistinguishability of Encryptions for $\{0, 1\}$) *For all polynomial time machines M, for all $c > 0$, $\exists n_c$ s.t. for $n > n_c$*

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

where the internal probabilities are taken over the coin flips of, M and the choice of r and the external probability is taken over $e \in_R GP(1^n)$.

We say that this formulation of indistinguishability discusses *individual* probabilities (that is, probabilities for what happens with individual keys e).

An alternative definition is:

Definition 3.5 (Aggregate Indistinguishability for $\{0, 1\}$) *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 probabilities are taken over the coin flips of GP, M and the choice of r.

This formulation discusses *aggregate* probabilities (aggregated over the choice of encryption key e).

Clearly, if (essentially all) individual e's give rise to small differences then the aggregated differences will be small. The converse holds as well:

Exercise 3.1 *Prove that Definition 3.4 and Definition 3.5 are equivalent.*

In the case of bit-by-bit encryption, any ability to distinguish encryptions of distinct messages (respectively, in the aggregate sense or for individual keys) immediately translates into an ability to distinguish encryptions of 0 from encryptions of 1 (respectively, in the aggregate sense or for individual keys); the proof is by a hybrid argument.

3.2.2 Construction of an SS-CPA Cryptosystem

The following cryptosystem, based on the difficulty of testing for quadratic residuousity, appears in [51]. The scheme is semantically secure against chosen plaintext attacks under the quadratic residuousity assumption, defined below.

We will need the following:

1. For $n \in \mathbb{N}$, define the predicate $Q_n(x)$ to be true if and only if $\exists r\, r^2 \equiv x \bmod n$. If $Q_n(x)$ holds we say that x is a *quadratic residue modulo n*. Otherwise, we say that x is a quadratic *non-residue*.

2. If n is prime then $Q_n(x)$ is easy to compute. (Exercise: show how to compute $Q_n(x)$ when n is prime. Hint: use Fermat's Little Theorem: $z^{n-1} \equiv 1 \bmod n$ for all z if n is prime.) For n prime, the *Legendre symbol* $(\frac{z}{n})$ is defined by

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

3. Let $N = pq$, where p and q are prime. The *Jacobi* symbol is defined to be

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

   The Jacobi symbol gives the *parity* of the predicates $Q_p(z) \oplus Q_q(z)$; it does not give the logical AND. Since z is a square modulo N iff it is a square modulo all the factors of N, $(\frac{z}{N})$ is in general *not* equal to $Q_N(z)$.

4. For $N = pq$: without knowing p,q it is easy to compute $(\frac{z}{N})$.

5. For $N = pq$ it is easy to compute $Q_N(z)$ if the factors of N are known.

6. The Quadratic Residuousity Assumption, stated below, says that for $N = pq$ it is hard to compute $Q_N(z)$ if the factorization is not known.

The cryptosystem below is a bit-by-bit encryption scheme, where encryptions of 0 and 1 are indistinguishable under the Quadratic Residuousity Assumption. As noted above, if on input $e \in_R GP(1^n)$, a pptm $\mathcal{F}$ on input e can find message pair $\{m_0, m_1\}$ whose encryptions under e are easily distinguished, then by a hybrid argument this can be converted into a distinguisher for encryptions of 0 and 1. Thus, under the QRA, this system is semantically secure under chosen plaintext attack.

Public Key. The public key consists of n, z where n is the product of two large odd primes $n = pq$ and z is a random quadratic non-residue with Jacobi symbol $(\frac{z}{n}) = 1$.

Encryption. Encryption is bit-by-bit. To encrypt 0, choose a random $r \in \mathbb{Z}_n^*$; the ciphertext is $r^2 \bmod n$. To encrypt 0, choose a random $r \in \mathbb{Z}_n^*$; the ciphertext is $r^2 z \bmod n$.

Decryption. Given the factorization of n, check quadratic residuousity by checking to see if the ciphertext is a square modulo both factors of n (this is easy).

Security Assumption Encryptions of 0 and of 1 are ptime indistinguishable under the so-called *quadratic residuousity assumption (QRA)* ([51]):

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

Exercise 3.2 *Prove that z in the public key does not leak information. (First argue that it is easy to generate a "short" list that, with very high probability, contains a random quadratic non-residue with Jacobi symbol 1.)*

Remark 3 *If N is the produce of two primes both congruent to $3 \bmod 4$ (aka "Blum integers"), then we may choose $z = -1$ in the Goldwasser-Micali scheme.*

3.3 Inadequacy of Semantic Security Against Chosen Plaintext Attacks

In a distributed protocol, in which the adversary controls *active* parties in the protocol, semantic security against a passive eavesdropper is frequently an inadequate security guarantee. For example, consider the following simple protocol in which two parties A and B flip a coin with the help of a trusted third party C.

C publishes a public key E.
A flips a coin and publishes $\alpha \in_R E(b_A)$.
B flips a coin and publishes $\beta \in_R E(b_B)$.
C computes $b_A \oplus b_B$ and publishes the result.

In fact, in the last step we can have C release the private key so that both A and B can check that C is telling the truth. Moreover, we can require that $\beta \neq \alpha$.

The *erroneous* intuition is that, since B does not know the value of b_A, B cannot control the outcome of the protocol. However, this is false. If the cryptosystem is *malleable*, then B may be able to completely control the outcome. This is the case with the cryptosystem described in Section 3.2.2. To arrange an outcome of 0, B chooses a random r, and sets $\beta = r^2\alpha \bmod N$. To arrange an outcome of 1, B chooses a random r and sets $\beta = r^2\alpha z \bmod N$.

> *Semantic Security is inadequate because the issue is independence, and secrecy does not imply independence.*

A second example attacks RSA (or any other multiplicative scheme). Consider the problem of *contract bidding.* Assume a setting in which the bids are "sealed" by encryption: there is a public key E and encryptions of bids are announced. Company A places its bid of $\$X$ by publishing $\alpha = E(X)$. If E is multiplicative, then Company B can underbid, for example, by publishing $(E(99)/E(100))\alpha$.

Although the bidding example may seem contrived, a real implementation of the RSA-based encryption in PKCS#1 was recently broken – in the sense that ciphertexts were actually decrypted – by an *active* attack that exploited the malleability of RSA [8]. In this particular real-life attack, there was a server that rejected incorrectly formatted messages. By manipulating the target ciphertext and testing to see if the resulting ciphertext was valid (by sending it to the server and seeing whether or not it would be rejected), the attack broke the target ciphertext by (eventually) learning every bit of plaintext. The *malleability* of RSA under chosen plaintext attack lead to a break of the *semantic security* under a chosen ciphertext attack (never mind the fact that, strictly speaking, RSA is not semantically secure because it is deterministic

– the attack would work against randomized versions of RSA in which the plaintext is padded with a random suffix[1]).

Remark 4 *1. Even if a cryptosystem is non-malleable under chosen plaintext attacks, it could still be vulnerable to this type of attack – if it is malleable under chosen ciphertext attacks.*

2. This attack shows the power of a "valid/invalid ciphertext" oracle.

As a final, more generic example of confusion about the meaning of semantic security, in their celebrated paper on a logic of authentication [19], 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 this reasoning (and the proof of security in [19]). For example, if the key is used to generate a one-time pad (or is used directly as a one-time pad), and the "any function" is $f(N_b) = N_b \oplus 1^{|N_b|}$ (that is, $f(x)$ flips all the bits of x), then impersonating A is trivial: just xor the ciphertext with $1^{|N_b|}$.

Moral. The definition of security of an encryption scheme must specify

1. The attack: What is the power of the adversary? This includes the computational power *and* a description of the adversary's access to the system.
2. The break: What is the definition of a failure?

One must then show, that under a specific computational hardness assumption, no adversary mounting the specified attack (as in 1) can break the system (as defined in 2).

[1]For risks of padding RSA in this fashion, see [25].

3.4 A Taxonomy of Security Definitions

3.4.1 Attacks

The literature considers three broad types of attacks against a cryptosystem:

- Chosen plaintext. This is the weakest form of attack that makes 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 [69], 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*. Informally, the adversary may access a decryption oracle any polynomial (in the security parameter) number of times, asking for decryptions of *adaptively chosen* ciphertexts (some of which may be invalid). Then the oracle is removed and a "challenge" ciphertext is given to the attacker.
- Chosen ciphertext in the sense of [72]; we use the term chosen ciphertext attack in the *post-processing* mode, abbreviated *CCA-post*. This is defined formally in Section 9. The key point is that the attacker sees the challenge ciphertext *before* the oracle is removed, and can ask the oracle to decrypt any polynomialy bounded number of adaptively chosen (possibly invalid) ciphertexts *except the challenge.*

To motivate cca-preprocessing attacks, consider a *public-key identification scheme*, in which the proof of identity is the ability to decrypt challenge messages encrypted in a given public key. Clearly, it is essential that during the attack (adaptively chosen ciphertexts) the adversary should not learn how to decrypt *future* challenges (in the pre-processing mode), say, by learning the decryption key.

Exercise 3.3 *Show that such a scheme is* not *secure against* concurrent *impersonation.*

In Section 9.6.2 we will show how to use this approach to build an *authentication scheme*, using any public-key cryptosystem that is non-malleable against cca-post attacks.

3.4.2 Breaks

The two types of breaks we consider are

- distinguishability of encryptions (equivalently, violation of semantic security)

- malleability

Intuitively, semantic security says:

> Seeing $\alpha \in E(m)$ does not help in finding m' such that m' is related to m.

Non-malleability says:

> Seeing $\alpha \in E(m)$ does not help in finding $E(m')$ such that m' is related to m.

Viewed this way, it is clear that non-malleability is an extension of semantic security. In Section 3.5, we will formalize this definition of semantic security and prove its equivalence to Definition 3.3.

Exercise 3.4 *Prove that any system that is malleable under chosen plaintext attack is semantically insecure under cca-post attacks.*

To explore the distinction among attacks when the notion of a break is malleability, we consider the example of bidding at an auction. The bids are encrypted. Intuitively, bidding and learning the outcome of the auction(s) constitutes a (weak) type of chosen ciphertext attack.

1. If a public encryption key is used for a *single* auction and secrecy is *not* required after the auction is over, then the system should be non-malleable against chosen plaintext attacks.

2. If the key is used repeatedly, for many auctions, and secrecy is not required after each auction is over, then the system should be non-malleable against chosen ciphertext attacks in the pre-processing mode.

3. If the key is used repeatedly and secrect *is* required after the auction is over, then the system should be non-malleable against chosen ciphertext attacks in the post-processing mode.

Case 2 is a little subtle. The adversary's actions in earlier auctions constitute a cca-preprocessing attack on the cryptosystem. Non-malleability against cca-pre should prevent such an attack from helping the adversary to win a later auction. Case 3 is more straightforward. Now, the adversary may in an early auction see a ciphertext $E(m)$ that it wishes to break. Its actions in subsequent auctions constitute some form of cca-post attack on that ciphertext.

Taken together, the three attacks and two breaks yield six separate notions of security for a cryptosystem. It turns out that under the stongest attack (CCA-post), non-malleability and semantic security are equivalent, so at most five of these are distinct. Remark 3.6 of [32] shows that at least four of the possibilities are distinct, and [6] shows that under a slightly different definition of non-malleability all five possibilities are distinct (the collapse of the two notions of break under cca-post holds under the new definition).

3.5 Semantic Security with Respect to Relations

[32] defines the following version of semantic security under chosen plaintext attack. 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 5 *1. In [51] the power of the adversary is separated 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. Accordingly, $\mathcal{A}$ chooses the message space $\mathcal{M}$, on which it will be tested. By letting $\mathcal{A}'$ choose $\mathcal{M}'$ (rather than "inheriting" $\mathcal{M}$ from $\mathcal{A}$), the simulator completely simulates the behavior of the adversary, so in this sense the definition is natural. A second reason for this choice is discussed in Section 9.6.1.*

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 3.6 (Semantic Security With Respect to Relations) *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 $\mathcal{A}$. Including the attack in the description of the adversary is a convention that will be followed throughout the notes.

Note the differences between our definition of semantic security with respect to relations and the definition of semantic security: in semantic security, the challenge is 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, both definitions are equivalent to indistinguishability[2].

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.

Theorem 3.1 *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 hist(α_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 hist(α_1) and $E_e(\alpha_2)$ and produces β_2. Let

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

Note that if 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

[2]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 indinstinguishability [64, 42]. We are using a fourth version of indistinguishability – a uniform version of the non-uniform one-pass version in [64]. In preparing these notes I realized that equivalence of this definition to a corresponding version of semantic security has not been proved in the literature, but I believe it holds.

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 probabilites (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 $\text{hist}(\alpha)$ for an $\alpha \in_R \mathcal{M}$. $\mathcal{A}'$ generates $\alpha' \in_R \mathcal{M}$ and gives to the simulated $\mathcal{A}$ the hint $\text{hist}(\alpha)$ 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 $\text{hist}(x) = 1$ 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 $1/2$). ■

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 [42, 51, 64] that the notions of of (traditional) semantic security, indistinguishability and semantically secure with respect to relations are all equivalent.

3.6 Non-Malleability Against Chosen Plaintext Attack

Here we will define non-malleability against chosen plaintext attacks, discuss simple ideas that do not work, and construct an nm-cpa cryptosystem. Non-malleability against cca-post attacks is discussed in Section 9.

3.6.1 Definition of Non-Malleability Against Chosen Plaintext Attack

We define the power of the chosen plaintext adversary $\mathcal{A}$ for the public-key case (analogous definitions can be made for the secret-key case). Let R be a polynomial time computable relation. Let n be the security parameter. $\mathcal{A}$ receives 1^n and the public key $e \in_R GP(n)$. 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. 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 describing $\mathcal{M}$. 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: 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 a ciphertext, produces a vector of ciphertexts $(f_1, f_2, \ldots)$, where each $f_i \in E_e(\beta_i)$, and a string σ describing $\mathcal{M}'$. 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 $\mathcal{M}'$.

Definition 3.7 *A scheme $\mathcal{S}$ for public-key cryptosystems is non-malleable under chosen* plaintext *attacks 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.*

3.6.2 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 $E(B \circ m + 1)$.

3.6.3 Construction of NM-CPA Cryptosystem

We informally describe a method for constructing a non-malleable cryptosystem under chosen plaintext attack from any cryptosystem that is semantically secure against chosen plaintext attack (and thus, from any trapdoor function). These ideas will be used again in the construction in Section 9.

The main idea is to have many *independent* instances of the underlying (semantically secure) scheme; each encryption of a message will use a *different* subset of the keys; difference is enforced using a one-time signature scheme. If the plaintexts do not all agree then the recipient regards the message as invalid.

We will require a cryptographic hash function h that can hash strings of length any polynomial in n to strings of length n. Such functions will be studied in Section 4.2. For now, assume simply that, given $h(x)$ it is "impossible" to find $y \in h^{-1}(h(x))$.

DDN Lite

Public Key Let (GP, E, D) be a probabilistic public-key cryptosystem that is *semantically* secure against chosen plaintext attack. Let n be the security parameter. The generator $GP(1^n)$ is run $2n$ times to obtain $2n$ pairs (e, d). The public key is the set of $2n$ keys e arranged in k pairs:

$$\begin{matrix} e_1^0 & e_2^0 & \dots & e_n^0 \\ e_1^1 & e_2^1 & \dots & e_n^1 \end{matrix}$$

and a description of the function h.

Private Key The private key is the set of decryption keys d_i^b generated by $GP(1^n)$ together with the e_i^b, $1 \leq i \leq n$, $b \in \{0,1\}$.

Encryption Let S be a *one-time* signature scheme generator. When S is run on input 1^n it produces (public key, private key) pairs (K, L), where L is used for signing messages and K is used for verifying signatures.

To encrypt a message m (of any length):

1. Run $S(1^n)$ to obtain a (public key, private key) pair (K, L). Compute $h(K)$ and write the result as $h(K) = b_1 b_2 \ldots b_n$.
2. For $i = 1, \ldots, n$, let c_i be an encryption of m under key $e_i^{b_i}$.
3. Sign $c_1, \ldots c_n$ using L. Let σ be the result.

The ciphertext is

$$(K, c_1, \ldots, c_n, \sigma)$$

Decryption Verify the signature S on $c_1, \ldots, c_n$ using K. If the signature is valid, verify that all n plaintexts are the same. If not, then the ciphertext is invalid. Otherwise, the decryption is the (common) plaintext value.

The *intuition* for the proof of non-malleability is as follows (this is not rigorous!). Suppose an adversary $\mathcal{A}$ sees a valid ciphertext $(K, c_1, \ldots, c_n, \sigma)$ of a message m. Its goal is to generate a valid ciphertext $(K', c'_1, \ldots, c'_n, \sigma')$ of a related message m'.

The adversary has two choices: to set $K' = K$ or to set $K' \neq K$. If $\mathcal{A}$ chooses to set $K' = K$, then the signature scheme ensures that $\mathcal{A}$, who does not know L, cannot create a valid signature under K for any new message. Thus $c'_1, \ldots, c'_n = c_1, \ldots, c_n$ and $\sigma' = \sigma$. Thus, if $K' = K$ then the only valid ciphertext that the adversary can create is $(K, c_1, \ldots, c_n, \sigma)$ – that is, $\mathcal{A}$ can only produce an exact copy of the target ciphertext. Under the definition of non-malleability this is a failure for $\mathcal{A}$.

If $\mathcal{A}$ chooses to set $K' \neq K$, then since K' is used to select encryption keys from the pairs (e_i^0, e_i^1), for some i, given an encryption $e_i^b(m)$ the adversary must generate an encryption under the other key in the pair $e_i^{\bar{b}}(m')$ of a related message m'. The intuition is that even if the e_i^b are completely malleable, seeing an encryption of m under one key, e_i^b, will not help to generate an encryption of a related m' under a completely different, independently chosen key $e_i^{\bar{b}}$. So the original ciphertext is "no help" in generating an encryption of a related message.

Exercise 3.5 1. *Formalize the intuition above: prove that DDN Lite is non-malleable against chosen plaintext attacks.*

2. *Show that the scheme is* insecure *against cca-post attacks.*

4 Cryptographic Hash Functions

We discuss *collision intractable* and *universal one-way* hash functions. Following [68], we define uowhfs and discuss some of their properties. We then consider constructions based on the subset sum problem ([58]), see that such instances are as hard *on average* as breaking the hardest instances of certain lattice problems [1], and finally observe that the constructions based on subset sum are actually collision intractable.

4.1 Universal One-Way Hash Functions (UOWHFs)

The material in this section is from [68].

Consider an environment in which users share computer programs that reside in the common (read/write) space. We assume that a small read-only area is available in the shared memory. To prevent computer viruses from modifying the programs, the users would like to authenticate the programs before their use. A possible way to do this is to use the read-only area as a common security server which stores the hash value of the files of the common space, where the hash function h itself is publicly known and can be stored in the read-only space as well. The property required from the hash function h is that for a given value x it is computationally hard to find a y such that $h(y) = h(x)$ and $y \neq x$.

We call the hashing of a large file a *public fingerprint*. Notice that this scheme does not require the user to own a private secure space, the compression method is public, and (in the spirit of modern cryptography) private keys are not required. Furthermore, it can provide authentication even to a new or a casual user; this cannot be achieved by previously suggested (private) fingerprinting techniques [77, 70].

The application of some kind "secure" hash function to constructing digital signatures of long messages was recognized since the earlier days of modern cryptography [30]. The property that researchers initially sought was *collision intractability*: given the description of the hash function h it should be hard to find $x \neq y$ such that $h(x) \neq h(y)$. Naor and Yung [68] suggested a relaxation in which first x is chosen arbitrarily by an adversary, then an h is chosen at random and the adversary is challenged to come up with a $y \neq x$ such that $h(x) = h(y)$. This proved to be very useful: such relaxed hash functions can be constructed based on any 1-1 one-way function (trapdoors are not needed!), yet the requirement is sufficiently strong to support applications such as public fingerprints, digital signatures, and non-malleable cryptosystems (see Sections 3.6 and 9).

Definition 4.1 (polynomially related sequences) *Let $\{n_{0_i}\}$, $\{n_{1_i}\}$ be two increasing sequences such that $\forall i\, n_{0_i} \leq n_{1_i}$ and for which there exists a polyonmial $q(\cdot)$ such that $\forall i\, q(n_{0_i}) \geq n_{1_i}$. Then the sequences $\{n_{0_i}\}$ and $\{n_{1_i}\}$ are said to be* polynomially related.

Let $\{n_{0_i}\}$ and $\{n_{1_i}\}$ be polynomially related and let H_k be a collection of functions such that for all $h \in H_k$, $h : \{0,1\}^{n_{1_k}} \mapsto \{0,1\}^{n_{0_k}}$. (Since the sequences $\{n_{0_i}\}$ and $\{n_{1_i}\}$ are polynomially related, the functions are only hashing down "polynomially.") Let $U = \bigcup_k H_k$.

Let $\mathcal{A}$ be a probabilistic polynomial time *collision adversary*, that on input 1^k outputs $x \in \{0,1\}^{n_{1_k}}$ called an *initial value*. $\mathcal{A}$ is then given a description of $h \in_R H_k$. $\mathcal{A}$ then attempts to find $y \neq x \in \{0,1\}^{n_{1_k}}$ such that $h(x) = h(y)$. In other words, after getting the hash function $\mathcal{A}$ tries to find a collision with the initial value (which was chosen by $\mathcal{A}$).

Definition 4.2 (family of universal one-way hash functions) *A family $U = \cup_k H_k$ as described above is a* family of universal one-way hash functions *if for all polynomials $p(\cdot)$ and for all ppt collision adversaries $\mathcal{A}$, for sufficiently large k:*

1. *If the initial value is $x \in \{0,1\}^{n_{1_k}}$, then*

$$\Pr[\mathcal{A}(h,x) = y, h(x) = h(y), y \neq x] < \frac{1}{p(n_{1_k})}$$

   where the probability is taken over the choice of $h \in_R H_k$ and the random choices of $\mathcal{A}$.

2. *$\forall h \in H_k$ there is a description of h of length polynomial in n_{1_k}, such that given x and the description and h, one can compute $h(x)$ in polynomial time.*

3. *H_k is accessible : there exists a probabilisitic polynomial time algorithm G such that $G(1^k)$ generates* uniformly *at random a description of $h \in H_k$.*

H_k is a collection of *descriptions* of functions; two different descriptions may correspond to the same function.

Remark 6 *Recall that the point of non-malleability is to prevent an adversary, having seen a ciphertext $E(m)$ generated by a "good" player, from generating $E(m')$ where m' is somehow related to m. In the use of hash functions in DDN Lite, the hash function h is part of the receiver's public encryption key, and the "starting point" is the public key verification K of the one-time signature scheme chosen by the good player (the preparer of $E(m)$). Since the good player chooses the signature scheme independently of h, this situation is no better for a collision adversary than the one in which the starting point is chosen adversarially and h is chosen at random. Thus, the DDN Lite construction requires only uowhfs, not the full power of collision-itractability. This will also be true of the construction in Section 9.*

The next definition gives us a way of talking about compositions of *collections* of functions.

$$
\begin{array}{lcccc}
 & n_{0_1} & n_{0_2} & n_{0_3} & \cdots \\
U_1 = \cup_k H_{1,k} : & \uparrow & \uparrow & \uparrow & \cdots \\
 & n_{1_1} & n_{1_2} & n_{1_3} & \cdots \\
U_2 = \cup_k H_{2,k} : & \uparrow & \uparrow & \uparrow & \cdots \\
 & n_{2_1} & n_{\ell(2)_2} & n_{2_3} & \cdots \\
\vdots & \vdots & & \vdots & \\
 & \uparrow & & \uparrow & \\
 & n_{(\ell(1)-1)_1} & & n_{\ell(3)_3} & \cdots \\
U_{\ell(1)} = \cup_k H_{\ell(1),k} : & \uparrow & & & \cdots \\
 & n_{\ell(1)_1} & & & \cdots \\
 & & & & \\
U = \cup_k H_k & = H_1 & = H_2 & = H_3 & \cdots
\end{array}
$$

Figure 1: $\forall h \in H_i \quad h : \{0,1\}^{n_{\ell(i)_i}} \mapsto \{0,1\}^{n_{0_i}}$

Definition 4.3 (ℓ-composition) *Let $H_1, H_2, \ldots H_\ell$ be collections of functions such that $\forall i\ \forall h_i \in H_i$, $h_i : \{0,1\}^{n_i} \mapsto \{0,1\}^{n_{i-1}}$, and $n_{i-1} < n_i$. We call the collection $H = \{h | h = h_1 \circ h_2 \circ \ldots \circ h_\ell\}$ an* ℓ-composition *of $H_1, H_2, \ldots, H_\ell$. H is a multiset; if $h_1 \circ h_2 \circ \ldots \circ h_\ell = h'_1 \circ h'_2 \circ \ldots \circ h'_\ell$ for different $(h_1, h_2, \ldots, h_\ell)$ and $(h'_1, h'_2, \ldots, h'_\ell)$, both instances are members of H.*

The ℓ-composition H is itself is a collection of functions, where for all $h \in H$, $h : \{0,1\}^{n_\ell} \mapsto \{0,1\}^{n_0}$. Moreover, if $\exists q(\cdot)$ such that $q(n_0) \geq n_\ell$, then since $\forall i$ $n_i > n_{i-1}$ we have $\ell \leq q(n_0)$, so in this case the composition still hashes polynomially.

The next theorem says that uowhfs compose. The hypothesis of the theorem is illustrated in Figure 4.1. The statement of the theorem is followed by an informal discussion of what the theorem says.

Theorem 4.1 ([68]) *Let $\{n_{0_i}\}, \{n_{1_i}\}, \{n_{2_i}\}, \ldots$ be a sequence of increasing sequences, let $U_1, U_2, \ldots$ be a sequence of families of UOWHF such that $U_i = \bigcup_k H_{i,k}$ where $\forall h \in H_{i,k}$ $h : \{0,1\}^{n_{i_k}} \mapsto \{0,1\}^{n_{(i-1)_k}}$, and $\forall i, k :\ n_{i_k} > n_{(i-1)_k}$. Assume that the U_i's are* simultaneously hard*: for every polynomial p and every probabilistic polynomial time algorithm $\mathcal{A}$, there is a K such that $\forall k > K$ and for all $i \geq 1$, A cannot succeed in finding collisions in $H_{i,k}$ with probability greater than $\frac{1}{p(n_{i_k})}$. Let $\ell : N \mapsto N$ be a polynomial time computable function such that $q(n_{0_k}) > n_{\ell(k)_k}$ for some polynomial q and $n_{\ell(k+1)_{k+1}} > n_{\ell(k)_k}$. Let H_k be the $\ell(k)$-composition of $H_{1,k}, H_{2,k}, \ldots H_{\ell(k),k}$ and let $U = \bigcup_k H_k$. Then U is a family of universal one-way hash functions.*

Discussion of the Theorem For each k, the collection $H_{i,k}$ plays a role in Theorem 4.1 that is analagous to the set H_i in Definition 4.3. In Figure 4.1, each choice of k specifies a column. The function $\ell(k)$ describes which collection of rows in column k

$$
\begin{array}{lccccc}
 & 1 & 2 & 3 & 4 & \dots \\
U_1 = \cup_k H_{1,k} : & \uparrow & \uparrow & \uparrow & \uparrow & \dots \\
 & 2 & 3 & 4 & 5 & \dots \\
U_2 = \cup_k H_{2,k} : & & \uparrow & \uparrow & \dots & \\
 & & 4 & 5 & 6 & \dots \\
U_3 = \cup_k H_{3,k} : & & & \uparrow & \uparrow & \dots \\
 & & & 6 & 7 & \dots \\
 & & & & \uparrow & \dots \\
\dots & & & & 8 & \dots \\
 & & & & & \\
U = \cup_k H_k & = H_1 & = H_2 & = H_3 & = H_4 & \dots
\end{array}
$$

Figure 2: $\forall i : \ell(i) = 2i$; $\forall h \in H_i,\ h : \{0,1\}^{2i} \mapsto \{0,1\}^i$

are being composed to form the collection H_k. Since $n_{\ell(k+1)_{k+1}} > n_{\ell(k)_k}$, the sequence $\{n_{\ell(k)_k}\}$ is increasing. Finally, since for some polynomial $q(\cdot)$, $q(n_0) > q(n_{\ell(k)_k})$, we have that the increasing sequences $\{n_{0_k}\}$ and $\{n_{\ell(k)_k}\}$ are polynomially related.

It follows from Theorem 4.1 that families of universal one-way hash functions compose. Thus, to obtain polynomial shrinking it is sufficient to find a method for shrinking by a single bit. (For example, Figure 2 illustrates the application of Theorem 4.1 to constructing hash functions that halve the length of their inputs from functions that shrink their inputs by a single bit. In this example, $\ell(k) = 2k$.) Naor and Yung gave a method for constructing uowhfs that shrink by a single bit, assuming the existence of any 1-1 one-way function. In particular, neither trapdoors nor permutations are necessary. They then showed how to use uowhfs to construct a digital signature scheme that is *existentially unforgeable against chosen message attacks.* This is the strongest type of security studied in the literature; it says that an adversary, given signatures of any polynomial number of adaptively chosen messages, cannot generate even one valid (message, signature) pair. Thus, strong digital signatures do not require trapdoor functions.

4.2 Leftover Hash Lemma

In the next section we will describe an important family of hash functions based on the subset-sum problem. The (original) proof that these functions yield uowhfs requires the celebrated Leftover Hash Lemma. This version of the lemma appears in [60] and is attributed Rackoff [71]. The original, slightly weaker, version is due to Impagliazzo, Levin, and Luby [57].

Definition 4.4 (Universal) *A collection* $H = \{h_\lambda : \{0,1\}^n \mapsto \{0,1\}^\ell\}_{\lambda \in \Lambda}$ *is* universal *if* $\forall x \neq y,\ x, y \in \{0,1\}^n$,

$$\Pr_\lambda[h_\lambda(x) = h_\lambda(y)] = \frac{1}{2^\ell}$$

Definition 4.5 (Almost-Universal) *A collection* $H = \{h_\lambda : \{0,1\}^n \mapsto \{0,1\}^\ell\}_{\lambda \in \Lambda}$ *is* almost-universal *if* $\forall x \neq y,\ x, y \in \{0,1\}^n$,

$$\Pr_\lambda[h_\lambda(x) = h_\lambda(y)] \leq \frac{1}{2^\ell} + \frac{1}{2^n}$$

Definition 4.6 (Collision Probability) *Let D be a distribution on a finite set S. The* collision probability *of D is the probability that two elements chosen independently according to D are the same.*

Definition 4.7 (statistically close within ε) *Distributions D and D' on a finite set S are said to be* statistically close within ε *if* $\forall X \subset S$,

$$|D(X) - D'(X)| < \varepsilon$$

where $D(X)$ denotes the probability that a randomly chosen element $d \in_R D$ is in X.

Definition 4.8 (ε-quasi random) *Let D be a distribution on a finite set S. Then D is ε-quasi random if D is statistically close within ε from the uniform distribution on S.*

Lemma 4.1 (Leftover Hash Lemma) *Let $X \subset \{0,1\}^n$, $|X| \geq 2^\ell$. Let $e > 0$, and let H be an almost universal collection of functions mapping n bits to $\ell - 2e$ bits. Then the distribution $(h, h(x))$ is quasi-random within $1/2^e$ on the set $H \times \{0,1\}^{\ell-2e}$, where $h \in_R H$ and $x \in_R X$.*

Proof: **Claim 4.1** *The collision probability of the distribution $(h, h(x))$ is at most* $(1 + 2/2^{2e})/(|H|2^{\ell-2e})$.

Let $h_1, h_2 \in_R H$ and let $x_1, x_2 \in_R X$.

$$\begin{aligned}
\mathrm{coll}((h, h(x)) &= \Pr[h_1 = h_2 \text{ and } h_1(x_1) = h_2(x_2)] \\
&= \Pr[h_1(x_1) = h_2(x_2) \mid h_1 = h_2] \cdot \Pr[h_1 = h_2] \\
&= \Pr_h[h(x_1) = h(x_2)] \cdot \Pr[h_1 = h_2] \\
&= \Pr_h[h(x_1) = h(x_2)] \cdot |H|^{-1}
\end{aligned}$$

Now, $\Pr_h[h(x_1) = h(x_2)] =$

$$\begin{aligned}
&\Pr_h[h(x_1) = h(x_2) \mid x_1 = x_2]\Pr[x_1 = x_2] + \Pr_h[h(x_1) = h(x_2) \mid x_1 \neq x_2]\Pr[x_1 \neq x_2] \\
&\leq (1)\Pr[x_1 = x_2] + \Pr_h[h(x_1) = h(x_2) \mid x_1 = x_2](1) \\
&\leq \tfrac{1}{2^\ell} + \left(\tfrac{1}{2^{\ell-2e}} + \tfrac{1}{2^n}\right)
\end{aligned}$$

by the independence of the choice of x_1 and x_2 and the almost-universality of H.

Thus the overall collision probability of the distribution $(h, h(x))$ is bounded above by

$$\left(\frac{1}{|H|}\right)\left(\frac{1}{2^\ell} + \frac{1}{2^{\ell-2e}} + \frac{1}{2^n}\right)$$

The claim now follows from simple algebraic manipulation and the fact that $\ell \leq n$.

Claim 4.2 *Let D be a distribution on a finite set S. If the collision probability of D is at most $(1 + 2\delta^2)/|S|$, then D is quasi-random within δ.*

The claim is proved by contradiction. Suppose that the hypothesis holds but that for some set $X \subset S$,

$$|D(X) - \frac{|X|}{|S|}| \geq \delta$$

Assume without loss of generality that $D(X) = \frac{|X|}{|S|} + \delta$. The collision probability of D is the probability that $d_1 = d_2$ when both are chosen according to D, so the collision probability is the sum of (Prob. of collision when d_1 and d_2 are both in X)(Prob. both in X) and (Prob. of collision when both in $S \setminus X$)(Prob. both in $S \setminus X$). Thus,

$$\begin{aligned} \text{coll}(D) &\geq \frac{1}{|X|} D(X)^2 + \frac{1}{|S| - |X|}(1 - D(X))^2 & (16) \\ &= \frac{1}{|S|} + \frac{\delta^2}{|X|} + \frac{\delta^2}{|S| - |X|} & (17) \end{aligned}$$

This is minimized when $|X| = |S|/2$, where we get a lower bound on the collision probability of $(1 + 4\delta^2)/|S|$, contradicting the hypothesis. Thus, the assumption that D is not quasi-random within δ must be false, and we have obtained the desired contradiction.

From the first claim, we have that the collision probability of $(h, h(x))$ is at most $(1 + 2/2^{2e})/|H|2^{\ell-2e}$. The set S of all pairs of the form $(h, h(x))$ has size $|H|2^{\ell-2e}$. Thus, from the second claim, taking $\delta = 1/2^e$, we get that the distribution $(h, h(x))$ is quasi-random within 2^{-e}. ■

4.3 Hash Functions Based on Subset Sum

We now present a method, due to Impagliazzo and Naor, for constructing families of universal one-way hash functions that are provably as hard to break as solving average-case instances of a class of subset sum problems [58]. The work is interesting for several reasons:

1. The construction is extremely simple, efficient, and parallelizable.

2. *Average-case* instances of these subset sum problems are now known to be provably as hard to break as the *worst-case* instances of a class of problems involving lattices [1].

3. These hash functions are actually *collision-intractable* (see [44]). The average-case/worst-case analysis holds for this property as well.

4. When the parameters are modified (changing the class of subset sum problems), the construction yields a pseudo-random generator that is secure provided average-case instances of the corresponding subset sum problems are hard (no worst-case/average-case equivalence is known for this case).

The results in this section appear in [58] and [1]. See [33] for a survey of these and related results.

Definition 4.9 (Subset Sum Problem) *The subset sum problem of dimensions n and $\ell(n)$ is: given n numbers, $\vec{a} = (a_1, a_2, \ldots, a_n)$, each $\ell(n)$ bits long, and a number T, find a subset $S \subset \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i \equiv T \bmod 2^\ell$.*

Each choice of the function $\ell(n)$ determines a family of problems: invert the following function:

$$f(\vec{a}, S) = \left(\vec{a}, \sum_{i \in S} a_i \bmod 2^{\ell(n)}\right)$$

i.e., invert the function which pairs $\vec{a}$ with the sum of the a_i's for $i \in S$. We usually think of $a_1, a_2, \ldots, a_n$ as a fixed "public" description (like the modulus in RSA), and view f as mapping an n-bit string S to an $\ell(n)$-bit string. In this case we will write $f_{\vec{a}}(S)$ for $f(\vec{a}, S)$. Also, we often identify the subset $S \subset \{1, \ldots, n\}$ with its incidence vector $s \in \{0,1\}^n$ and write $f_{\vec{a}}(s)$.

We say that the subset sum problem for length $\ell(n)$ is *hard* if the corresponding collection of functions is one-way in the sense of Definition 1.4 for the special case in which Algorithm $D(i)$ samples an element *uniformly* from D_i (Definition 1.4 requires the functions to be hard to invert on inputs $f(x)$ where x is drawn according to *any* polynomial-time sampleable distribution; so the kind of hardness required here is a *weaker* requirement than the one in Definition 1.4.) Thus, the goodness of the hash functions constructed here will depend on the difficulty of inverting random instances of the subset sum problem with parameters n and $\ell(n)$ (chosen below): first $\vec{a}$ is chosen, with each $a_i \in_R \{0,1\}^\ell$, then $s \in_R \{0,1\}^n$ is chosen; the random instance is $(\vec{a}, \sum_{i \in s} a_i)$.

The character of the subset sum problem is quite different when $\ell(n) > n$ and when $\ell(n) < n$. If $\ell(n) > n$, then the function is almost 1-1: intuitively, collisions don't even *exist*. In this case the function expands the length of the input, and the functions can be used for pseudo-random generators. (See [58] for details.) If

$\ell(n) < n$, then the function is many-one, almost onto, and contracts the input; these functions will be used for one-way hashing.

Theorem 4.2 *Let $l(n) = (1 - c)n$ for $c > 0$. If the subset sum function for length $l(n)$ is one-way, then it is also a family of universal one-way hash functions.*

Proof: We will abuse notation slightly and sometimes write $f_{\vec{a}}(s)$ to denote $\sum_{i \in s} a_i$.

Suppose, for the sake of contradiction, that there exists a probabilistic polynomial time *collision finding* algorithm $\mathcal{C}$ that selects s_1 (without access to $\vec{a}$) and then, with non negligible probability over choice of $\vec{a}$ and $\mathcal{C}$'s coin flips, succeeds in finding $s_2 \neq s_1$ such that $f_{\vec{a}}(s_1) = f_{\vec{a}}(s_2)$ (that is, in finding a collision). We will show how to use $\mathcal{C}$ to invert random instances of the subset sum problem with non negligible probability.

Given a random instance $((a_1, a_2, \ldots, a_n), T)$ of the subset-sum inversion problem, we construct an input to $\mathcal{C}$ using the following procedure P:

1. Let $\mathcal{C}$ select an $s_1 \in \{0,1\}^n$ (assume for now that s_1 is non-empty; we will return to this point later).
2. Compute $T_{s_1} = \sum_{i \in s_1} a_i = f_{\vec{a}}(s_1)$. Choose a random j such that $j \in s_1$, and define
$$a'_i = \begin{cases} a_i & \text{if } i \neq j \\ a_i - T_{s_1} + T \bmod 2^{\ell(n)} & \text{if } i = j \end{cases}$$
Note that $\mathcal{C}$ has not yet seen $\vec{a}$ or $\vec{a'}$.
3. Give the instance $(\vec{a'}, s_1)$ to $\mathcal{C}$. The algorithm attempts to find $s_2 \neq s_1$ such that $f_{\vec{a'}}(s_2) = f_{\vec{a'}}(s_1)$.

Suppose $\mathcal{C}$ succeeds in finding an s_2 that collides with s_1. If $j \notin s_2$, then s_2 is a solution to the original problem, since swapping a_j and a'_j does not affect the sum over s_2. That is, in this case

$$f_{\vec{a}}(s_2) = f_{\vec{a'}}(s_2) \tag{18}$$

Since by assumption $\mathcal{C}$ has found a collision,

$$\begin{aligned} f_{\vec{a'}}(s_2) &= f_{\vec{a'}}(s_1) \\ &= f_{\vec{a}}(s_1) - T_{s_1} + T \end{aligned} \tag{19}$$

By definition of T_{s_1} and Equations 18 and 19, we have

$$f_{\vec{a}}(s_2) = T \tag{20}$$

Finally, we can assume that $\exists j \in s_1 \backslash s_2$, (that is, $s_1 \not\subset s_2$), since otherwise we could use $\mathcal{C}$ for solving the subset sum problem on $T = 0$ (see Exercise 4.1; this also addresses the case in which $\mathcal{C}$ chooses $s_1 = 0$). Since j was chosen at random, the probability that $j \notin s_2$ is at least $1/n$. (Actually it is closer to $1/2$, however $1/n$ suffices for the proof.) It remains only to show that the probability distribution on the inputs to $\mathcal{C}$ in procedure P is sufficiently close to the distribution on inputs to an adversary in the definition of a family of universal one-way hash functions. (We call this latter distribution the "usual" one seen by $\mathcal{C}$.) If this is the case, then by assumption $\mathcal{C}$ would have a $1/poly$ chance of breaking the subset sum problem, contradicting our assumption that this problem is hard on average.

Exercise 4.1 *Show that for randomly chosen $\vec{a}$, finding s such that $\sum_{i \in s} a_i \equiv 0 \bmod 2^\ell$ is equally hard as inverting random instances of f (where the target is typically non-zero), in the sense that given an oracle for one of these problems there is a polynomial-time bounded oracle machine that solves the other problem with polynomial probability.*

Lemma 4.2 ([58]) *Let $\ell(n) \leq cn$ for $c < 1$. For all but an exponentially small fraction of $\vec{a} = (a_1, a_2, \ldots, a_n)$, the distribution given by $f_{\vec{a}}(S) = (\vec{a}, \sum_{i \in S} a_i)$ for a randomly chosen S is quasi-random within an exponentially small amount.*

To prove Lemma 4.2, first note that we have a collection $H = \{f_{\vec{a}}\}$ of functions where for every $h \in H$, $h : \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$. Let $s_1 \neq s_2 \in \{0,1\}^n$. Without loss of generality, assume $n \in s_2$ and $n \notin s_1$. Let $\vec{a}$ be selected uniformly at random. Then $f_{\vec{a}}(s_1) = f_{\vec{a}}(s_2)$ if and only if

$$a_n \equiv \sum_{i=1}^{n-1} (s_{1_i} - s_{2_i}) a_i \bmod 2^{\ell(n)}$$

Since $a_n \in_R \{0,1\}^{\ell(n)}$, this occurs with probability $2^{-\ell(n)}$. Thus H satisfies the condition of the Leftover Hash Lemma. Writing $\ell(n) = n - 2e$ we have $e = \frac{n-\ell(n)}{2}$, so by the Leftover Hash Lemma the distribution $(h, h(x))$ is $1/2^{\frac{n-\ell(n)}{2}}$-quasi random. Since $n - \ell(n) = n - cn$ is some positive constant times n, this distribution is quasi-random within an exponentially small amount. From this Lemma 4.2 follows easily.

Exercise 4.2 *Complete the proof of Lemma 4.2.*

We use Lemma 4.2 to argue that the distribution on the inputs to $\mathcal{C}$ in Procedure P is essentially the same as the usual input distribution seen by $\mathcal{C}$.

1. $\mathcal{C}$ chooses s_1 exactly the same way in both settings.

2. For all $i \neq j$, a_i is distributed uniformly and independently.

3. Since $(\vec{a}, T)$ is a random instance of the subset sum problem, and in particular, T was chosen by first choosing $s \in_R \{0,1\}^n$ and then setting $T = \sum_{i \in s} a_i$, it follows from Lemma 4.2 that T is distributed quasi-randomly given $a_1, a_2, \ldots, a_n$, so $a'_j = a_j - T_{s_1} + T$ is quasi-random. Thus, the input $(s_1, \vec{a'})$ is polynomially indistinguishable from the input the $\mathcal{C}$ "usually" receives when finding collisions (see Exercise 4.3 below).

This completes the proof of Theorem 4.2. ∎

Exercise 4.3 *Show that the distribution* $(s_1, (a_1, \ldots, a_{j-1}, a_j - T_{s_1}, a_{j+1}, \ldots a_n))$ *is* easily *distinguishable from the usual input to* $\mathcal{C}$*.*

5 An Average-Case / Worst-Case Equivalence for Subset-Sum-Based Hashing

Cryptographic constructions necessarily require random choices: if, for example, the choice of a key were deterministic, then the key could not be secret. Thus, the security of the construction relies on the intractability of a *random* instance of the problem on which the construction is based. It was therefore a longstanding goal in cryptography to find a "hard" problem for which one can establish an explicit connection between the hardness of random instances and the hardness of the hardest, or worst-case, instances.

Such a connection is the contribution of the celebrated paper of Ajtai, "Generating Hard Instances of Lattice Problems" [1]. Specifically, the paper presents a random problem whose solution would imply the solution of three famous worst-case problems:

1. Find the length of a shortest nonzero vector in an n-dimensional lattice approximately, up to a polynomial factor.
2. Find the shortest nonzero vector in an n-dimensional lattice L where the shortest vector v is unique in the sense than any other vector whose length is at most $n^c\|v\|$ is parallel to v, where c is a sufficiently large absolute constant.
3. Find a basis $b_1, \ldots, b_n$ in the n-dimensional lattice L whose length, defined as $\max_{i=1}^{n} \|b_i\|$, is the smallest possible up to a polynomial factor.

In this section we review some basic definitions regarding lattices, describe Ajtai's random problem, and explain the connection to the subset-sum problems of the type used by Impagliazzo and Naor for constructing hash functions.

5.1 Definitions Regarding Lattices

The fundamental concepts concerning lattices can be found in [20, 54, 55].

If $a_1, ..., a_n$ are linearly independent vectors in $\mathbb{R}^n$, then we say that the set $\{\sum_{i=1}^{n} k_i a_i | k_1, ..., k_n \in \mathbb{Z}\}$ is a lattice in $\mathbb{R}^n$. We will denote this lattice by $L(a_1, ..., a_n)$. The set $a_1, ..., a_n$ is called a basis of the lattice; its length is $\max_{1 \leq i \leq n} \|a_i\|$. The determinant of a lattice L will be the absolute value of the determinant of the matrix whose columns are the vectors $a_1, ..., a_n$. We let $\mathrm{bl}(L)$ denote the length of the shortest basis for L.

If $a_1, \ldots, a_n \in \mathcal{R}^\backslash$ are linearly independent vectors, then $\mathcal{P}^-(a_, \ldots, a_n)$ denotes the half-closed parallelepiped $\{\sum_{i=1}^{n} \gamma_i a_i | 0 \leq \gamma_i < 1, i = 1, \ldots, n\}$. By "$x \bmod \mathcal{P}$" we mean the unique vector $x' \in \mathcal{P}^-(a_1, \ldots, a_n)$ so that $x - x'$ is an integer linear combination of the vectors $a_1, \ldots, a_n$.

5.2 Ajtai's Random Lattice Problem

For $n, m, q \in \mathcal{N}$ such that $n \log q < m \leq \frac{q}{2n^4}$ and $q = O(n^c)$ for a fixed $c > 0$, given a matrix $M \in \mathbb{Z}_q^{n \times m}$ (that is, an $n \times m$ matrix of integers in [0,q-1] of a certain form described below), find a vector $x \neq 0 \in \mathbb{Z}_q^m$ so that $Mx \equiv 0 \bmod q$ and $\|x\| < n$. The lattices are defined modulo q, in the sense that if two vectors are congruent modulo q then either both are in the lattice or neither is in the lattice. Thus the matrix M and the integer q define the lattice: $x \in \Lambda(M, q)$ iff $Mx \equiv 0 (\mathrm{mod} q)$.

The matrix M is obtained as follows. Randomize vectors $v_1, \ldots, v_{m-1}$ independently and with uniform distribution on the set of all vectors $\langle x_1, \ldots, x_n \rangle \in \mathbb{Z}_q^n$. Independently randomize a $0, 1$ sequence $\delta_1, \ldots, \delta_{m-1}$, where the numbers δ_i are chosen independently and uniformly. Then define $v_m = -\sum_{i=1}^{m-1} \delta_i v_i \bmod q$ with the additional constraint that each component of v_m is an integer in $[0, q-1]$. The matrix M has columns $v_1, \ldots, v_m$. The class of lattices $\Lambda(M, q)$ defined by matrices of this type will be called λ. The random problem, which we call $\mathcal{R}$, is to find a vector in $\Lambda(M, q)$ of length less than n. Note that $(\delta_1, \ldots, \delta_{m-1}, 1) \in \Lambda(M, q)$ and its length is $O(\sqrt{m})$, so this vector is a solution when $m < n^2$.

Ajtai proved that an efficient solution to *random* instances of $\mathcal{R}$ implies efficient solutios for the following three famous worst-case problems:

1. Find the length of a shortest nonzero vector in an n-dimensional lattice approximately, up to a polynomial factor.

2. Find the shortest nonzero vector in an n-dimensional lattice L where the shortest vector v is unique in the sense than any other vector whose length is at most $n^c \|v\|$ is parallel to v, where c is a sufficiently large absolute constant.

3. Find a basis $b_1, \ldots, b_n$ in the n-dimensional lattice L whose length, defined as $\max_{i=1}^n \|b_i\|$, is the smallest possible up to a polynomial factor.

Let L be an n-dimensional lattice, let $a_1, \ldots, a_n$ be a set of linearly independent vectors in L and let $M = \max_{i=1}^n \|a_i\|$. The heart of Ajtai's work is a procedure which, if $M > n^c \mathrm{bl}(L)$ for a fixed consant c, uses an oracle for the random lattice problem just defined to obtain another set of linearly independent elements in L whose maximum length is at most $\frac{1}{2} \max_{1 \le i \le n} \|a_i\|$.

In rough outline the procedure works as follows. Starting from $a_1, \ldots, a_n$, construct a set of linearly independent lattice vectors $f_1, \ldots, f_n$ such that $\max_{i=1}^n \|f_i\| \le n^3 M$ and $W = \mathcal{P}(f_1, \ldots, f_n)$ is close to a cube, in the sense that each vertex of W will be at most distance nM from a fixed cube. If the space is covered with the cells of a lattice determined by a short basis, then most of the cells intersecting W lie completely in the interior of W. This implies that every parallelepiped of the form $u + W$, $u \in \mathcal{R}^\backslash$, has roughly the same number of lattice points. Moreover, this also holds for parallelepipeds of the form $u + \frac{1}{q} W$ for $q = [n^{c_2}]$, where c_2 is sufficiently small with respect to c. Thus, if we pick a lattice point v at random from a set D of parallelepipeds of the form $u + \frac{1}{q} W$ with non-overlapping interiors, then the distribution induced on D – that is, the choice of which element in D contains v – is very close to the uniform distribution.

The set D of parallelepipeds $u + \frac{1}{q} W$ that is of interest to us is that obtained by cutting W into q^n small parallelepipeds by dividing each of the vectors f_i into q pieces of equal length. Thus each of the small parallelepipeds is of the form $(\sum_{i=1}^n t_i \frac{f_i}{q}) + \frac{1}{q} W$, where $0 \le t_i < q$, $i = 1, \ldots, n$ is a sequence of integers; that is, $\langle t_1, \ldots t_n \rangle \in \mathbb{Z}_q^n$. Let us call the vector $o = \sum_{i=1}^n t_i \frac{f_i}{q}$ the *origin* of the parallelepiped. We will name an element of D by the vector $t(o) = \langle t_1, \ldots, t_n \rangle$ of coefficients of the $\frac{f_i}{q}$ defining its origin. If we choose a random set of lattice points $\xi_1 \ldots, \xi_m$ in W and look at, for each ξ_j, the name $t(o_j)$ of the parallelepiped containing ξ_j, then we get a sequence $t(o_1), \ldots, t(o_m)$ of elements chosen almost uniformly from D. Express each ξ_j as the sum of the origin o_j and an offset $\delta_j \in \frac{1}{q} W$. Note that the offset is relatively short: since δ_j is contained in $\frac{1}{q} W$, $\|\delta_j\|$ is bounded by n times the length of the longest side of W. That is, $\max_{1 \le j \le m} \|\delta_j\| \le n(\frac{1}{q} n^3 M)$.

By definition of D and the fact that the distribution induced on D by the choice of ξ is almost uniform, each $t(o_j)$ is distributed almost uniformly in $\mathbb{Z}_q^n$. Let $m = [c_1 n \log n]$. Consider the sequence $t(o_1), \ldots, t(o_m)$ as a value of the random variable λ (it is shown in [1] that the distribution of $t(o_1), \ldots, t(o_m)$ is extremely close to that of λ). If there exists an algorithm $\mathcal{A}$ that can solve Ajtai's random lattice problem, then using $\mathcal{A}$ we can find a short (length at most n) vector $h = \langle h_1, \ldots, h_m \rangle \in \mathbb{Z}^m$ satisfying $\sum_{j=1}^m h_j t(o_j) \equiv 0 \bmod q$.

Writing the lattice vector $\sum_{j=1}^m h_j \xi_j$ as the weighted sum of origins and offsets, we get

$$w = \sum_{j=1}^{m} h_j \xi_j = \sum_{j=1}^{m} h_j o_j + \sum_{j=1}^{m} h_j \delta_j \ .$$

Critically, since $\sum_j h_j t(o_j) \equiv 0 \bmod q$, we have that $\sum_j h_j o_j$ is an integer linear combination of the vectors $(f_1, \ldots, f_n)$. Since the f_i are lattice vectors, so is $\sum_j h_j o_j$. Since w is also in L the difference $w - \sum_j h_j o_j = \sum_j h_j \delta_j \in L$. Finally, since $|\sum_{j=1}^{m} h_j^2| \leq n^2$ and, as noted above, each of the offsets is also relatively short, the lattice vector $\sum_{1 \leq j \leq n} h_j \delta_j$ is relatively short: $\| \sum_{1 \leq j \leq n} h_j \delta_j \| \leq n^2 (n^4 M \frac{1}{q})$, which is less than $\frac{M}{2}$ if q is sufficiently large (say, $q \geq n^7$).

5.3 Connection to Hashing

We can represent the hash function $f_{\vec{a}}(x) = \sum_{i=1}^{m} x_i a_i \bmod 2^{\ell(m)}$. as a $1 \times m$ matrix M with columns $a_1, \ldots, a_m$. Given $x \in \{0,1\}^m$, the value of the function is $Mx \bmod 2^{\ell(m)}$.

As we next explain, Ajtai's proof shows that the ability to solve a random instance of the subset sum problem implies the ability to solve the worst-case lattice problems listed in Section 5.2 (additional details appear in [1]). So if we assume that these worst-case problems are hard for dimension n, then these randomized subset sum problems will be hard as well. To illustrate this connection, let $q = [n^{c_2}]$ and $m = [c_1 n \log n]$ as in the discussion of Ajtai's reduction in Section 5.2. Let $N = q_1 q_2 \ldots q_n$ where each q_i is a distinct prime in $[q, 2q]$. Let $a_1, \ldots, a_m, b$ be random integers modulo N. Consider the subset sum problem of finding $x \in \{0,1\}^m$ such that $\sum_{i=1}^{m} x_i a_i \equiv b \bmod N$.

Remarks.

(1) The numbers a_i are of length $l(m) \approx n \log q = (1-c)m$ for some $c > 0$ if $c_1 > c_2$. So subset sum problems of this type are essentially those in the Impagliazzo-Naor family of hash functions.

(2) If $x \in \{0,1\}^m$ then $\|x\| \leq \sqrt{m} < n$.

We may express each a_i as a vector of remainders modulo the primes $q_1, \ldots, q_n$: $a_i' = (a_1^i, \ldots, a_n^i)$, where $a_j^i \in \mathbb{Z}_{q_j}$, for $1 \leq i \leq m$ and $1 \leq j \leq n$. Note that if a_i is chosen uniformly from $\mathbb{Z}_N$ then a_i' is implicitly chosen uniformly from $\mathbb{Z}_{q_1} \times \ldots \times \mathbb{Z}_{q_n}$. Similarly, let b' be the Chinese remainder decomposition of b. Let M be the $n \times m$ matrix with columns $a_1', \ldots, a_n'$. If we can find $x \in \{0,1\}^m$ satisfying $\sum_{i=1}^{m} x_i a_i \equiv b \bmod N$, then $Mx \equiv b'$ (where the jth component of the product is reduced modulo q_j, $1 \leq j \leq n$).

The hardness of this problem follows from Ajtai's proof. The key modification is as follows. Recall that $W = \mathcal{P}(f_1, \ldots, f_n)$. Rather than cutting each vector f_i, $1 \leq i \leq n$, into q equal pieces (for a fixed q), instead for each $1 \leq i \leq n$, cut f_i into q_i

pieces. Thus, instead of having q^n little parallelepipeds we will have $N = q_1 \dots q_n$ of them. Any solution x plays the role of the solution $h = \langle h_1, \dots, h_m \rangle$ in the original proof. See [1] for more details and extensions of these results.

6 String Commitment

A string commitment protocol is a pair of two-party protocols, (COMMIT, REVEAL). We think of the committer as a *sender* and the party to whom the commitment is made as the *receiver*. At a high level, during the COMMIT protocol, the sender and receiver exchange messages, creating a conversation that *binds* the sender to a particular value, say, v, without revealing anything about v to the receiver (that is, the committed values enjoy some type of secrecy). Conceptually, the sender places the value v in an envelope, which is then sealed; the sender can at no point substitute a value $v' \neq v$ for v in the envelope – hence the sender is "committed". During the REVEAL protocol, the sender "opens" the envelope. At this point the sender reveals v and proves that v was in fact the committed value. Thus, the sender should be unable to produce any value other than v during the REVEAL protocol. Henceforth, when we speak of a commitment protocol, we assume that if both parties follow the protocol correctly then at the end of the REVEAL phase the receiver will accept the opened value with overwhelming probability.

Of course, like encryption, string commitment is not performed in a vacuum. If A is committing to a string σ to party B, and B is committing to τ to party C, the mere *secrecy* of σ does not ensure *independence* from τ (so much for the envelope analogy). A *non-malleable commitment scheme* enforces this independence.

In this section we will discuss two definitions of security for (malleable) string commitment, and we will present several constructions. We will then define and construct non-malleable string commitment protocols.

6.1 Two Varieties of (Malleable) 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 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. For this reason, these are also called *strong-sender* commitments. In information theoretic commitment it is possible to open the commitment in many ways, but the assumed computational boundedness of the sender prevents him from finding a second way. Such a scheme is designed to be secure against an arbitrarily powerful receiver and a probabilistic polynomial time prover, and hence is sometimes referred to as *strong-receiver* commitment.

Exercise 6.1 *Show that if both parties are arbitrarily powerful then commitment is impossible.*

Strong-Sender Commitment In strong-sender commitment, the committed values enjoy semantic security. We specify what this means, using the notions of security with respect to relations (as in Section 3.5, it is equivalent to the traditional notion of semantic security). We will define two probabilities. For the first probability, 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 $\mathrm{hist}(\alpha)$, where hist is a probabilistic polynomial time computable function. The commitment protocol is executed where the sender S follows the protocol, committing to α, and the receiver R 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 (S, R) execution of the string commitment protocol. $\mathcal{A}'$ chooses a distribution $\mathcal{M}'$. An $\alpha \in_R \mathcal{M}'$ and $\mathrm{hist}(\alpha)$ 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 6.1 *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 relation $R(\alpha, \beta)$ and function hist(m) computable in probabilistic polynomial time $|\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)|$ is subpolynomial.*

Remark 7 *Note that this definition corresponds closely to the definition of semantic security with respect to relations of a cryptosystem under chosen plaintext attack.*

Since this definition is for an arbitrarily powerful sender, the *binding* must be unconditional, but probabilistic: with high probability over R's coin-flips, following the commit stage there is at most one string that R accepts (as the value committed) in the REVEAL stage. Thus, for example, during the course of the COMMIT phase, R may run a test to ensure that there is a well-define committed value, and if R is (exponentially) unlucky then it may falsely be convinced that this is the case.

Strong-Receiver Commitment In strong-receiver commitment, the conversation during the COMMIT protocol must yield no information about the committed value even information-theoretically, so secrecy is perfect. Assume the sender is non-faulty. The requirement is: for every possible committing conversation c, for every possible pair of values v, v' in the domain of the commitment scheme, the probability that conversation c occurs given that the sender's value is v, is exactly equal to the probability that c occurs given that the sender's value is v'.

Clearly, in this case there is more than one possible way to open a commitment. The binding requirement then says that the sender can find at most one way of opening the commitment. We now formalize this requirement.

If c is a conversation between a (possibly faulty) sender S^* and receiver R during the COMMIT stage, we let $\text{VIEW}(R, c)$ and $\text{VIEW}(S^*, c)$ be, respectively, the receiver's and sender's resulting views. We assume that a party's view includes the local coin flips used in the execution of the COMMIT protocol.

We now describe the binding requirement. Let O^* be an interactive probabilistic polynomial-time machine. Intuitively, O^* is trying to "cheat" during the opening of the commitment, while the faulty sender S^* was trying to lay the groundwork during the execution of COMMIT to make such cheating possible. The value σ is *feasible* with respect to O^* under the joint view $(\text{VIEW}(S^*, c), \text{VIEW}(R, c))$, if the probability that $R(\text{VIEW}(R, c))$ after interacting with $O^*(\text{VIEW}(S^*, c))$ in the REVEAL phase, accepts σ is non-negligible. The probability is taken over flips of O^* and R during the execution of REVEAL. The binding requirement is that for all probabilistic polynomial time interactive machines S^* and O^*, with overwhelming probability over the coin tosses of S^* and R during the COMMIT phase, there will be at most one feasible value σ with respect to O^* under the resulting joint view.

6.2 Strong-Sender Commitment

Creating a strong-sender commitment protocol from a one-way permuation is left as an exercise. In this section we describe Naor's bit commitment scheme [65], which requires only a pseudo-random generator (which, in turn, can be constructed from any 1-way function [56]).

Exercise 6.2 *Construct a bit-commitment scheme based on any one-way permutation.*

Let G be a pseudo-random generator that maps an n-bit seed into a $3n$-bit long pseudo-random sequence, and let $[G(s)]_i$ denote the ith bit produced by G on input s. Naor's protocol works as follows.

- The COMMIT protocol:
    1. The receiver chooses a random $3n$-bit string r and sends r to the sender.

2. To commit to a bit B, the sender chooses an n-bit seed s, and sends to the receiver the $3n$-bit long string $d = d_1 d_2 \ldots d_{3n}$, where

$$d_i = \begin{cases} [G(s)]_i & \text{if } r_i = 0 \\ [G(s)]_i \oplus B & \text{if } r_i = 1 \end{cases}$$

- The REVEAL protocol: The sender sends s and B; the receiver computes $G(s)$ and verifies that d is consistent with these values.

Exercise 6.3 *Verify that at the end of the* COMMIT *stage the receiver has at most a subpolynomial advantage in guessing B.*

We now argue that the protocol satisfies the binding requirement for strong-sender string commitment. A pair of strings s_1 and s_2 *fools* a $3n$-bit string r if

1. For all i such that $r_i = 0$: $[G(s_1)]_i = [G(s_2)]_i$.
2. For all i such that $r_i = 1$: $[G(s_1)]_i \neq [G(s_2)]_i$.

It follows immediately from the definition that the pair (s_1, s_2) fools r if and only if $r = G(s_1) \oplus G(s_2)$. Thus, each pair of seeds can fool at most one string r.

Claim 6.1 *The probability, over choice of r, that there exists a pair of seeds s_1 and s_2 that fools r is at most 2^{-n}.*

Proof: As noted above, each pair of seeds can fool only one string r. There are 2^{2n} pairs of seeds, and there are 2^{3n} choices for r, so the probability over choice of r that there exists a fooling pair for r is at most $2^{2n}/2^{3n} = 2^{-n}$. ■

Naor extends this result to obtain a string commitment protocol with *amortized* communication complexity of $O(1)$ bits (see [65]). A different amortization was suggested by Kilian: To commit to a string $\sigma = \sigma_1\sigma_2 \ldots \sigma_m$, first choose an m-bit seed s, commit to each bit of s, say, using Naor's scheme, and then send $s \oplus \sigma$. However, if the security parameter is n, then the amortization becomes effective only when $m \geq n^2$.

6.3 Strong-Receiver Commitment

The following strong-receiver string commitment scheme is due to Chaum, van Heijst, and Pfitzmann [23].

- The COMMIT phase:

1. Receiver sends to Sender a large prime p and g_1, g_2, where g_1 and g_2 both generate the same q-sized subgroup of $\mathbb{Z}_p^*$, where q is a large prime dividing $p-1$.
2. To commit to a string $x \in \mathbb{Z}_q$, choose $r \in_R \mathbb{Z}_q$ and send $g_1^x g_2^r \bmod p$.

- The REVEAL phase: Sender sends x and r to Recevier; Receiver verifies consistency with the informormation received during the COMMIT phase.

Note that if the committer knows s such that $g_2 \equiv g_1^s \bmod p$ then the "commitment" can be opened arbitrarily, so the security of the commitment relies on the hardness of finding discrete logarithms modulo p.

Exercise 6.4 *([58]) Construct a strong-receiver bit commitment scheme based on the subset sum problem.*

Open Question 6.1 *The* selective decommitment *problem first arose in a slightly different form,* selective decryption, *in the context of distributed computing. In the distributed setting, processors use encryption to hide information from an adversary. If the adversary is dynamic, then the adversary can choose which processors to make faulty – and therefore which of a collection of ciphertexts to "open" – as a function of the ciphertexts that it sees. The adversary may therefore see the decryptions of selected messages, hence the term* selective decryption. *The problem is to prove security of the unopened ciphertexts. While each unselected ciphertext individually remains secret, it is theoretically possible for the adversary to compute some unexpected function of the vector of all the plaintexts. The* selective decommitment *problem is similar, except that instead of seeing encryptions of messages the adversary is given* commitments *to the messages. This problem is poorly understood even when the commitments leak no information about the plaintext values* information-theoretically *(the strong-receiver case). Some positive results for the latter case were recently obtained in [34].*

7 Zero Knowledge Interactive Proofs

Interactive proof systems were introduced independently by Babai [4] and Goldwasser, Micali, and Rackoff [52]. Babai and Moran express it nicely [5]:

> One can view NP as the complexity class that captures the notion of efficient provability by classical (formal) proofs. We consider broader complexity classes ... in the hope to formalize the notion of efficient provability by *overwhelming statistical evidence.* Such a concept should combine the nondeterministic nature of (classical) proofs and the statistical nature of conculsions via Monte Carlo algorithms such as a Solovay-Strassen style "proof" of primality.

In an interactive proof system for membership in a language L, there are two parties, the *prover* and the *verifier*. They share a common input x, which the prover wishes to convince the verifier is in L. In addition, each party may have other inputs (modelling other information gathered during the lifetime of the system). The proof system should be complete and sound: if $x \in L$ then the prover should with overwhelming probability be able to convince the verifier to accept; if $x \notin L$ then the prover should have only negligible probability of convincing the verifier to accept. The probability space depends on the particular model being studied.

In Babai's formulation (Arthur-Merlin games), the prover (Merlin) has arbitrary computational power, and the verifier (Arthur) is a polynomial-time machine that can flip coins. Arthur's coins are *public*: Merlin can see the outcome. In the Goldwasser, Micali, and Rackoff formulation (IP), the respective computational power of the two parties is as in AM, but the verifier's coins may be private. In both cases, the probability space is over the coin flips of the two parties. Goldwasser and Sipser showed that the set of languages provable in the two types of interactive systems is the same. There is a rich literature on the subject; however, the case of arbitrarily powerful provers is not of interest in this course, where *all* parties are assumed to be polynomial-time bounded. We mention only one result, due to Shamir: IP = PSPACE [75].

In the second part of their original paper on interactive proofs, Goldwasser, Micali, and Rackoff introduced the concept of knowledge complexity and, in particular, *zero-knowledge* proofs of language membership. These are proofs that "communicate nothing but their validity." This intuitive notion has many possible formulations. The one in [52] is based on the concept of *simulation*. Fix an input $x \in L$. Since both the prover and the verifier may flip coins, x gives rise to a probability distribution on (real) conversations between the prover and the verifier. Suppose there exists a polynomial-time bounded simulator S that, on input x, can generate simulated conversations that "look" like real conversations to any polynomial-time bounded observer: that is, the distribution on (simulated) conversations generated by $S(x)$ is polynomial-time indistinguishable from the distribution on real conversations. Then intuitively conversations between the prover and *this* verifier "leak" no information since, in some sense, the verifier "could have produced" (something indistinguishable from) these conversations herself. The goal of zero-knowledge proof systems is to ensure that no information is leaked to *any* (possibly "cheating") probabilistic polynomial time verifier.

We give two definitions of zero-knowledge. The first is stronger than the definition in [52]; however all known proofs of zero-knowledge satisfy this stronger definition[3]. The notation $(P,V)(x,y,z)$ denotes the situation in which the prover and verifier have common input x, while y and z are, respectively, the private inputs of the prover and the verifier. In the next two definitions, T, S, V are probabilistic polynomial time machines.

[3] There is one anomalous case, but it requires extremely non-standard assumptions.

Definition 7.1 (Black Box Zero Knowledge (Membership in L)) $\exists S \, \forall V \, \forall x \in L \, \forall y, z \, \forall T$,

$$|\Pr[T((P,V)(x,y,z))] - \Pr[T(S^V(x,z))]| < \nu(n)$$

where the probabilities are over the coin flips of T, S, P, and V (as appropriate).

In the next definition the simulator can depend on V. This is essentially the [52] definition, with the *history* variable (z) whose necessity was shown in [50].

Definition 7.2 ("Ordinary" Zero-Knowledge (Membership in L)) $\forall V \, \exists S \, \forall x \in L \, \forall y, z \, \forall T$

$$|\Pr[T((P,V)(x,y,z))] - \Pr[T(S(x,z))]| < \nu(n)$$

where the probabilities are over the coin flips of T, S, P, and V (as appropriate).

7.1 NP $\in$ ZK

Impagliazzo and Yung showed that every language that has an interactive proof has a zero knowledge interactive proof; together with Shamir's result that IP = PSPACE this says that ZKIP = PSPACE (under suitable assumptions) [59]. We will prove an easier result: under the assumption that 1-way functions exist, every language in NP has a zero-knowledge interactive proof [4].

The proof in [49] uses the NP-completeness of graph 3-colorability. We will describe a proof, due to Manuel Blum, based on Hamiltonicity [9]. To prove a general NP statement one must first translate it into an instance of the Hamiltonian Cycle problem, using the publicly known Karp reduction. In the interactive proofs given here and in [49], the prover may be probabilistic polynomial time bounded, provided it has a witness (3-coloring or cycle). A system in which the prover is probabilistic polynomial time bounded is called an *argument*.

Definition 7.3 (Proper Matrix, Proper Permutation Matrix) *A* proper permuation matrix *of size n is an $n \times n$ binary matrix with exactly n ones, such that every row and every column contains a unique one and the $n \times n$ permutation matrix induced by the rows and columns containing 1's is Hamiltonian. An n-*proper matrix *(of any size) is a binary matrix with exactly n ones, such that no row or column contains more than one 1 and the $n \times n$ permutation matrix induced by the rows and columns containing 1's is Hamiltonian.*

A graph G on n nodes contains a Hamiltonian cycle if and only if its adjacency matrix contains a proper permutation matrix of size n.

[4]Goldreich, Micali, and Wigderson were the first to show NP $\in$ ZK; they assumed existence of semantically secure probabilistic encryption [49]; independently but later, related results were obtained by Brassard and Crepeau under the quadratic residuousity assumption [17], and then again under the assumption that factoring is hard [18]. See also [21, 22].

Theorem 7.1 *There is a zero knowledge interactive argument for the language of Hamiltonian Graphs.*

If G is an n-vertex graph with adjacency matrix $A(G)$, then a cycle w is a proper permutation matrix M_w contained in $A(G)$. The protocol $(P, V)(G, w)$, where G is the common input and w is the witness known to P, consists of k sequential repetitions of the following *basic block*:

1. P chooses a random permutation π on the vertices of G and commits to the adjacency matrix of the permuted graph $A(\pi(G))$, say, using Naor's bit commitment scheme for each bit.

2. V sends $b \in_R \{0, 1\}$ to P.

3. If $b = 0$ then P sends to V a description of π, and reveals all the entries in $A(\pi(G))$.

   If $b = 1$ then P reveals the entries the entries in $A(\pi(G))$ corresponding to $\pi(M_w)$.

4. If $b = 0$ then V checks that the commitment received in Step 1 was indeed a commitment to an adjacency matrix of $\pi(G)$.

   If $b = 1$ then V checks that the revealed values are all 1's and that they constitute a proper permutation matrix on n elements (where n is the number of vertices in G).

   If either check fails, then V rejects G.

Completeness is obvious: if both parties follow the protocol correctly and G is indeed Hamiltonian, then the verifier will accept G. Soundness is argued by showing that, if G is not Hamiltonian, then in any execution of the basic block, V will reject G with probability at least $1/2$. The proof uses the fact that if V were to receive the answers to *both* possible questions, then V would be able to construct a Hamiltonian cycle. Zero-knowledge is argued by describing a simulator $S = S(V)$ that on input G constructs simulated conversations that are polyomial-time indistinguishable from real conversations. Since S does *not* have access to a witness of G's Hamiltonicity, these simulated conversations cannot possibly leak any information about the witness. Since these conversations are p-time indistinguishable from real conversations, the real conversations leak no information about the witness.

We briefly sketch the simulator. The proof that the simulated conversations are polynomial time indistinguishable from true conversations is left as an exercise. Assume the real protocol runs for k iterations of the basic block. Here, k is a second security parameter; there is also the security parameter used in the commitment scheme. These are frequently taken to be equal. In this simulation we are ignoring any private "auxillary" inputs to the verifier. The simulation works even with such additional inputs.

Let V^* be a (possibly cheating) verifier. We describe the simulation of the first execution of the basic block. Successive executions are simulated analagously. We assume the simulator knows V^*'s program and has access to V^*'s initial state call it v_0 (remember that the point of the simulation is to show that V^* could have "produced" the conversation herself, so assuming that S has access to any information to which V^* has access is reasonable).

1. S flips a coin c. Intuitively, c represents S's guess of the value of b that V^* will send. V^* is *supposed* to be choosing b uniformly from $\{0, 1\}$; however, the intuition is that regardless of how V^* actually chooses b the probability that $b = c$ will be subpolynomially close to $1/2$.

   - If $c = 0$ (that is, S guesses that b will equal 0), then S prepares to answer the query 0: S chooses a valid permutation π and correctly commits to the adjacency matrix $A(\pi(G))$.
   - If $c = 1$, then S prepares something that will be polynomial time indistinguishable from an answer to the query 1: S chooses a random permuation matrix M and commits to the adjacency matrix of M (note that G plays no role here!).

   Let α be the string prepared by S. S simulates V^* when started in state v_0 with G on the common input tape and α on the communication tape.

2. If V^* does not respond with a bit b then S terminates the simulation and outputs that part of the conversation (including V^*'s response) produced so far. Assuming V^* has responded with a bit b:

   - If $b = c$ then S reveals the required fields of the adjacency matrix. Let γ be the string containing the REVEAL messages. S outputs $\alpha b \gamma$ as the simulated execution of the basic block. S also simulates V^* on γ to obtain the starting state for the next execution of the basic block.
   - If $b \neq c$, then S starts again from Step 1.

Exercise 7.1 *Prove that for all Hamiltonian graphs G, the simulator described above produces (simulated) conversations that are polynomial-time indistinguishable from real conversations produced by $(P, V^*)(G, w)$.*

Open Question 7.1 *Is zero-knowledge (with auxiliary inputs) closed under parallel composition? That is, suppose n copies of the basic block are run* synchronously in parallel, *yielding a "generic" 3-round protocol. Is the 3-round protocol zero-knowledge?*

It is known that L has a 3-round black-box *zero-knowledge protocol if and only if $L \in$ BPP [46], so any positive result must exploit the fact that the simulator may depend on the (possibly cheating) verifier's finite control. (An analgous similar result holds for 4-round protocols that may be executed asynchronously [62].) No positive result is known even for substantial weakenings of the definition of zero-knowledge (see, e.g., [34]).*

[34] elucidates a strong connection between the selective decommitment problem and the question of 3-round (weak) zero-knowledge. Intuitively, if there is a commitment scheme that is sufficiently secure against selective decommitment, then no information is leaked during execution of the generic 3-round protocol (under the right definitions). If the definition of security against selective decommitment is expressed in terms of simulation, then implementing the generic 3-round protocol with a commitment scheme secure in this sense yields a simulatable interactive proof.

7.2 NP $\in$ ZK(5)

A very simple construction yields a 5-round zero knowledge interactive proof (or agument) system for Hamiltonicity. The verifier uses private coins. [46] have shown that no language outside of BPP has a constant-round *black-box* public-coin zero knowledge interactive proof.

1. The prover sends to the verifier the description of a strong-receiver commitment scheme K.

2. The verifier uses K to commit to a query vector of n random bits.

3. The prover chooses n permutations and sends commitments to the adjacency matrices of the n different permutations of the input graph.

4. The verifier reveals its query vector.

5. The prover replies to the queries.

The simulator works as follows. In the third round, the prover commits to random matrices. The verifier will open its commitment with essentially the same probability as it does in real executions. Once the query vector is known we rewind the verifier, give it commitments to values that will yield appropriate replies to the (now known) queries, and then, once the query vector is (again) revealed, answer the queries. If the query vector is not revealed on the second try, then repeat the commitments to the correct answers as often as necessary until the verifier again opens its commitment vector. Piece together the simulated conversation.

Exercise 7.2 *Rigorously describe a simulation procedure for the 5-round protocol and prove that the zero knowledge condition holds.*

8 Zero Knowledge II: The {F,F,S} Results and NIZKs

Most of the results of this section appear in a collection of (overlapping) papers by (subsets of) Feige, Fiat, and Shamir [35, 36, 39, 37]. We will define and discuss several concepts – proofs of knowledge, trapdoor bit commitment schemes, witness indistinguishability, and witness hiding – and then combine them to obtain a constant-round zero knowledge argument for Hamiltonicity. Finally, using these concepts, we will discuss *noninteractive* zero knowledge proofs [11, 14, 38].

8.1 Trapdoor Commitment Schemes

Definition 8.1 (Trapdoor Bit Commitment) *A* trapdoor *bit commitment scheme is a bit commitment scheme in which at the end of the* COMMIT *phase the* receiver *can construct (using some trapdoor information) "commitments" that are indistinguishable from the sender's commitments but that the receiver can open at will, either to 0 or to 1.*

Feige and Shamir [35] observed that the basic block in the zero-knowledge interactive proof for Hamiltonicity described above, yields a trapdoor commitment scheme (based on any non-trapdoor commitment scheme). Specifically:

1. $R \longrightarrow S$: R chooses a Hamiltonian graph G (in which S is presumed unable to find a Hamiltonian cycle; see [35] for more details), and sends it to S.
2. $S \longrightarrow R$:
   - To commit to 0, S chooses a random permutation π and commits (using a non-trapdoor commitment scheme) to the adjacency matrix of $\pi(G)$. To reveal 0, S opens the entire adjacency matrix.
   - To commit to 1, S commits to a random permuation matrix M. To reveal 1, S opens the "1" entries in M.

Note that if R knows a cycle in G, then R can commit to the adjacency matrix of $\pi(G)$ and then either reveal the entire matrix or reveal a cycle in the permuted graph. Thus the commitment is a trapdoor commitment.

8.2 Proofs of Knowledge

If an arbitrarily powerful prover P proves, say, in ZK, that a graph G is Hamiltonian, then we can assume that P "knows" or "could produce" a Hamiltonian cycle for G. However, if the prover is probabilistic polynomial time bounded, then we can't necessarily make such an assumption (even if we had a good definition). On the other hand, the protocol given in Section 7.1 seems to have this property, since, as noted earlier, if the prover were to answer both of the possible queries then the replies could be combined to obtain a Hamiltonian cycle. In another setting, sometimes a fact – *e.g.*, that x is a quadratic residue modulo N – is not in question (for example, x may have been issued by a trusted center); what needs to be proved is that a party A "knows" a square root of x mod N. Feige, Fiat, and Shamir formalize the notion of a proof of knowledge that $x \in L$ by requiring that there exists a probabilistic polynomial time *knowledge extractor* that, by interacting with the prover (possibly including rewinding the prover to an earlier point in the proof) can produce a witness for $x \in L$ (more precisely, they require that the probability that the verifier will accept the proof is negligibly close to the probability that the extractor will succeed). As we will see in Section 8.5, the formalization turns out to be extremely useful technically when proofs of knowledge are used as subroutines in other constructions. Our presentation follows closely that in [35].

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

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

1. Completeness: $\forall x \in L\, \forall w \in w(x)$
$$\Pr[(P, V)(x, w) \text{ accepts}] > 1 - \nu(n)$$
where the probability is over the coin flips of P and V.

2. Soundness: there exists an expected probabilistic polynomial time knowledge extractor M such that for all (possibly cheating) probabilistic polynomial time provers P' and $\forall x \forall w'$
$$|\Pr[(P', V)(x, w') \text{ accepts}] - \Pr[M(x, P'(x, w')) \in w(x)]| < \nu(n)$$
where the probability is taken over the coin flips of V and M.

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

A proof of knowledge is zero knowledge if it satisfies (one of) the definitions of zero knowledge.

8.3 Witness Indistinguishability

The concept *witness indistinguishability* was proposed by Feige and Shamir [36] as an alternative to zero-knowledge. Unlike the case with zero-knowledge, witness indistinguishabiltiy is closed under parallel and sequential composition. In this section all machines (provers, verifiers, and cheating versions of each) are probabilistic polynomial time bounded.

Definition 8.2 (Witness Indistinguishability) *Proof system (P, V) is witness indistinguishable if for any V', for all sufficiently large x, for all $w_1, w_2 \in w(x)$, and for all auxiliary input z to V', the distribution on the views of V' following an execution $(P, V')(x, w_1, z)$ is indistinguishable from the distribution on the views of V' following an execution $(P, V')(x, w_2, z)$.*

Note that the auxiliary input z can even be the two witnesses w_1, w_2. Thus, *even knowing both witnesses*, V' should not be able to distinguish which witness is being used by P. The proof of the next theorem relies on the fact that the simulator whose existence is implied by the zero knowledge property produces its view *without access to any witness.*

Theorem 8.1 ([35]) *Every zero-knowledge protocol is witness indistinguishable.*

Theorem 8.2 ([35]) *Witness indistinguishability is preserved under parallel composition of protocols.*

Exercise 8.1 *Prove Theorem 8.2 using a hybrid argument.*

8.4 Witness Hiding

We now turn to the concept of *witness hiding.* Intuitively, for example in the terminology of the previous section, a protocol (P, V) for membership in a language $L \in \mathrm{NP}$ is witness hiding if execution of the protocol on an input $x \in L$ does not *help* a (possibly cheating) verifier to compute a witness $w \in w(x)$.

Definition 8.3 (Witness Hiding) *Let (P, V) be a proof of knowledge system for a relation R and let G be a generator for this relation ($G(1^n)$ outputs an instance $(x, w) \in R$ of length n). (P, V) is* witness hiding *on (R, G) if there exist a witness extractor M which runs in expected polynomial time, such that for any nonuniform polynomial time V' the probabilty that V', after engaging in $(P, V')(x, w)$, outputs an element in $w(x)$ is negligibly close to $\Pr[M(x, V') \in w(x)]$, where the probability is taken over the distribution of the inputs and witnesses, as well as over the coin flips of P and M. The witness extractor is allowed to use V' and G as black boxes (oracles), but not P.*

Note that the definition of witness hiding is with respect to a specific generator.

It is not the case that witness indistinguishability implies witness hiding (for example, there may only be a single witness in $w(x)$, so even a protocol that announces $w(x)$ is witness indistinguishable. In the remainder of this section, we establish a connection between witness indistinguishability and witness hiding.

Definition 8.4 (Invulnerable Generator) *A random polynomial time algorithm G is an invulnerable generator for a relation R if on input 1^n it produces instances $(x, w) \in R$ of length n, such that for any nonuniform polynomial time* cracking *algorithm C*

$$\Pr[(x, C(x)) \in R] < \nu(n)$$

where the probability is taken over the coin flips of G and C.

Definition 8.5 (Compound Relation) *Let $R = \{(x, w)\}$. The compound relations R^2 associated with R is defined by*

$$((x_1, x_2), w) \in R^2 \text{ iff } (x_1, w) \in R \vee (x_2, w) \in R$$

If G is a generator for R, then a generator for R^2 can be obtained by running G twice, independently, and then discarding one of the witnesses. We call this generator G^2 (despite the fact that this notation suggests that the output is two (element,witness) pairs).

Theorem 8.3 ([36]) *Let (P, V) be a proof of knowledge system for R^2. Then if (P, V) is witness indistinguishable over R^2 then (P, V) is witness hiding over (R^2, G^2).*

The proof of this theorem is difficult. However, in typical cryptographic scenarios we assume that finding *any* witness is intractable. In this case we have the following simplified version of the theorem.

Theorem 8.4 ([36]) *Let G be an invulnerable generator for a relation R. Let (P, V) be a proof of knowledge system for R^2. Then if (P, V) is witness indistinguishable over R^2, then it is witness hiding over (R^2, G^2).*

Proof: Assume for the sake of contradiction that there exists a probabilistic polynomial time bounded (possibly cheating) V' that, after interacting with P can crack an instance of R^2 produced by G^2 with probability at least n^{-k}. We will construct a machine M that, without interacting with P, that has a non-negligible chance of cracking an instance of R produced by G, violating the assumed invulnerability of G.

On input x, M uses G to generate a solved instance $(x', w') \in R$. M then runs $(P, V')((x, x'), w')$ (that is, the common input is (x, x') and P's private input is w').

By assumption, V' generates a witness to one of the two instances x and x' with probability n^{-k}. By the witness indistinguishability of (P, V), the instance to which V' produces the witness is independent of the particular witness (x') used by P. Thus, with probability negligibly close to $n^{-k}/2$ V' produces a witness in $w(x)$. The probability is over G^2 (not $G!$), that is, over choice of both $x and x'$, and the coin flips of all the parties. ∎

8.5 NP $\in$ 4-Round ZK

We now have all the tools we need to construct a 4-round noninteractive zero knowledge argument for Hamiltonicity (and hence for any language in NP). At a high level, the idea is as follows: Let G be a (purported) Hamiltonian graph. First, the *verifier* chooses a trapdoor bit commitment scheme K and proves knowledge of the trapdoor to the prover using a 3-round witness hiding proof of knowledge system based on the generic 3-round protocol (not known to be zero knowledge). Then the *prover* uses the implementation of the generic 3-round protocol for Hamiltonicity in which the commitments are performed using K.

- Completeness is trivial.
- Soundness follows from the fact that (by the witness hiding property) the prover does not know the trapdoor and hence the generic 3-round protocol is itself still sound.
- Zero knowledge is argued as follows. Let V' be a possibly cheating verifier. The simulator interacts with V' to extract the trapdoor. (A witness extractor exists by definition of a proof knowledge.) Then, using the trapdoor, the simulation of the 3-round protocol is *straight line* – it does not even require rewinding: given the trapdoor information the simulator can create commitments that are indistinguishable from those created by the prover, and can open them at will either to 0 or to 1, as needed.

By overlapping executions of the first and second parts of the protocol, the total communication is 4-rounds (4 messages, alternately, starting with a message from V to P)[5].

8.6 Proofs of Identity

Feige, Fiat, and Shamir [37] suggest the following variation on a proof of identity due to Fiat and Shamir [39]. The scheme assumes the existence of a trusted center whose sole purpose is to publish a modulus n which is the product of two large primes of the form $4r + 3$. As noted above, for such n, -1 is a quadratic non-residue modulo n,

[5] In [35] this communication pattern is called 2 rounds – *i.e.*, two round trips.

with Jacobi symbol $(\frac{-1}{n}) = 1$. All proofs of identity, by all parties, will use the same n.

Each user A generates a *public identification key* p_A as follows:

1. Choose k random numbers $s_1, \ldots, s_k \in \mathbb{Z}_n$. These form the *private* identification key and must be kept secret.
2. For $1 \leq j \leq k$, choose $b_j \in_R \{-1, 1\}$ and set $I_j = b_j s_j^{-2} \bmod n$.
3. Publish $p_A = (I_1, I_2, \ldots, I_k)$.

Each s_j is a witness to the quadratic residuousity character of I_j; the s_j are "hidden" by the difficulty of extracting square roots modulo n (which is equivalent to factoring). The choice of $I_j = \pm s_j^{-2}$ rather than $\pm s_j^2$ has practical but not theoretical implications.

A proof of identity is performed as follows.

1. A chooses $r \in_R \mathbb{Z}_n$ and sends $X = \pm r^2 \bmod n$.
2. B sends $e_1 e_2 \ldots e_k \in_R \{0, 1\}^k$.
3. A sends $Y = r \prod_{e_j = 1} s_j \bmod n$.
4. B verfies that $x = \pm Y^2 \prod_{e_j = 1} I_j \bmod n$.

The next theorem follows from the *proof* of a related theorem in [37]. However, [37] precedes the work on witness hiding.

Exercise 8.2 *Prove that the identification scheme is a proof of knowledge of the square roots of the inverses of the I_j.*

Theorem 8.5 *The identity protocol is witness indistinguishable.*

The key to the proof of Theorem 8.5 is that each I_j has four square roots, but only two are known to A. Even an infinitey powerful B' cannot determine from the X's and Y's (of polynomially many adaptively chosen executions against B') which of the square roots A uses. This is because

$$Y = r \prod_{e_j = 1} s_j \bmod n$$

so if A replaces s_j by one of the other three square roots, then Y is multiplied by one of the three non-trivial square roots of 1. This effect can be cancelled by dividing r

by the same square root, leaving $X = \pm r^2 \bmod n$ unchanged. Since the r's are chosen uniformly, A produces the same X, Y values with the same probability distribution in both cases. Thus, A cannot leak to B' during the executions of (A, B') which square roots he knows.

Very informally, security for the identification scheme may be argued as follows. (The argument is particularly informal since we have note even defined security for an identification scheme.) Suppose that after some polynomial number of interactions with A, a probabilistic polynomial time B' could impersonate A with some probability p. Since the identification scheme is a proof of knowledge we can use the impersonator to extract square roots of randomly chosen squares modulo n: Choose a public key for A as described above. Interact with B' as needed. Now let B' impersonate A; with probability p^2 we can extract a square root for one of the I_j. By the witness indistinguishability, with probability $1/2$ this square root is not one of the 2 known to us from the time of the generation of the public key.

Exercise 8.3 *Define security for an identification scheme and show that the Fiat-Shamir protocol satisfies the identification scheme. Initially, confine the definition and proof to the case of* sequential *executions of the protocol. What happens if the scheme is executed by the prover many times concurrently?*

8.7 Non-Interactive Zero-Knowledge Proofs (NIZKs)

Blum, Feldman, and Micali [11] suggested the intriguing concept of a *non-interactive* zero-knowledge proof, to be carried out in a slightly modified computational model in which there is a *public random string*, called a *reference* string. Informally, given the reference string, σ, prover P on common input x and private input y, writes down a "proof" that $x \in L$; the proof will depend on σ. At any later time the verifier V, given x, σ, and the proof π, can check the validity of π. More accurately, if the triple (x, σ, π) passes V's acceptance test we can conclude that either σ is *not* truly random or $x \in L$. In other words, the probability over choice of σ that a cheating prover can convince V to accept $x \notin L$ is negligible. The "dirty secret" of non-interactive zero-knowledge proofs is that the simulator can choose the reference string for the simulated proofs.

In this section we will discuss Feige, Lapidot, and Shamir's constructions of non-interactive zero-knowledge proofs under standard assumptions: one-way permutations (arbitrarily powerful provers) or so-called *certified* trapdoor permutations (probabilistic polynomial time bounded provers) [38]. As usual, we will focus on the case of probabilistic polynomial time bounded provers. The difficulty will be in arranging that the honest but restricted provers can actually carry out the proof (completeness). Soundness will hold independent of the power of the prover, *i.e.*, even for arbitrarily powerful cheating provers. We next give a formal definition for a non-interactive zero knowledge proof, and then discuss some subtleties in the definition.

Definition 8.6 (Polynomially Bounded Relation) *A binary relation R is* polynomially bounded *if R is decidable in polynomial time and there exists a polynomial $p(\cdot)$ such that for all $(x, w) \in R$, $\|w\| \leq p(\|x\|)$. We let $L_R = \{x | \exists w : \ (x, w) \in R\}$.*

Definition 8.7 (Non-Interactive Zero Knowledge Proof) *A non-interactive proof system for a language $L_R \in NP$ is a pair of probabilistic algorithms (P, V), where V is polynomial time bounded (as is P in the case of arguments), for which there exist constants b, c satisfying:*

- *Completeness: $\forall (x, w) \in R$, $\forall \sigma$ of length $n^b k^c$:*

$$V(x, \sigma, P(x, w, \sigma)) = \text{ accept}$$

- *Soundness: $\forall n \geq 1$ at least $(1 - \nu(k))$ of the strings of length $n^b k^c$ satisfy:*

$$\forall x \in \{0,1\}^n \setminus L_R \forall y \ V(x, \sigma, y) = \text{ reject}$$

- *Zero-Knowledge: There exists an expected probabilistic polynomial time bounded simulator M such that, for all $x \in L_R$ the ensembles $\{(\sigma, P(x, w, \sigma))\}$ and $\{M(x)\}$ are non-uniform polynomial-time indistinguishable.*

$$\exists M \, \forall D = \{D_\ell\} \, \forall (x, w) \in R \, \forall d \geq 1 \, \exists K \geq 1 \, \forall k > \max(K, \|x\|)$$
$$\Pr[D_k(M(x)) = 1] - \Pr[D_k(\sigma, P(x, w, \sigma)) = 1]| < \frac{1}{k^d}$$

where the probability space is taken over the random choices of $\sigma \in_R \{0,1\}^{\|x\|^b k^c}$ and over the random coin flips of P and M.

Here, k is a security parameter that quantifies both soundness and zero-knowledge. For simplicity, we may take $k = n$.

There are several subtleties in the definition, most significantly:

(1) the definition is for *bounded* non-interactive zero knowledge proof systems, in which the prover may only prove a bounded number of statements (the number is determined by the length of the reference string); ultimately, we are interested in the case in which σ can be reused any polynomial (in the security parameter) number of times without compromising the zero-knowledge property. Note that this property is stronger than what is needed in many contexts, even when a reference string is re-used. For example, the cca-pre and cca-post cryptosystems of [69, 32] use NIZKs; the reference string is part of the public key. However, even though the same string is used for many encryptions, bounded zero-knowledge suffices (think about why this might be so!). This is not completely surprising: for example, a digital signature is decidedly not zero-knowledge, yet it affords enough protection of the private key that

the same key can be re-used for many messages. We discuss multiple non-interactive zero knowledge proofs in Section 8.7.4.

(2) the definition assumes independence of the statement to be proved and the reference string σ. Since ultimately we wish to use NIZKs in cryptographic applications, we must handle the adaptive case in which x may be chosen as a function of σ and the history. See Section 8.7.5 below.

8.7.1 Outline of the Basic NIZK Construction

For ease of presentation, we will initially assume the prover has unbounded computational power. At a high level the Feige, Lapidot, and Shamir construction of a non-interactive zero-knowledge proof of *directed* Hamiltonian cycle in an n-node directed graph G is as follows.

1. The reference string σ is interpreted as a (shorter) sequence of commitments to *hidden* random bits; the string of hidden (committed) bits is denoted H. Only the prover knows the bits of H, but during the construction of the proof the prover can reveal bits of H as necessary.
2. The hidden random string H is interpreted as a sequence of $n \times n$ adjacency matrices in such a way that (with overwhelming probability) at least one of these is the adjacency matrix of a proper permuation graph. For each matrix A the prover must either (1) prove to V that the matrix A is not the adjacency matrix of a permutation graph (done by revealing all the entries of the matrix) or (2) use A to prove that G is Hamiltonian, in a method to be described next. If G is non-Hamiltonian then P will fail to convince V unless none of the matrices defined by H are permutation matrices, which we will show occurs with negligible probability over choice of σ.

Let A be a commitment to a randomly chosen proper permutation matrix. For now, we will abstract out the issue of how the commitments are created. We will use two facts: P can decommit entries in A and V knows that A represents a permutatation matrix but knows nothing about the permutation.

P, knowing a cycle in G, computes a permuation π such that A represents a cycle in $\pi(G)$. To convince V that G is Hamiltonian, P sends to V the permutation π and the opening of all the (i, j) entries of A such that $(i, j) \notin E(\pi(G))$. V accepts the proof if and only if all the opened entries are zero. The intuition for zero knowledge is as follows. Assume that V knows that A is a proper permutation matrix (but doesn't know which permutation matrix it is). V receives only a random permutation π (it is random because A is random) and a collection of random commitments to bits (the entries of A), together with the opening of the bits $A(i, j)$ for which $(i, j) \notin E(\pi(G))$. All these openings are of zeros.

The simulator may therefore work as follows. On input G, the simulator chooses a random permutation π with uniform distribution and creates a random commitment to an all-zero adjacency matrix Z. The simulator then produces the commitment to Z, the permutation π, and the openings of the commitments to the bits $Z(i,j)$ for which $(i,j) \notin E(\pi(G))$ (which, of course, are all zeros). Assuming that the commitment scheme is semantically secure, the simulated transcripts are indistinguishable from real transcripts.

Roughly speaking, the commitment to A will be part of the reference string. As noted earlier, the simulator is permitted to generate simulated reference strings – in this case corresponding to the commitment to the all zero matrix Z. Since the common refence string will actually serve as a commitment to many matrices, in such a way that with overwhelming probability at least one of these is a permutation matrix, and since, let us assume that indeed at least one of these is a proper permutation matrix. Then since a cheating prover cannot falsely prove that a matrix is *not* a proper permutation matrix, the matrices that the prover does not "throw out" must contain at least one proper permutation matrix. For each A not discarded by the prover, the prover carries out a proof with the verifier as outlined above, so at least one of the proofs is valid. Thus, the system will be sound. Completeness is straightforward.

8.7.2 Obtaining H from σ

Let f be a one-way permutation on strings of length k, and let B be a hard-core predicate for f. σ is viewed as the concatenation of polynomially many blocks $u_1, u_2, \ldots$ each of k bits. Let I (for *intermediate*) be the string defined by:

$$I_j = B(f^{-1}(u_j))$$

Each successive block of $m = \log(n^3)$ bits of I yields a bit of H as follows: if all m bits are 1, then the resulting bit of H is 1, otherwise the resulting bit of H is 0. Thus, the probability that an m-bit block of I yields a 1 is $\frac{1}{n^3}$. If we partition H into consecutive blocks $z_1, z_2, \ldots$ of n^4 bits, then each block represents an adjacency matrix (say, in row-major form) of an $n^2 \times n^2$ matrix with an expected n non-zero entries. A simple argument (see [38]) shows that with probability at least $\Omega(n^{-3/2})$ the n 1's are located in n distinct rows and n distinct columns and the submatrix induced by these rows and columns is Hamiltonian. Such a matrix is n-proper; the $n \times n$ submatrix induced by these rows and columns is a permutation matrix. Thus, if the reference string (and therefore I and H) is sufficiently long, then with overwhelming probability one of the blocks z_j of H will represent an n-proper matrix and hence yields a proper permutation matrix.

The prover can open a bit of H as follows. Recall that each H_j is determined by m consecutive bits of I. To open a bit of I determined by block u_a of σ, the prover reveals $w = f^{-1}(u_a)$. The verifier can compute $B(f^{-1}(u_a))$, and can also check that $f(w) = u_a$.

To prove that a matrix is not n-proper, the prover opens all the commitments to the elements in the matrix. There are three cases, each of which can be easily checked by the verifier:

1. the matrix does not contain exactly n 1's;
2. the matrix contains exactly n 1's but there is some row or column with more than one 1; or
3. the matrix contains exactly n 1's, and these form a permutation matrix, but it is not a proper permutation (*i.e.*, a single cycle).

If the matrix is n-proper, then P can open the entries in the $n^2 - n$ rows and $n^2 - n$ columns that contain only zeroes. The resulting $n \times n$ submatrix (still hidden) can be used in the basic proof as described above.

To summarize, the NIZK for directed Hamiltonicity of a directed graph G on n nodes is as follows.

Let f be a fixed one-way permutation on $\{0,1\}^k$ and let B be a hard-core predicate for f. Let σ be a sufficiently long random reference string ($\Omega(n^{13/2}k^2m)$ bits). Then σ is a bit-by-bit commitment to a random string H, which is viewed as a sequence of $n^2 \times n^2$ adjacency matrices. With overwhelming probability (e^{-nk}) one of the matrices represents an n-proper matrix, from which we can obtain a proper permutation matrix of size n. For each matrix A in H, P either opens all the entries to show that it is not n-proper, or P uses A to prove Hamiltonicity of G. P cannot falsely convince the verifier that a graph is improper, so one of the unopened graphs is indeed n-proper. Thus, with overwhelming probability over choice of σ, P will not open all the graphs.

Completeness is immediate, but uses the arbitrary computing power of P to invert f (needed for opening the commitments). Soundness follows from the fact that f is a permutation (so the commitments are strong-sender commitments) and the fact that P uses some proper A in proving Hamiltonicity. Zero knowledge is more complicated and discussion is deferred to the next section.

Polynomial Time Bounded Provers If the prover is polynomial time bounded then we must use a collection of *certified* trapdoor permutations. This is a collection (I, D, F) of trapdoor permutations (Definition 1.5) with the additional property that, there is a polynomial time algorithm that, given the function index i, can determine that f_i is indeed part of the collection – and in particular that f_i is a permutation (1-1 and length preserving). Assuming we have such a collection, the prover runs $I(1^k)$ to obtain an index $i \in S$ and a trapdoor t_i for inverting strings $f_i(x)$. The prover adds i to the proof, and any information needed to help the verifier certify f_i as a member of the prespecified collection of trapdoor permutations.

The only problem with this approach is that the prover has control over i (and hence over choice of f_i), and can even choose i as a function of σ. This is combatted by brute force, increasing the length of the reference string. The prover may choose any trapdoor permutation from a family of $k! < 2^k$ permutations. If e^{-nk} was an upper bound on the fraction of bad choices for σ of length $n^{13/2}k^2m$ bits for a *fixed* f, then $2^k e^{-2nk} < 2^{-k}$ is an upper bound on the fraction of random strings of length $2n^{13/2}k^2m$ bits which can be bad relative to some f_i. Thus, with probability at least $1 - 2^{-k}$, σ will not be bad relative to *any* f_i, so even if the cheating prover is arbitrarily powerful it will not be able to find a bad choice for i because with overwhelming probability none exists.

8.7.3 Proving Zero Knowledge for the NIZK

We briefly outline the simulator for the NIZK proof of directed Hamiltonicity described above. The simulator's task is slightly easier when the prover is probabilistic polynomial time bounded, since in this case the simulator does not have to simulate inversion of a one-way permutation. We assume we are in this case. On input G and 1^k, the simulator outputs a string σ of length n^7k^2m and a "proof" of Hamiltonicity based on a modification of σ obtained as follows. Having chosen f_i, the simulator can determine which matrices in the hidden random string H are n-proper and which are not. For those that are not proper, the simulator opens all the entries of the matrix. For those that are proper, the simulator changes σ so that these matrices are now all zero. In particular, if a particular block of km bits in σ yields a 1 in one of these matrices, then the simulator repeatedly chooses new values for the bits in this block of σ until the resulting bit in the new hidden random string is 0 (the remaining bits in σ are not touched during this process, only the particular km that yielded the 1 that the simulator is trying to convert to 0). The simulator can then carry out a simulated proof of Hamiltonicity using each of these all-zero matrices as described above. See [38] for details both for this case and for the case of the unbounded prover.

8.7.4 Multiple NIZKs

Feige, Lapidot, and Shamir use a beautiful combination of a simple transformation and techniques of witness indistinguishability to achieve multiple non-interactive zero knowledge proofs based on a single reference string. The construction has the property that each of the proofs may be carried out independently of the others, so polynomially many provers may share the same reference string.

Definition 8.8 (General NIZK) *A non-interactive proof system for the language L_R is* general zero knowledge *if there exists a random probabilistic polynomial time bounded simulator M such that for any positive constant c, for any $m \leq n^c$, the two ensembles $\{(\sigma, P(x_1, w_1, \sigma), \ldots, P(x_m, w_m, \sigma))\}$ and $\{M(x_1, x_2, \ldots, x_m)\}$ are polyno-*

mial time indistinguishable. In any sequence of instances that indexes the ensembles, all x_1 are of the same length, n, and $\forall 1 \leq i \leq m$, $(x_i, w_i) \in R$.

Intuitively, the transformation is as follows. Fix a pseudo-random generator g. Rather than proving a given theorem T, the prover proves the statement $S =$ "σ is pseudo-random or T is true" In "real" proofs, σ is random (and so is extremely unlikely to equal the output of g on *any* seed), so if S is true then T is true; hence soundness is preserved. In the simulated proofs, the reference string will indeed be pseudo-random. Now, zero-knowledge proofs with polynomial time bounded provers are witness indistinguishable, so if S is proved using a bounded NIZK the proof will be witness indstinguishable. Witness indistinguishability is closed under composition, so non-interactive bounded witness indistinguishability implies non-interactive general witness indistinguishability. This is used to prove that the transformed system is general non-interactive zero-knowledge:

- An auxiliary machine Q, used only for the proof, is defined as follows: Q, like P, has witnesses for all the instances (theorems) $x_1, x_2, \ldots, x_{poly}$ proved by P. Let us call these *real* witnesses. Q simulates a reference string as described above, so it also has a witness s to the fact that the reference string is pseudo-random. Q uses its real witnesses (not s) in generating proofs.

- Q's output is indistinguishable from a truly random reference string and real transcripts produced by P because these differ only in the way in which the reference string was created: pseudo-randomly in Q's case and truly at random otherwise.

- Q's output is indistinguishable from the simulator's output by witness indistinguishability: Q uses the real witnesses, while the simulator uses s.

We now make this more precise, presenting the key definitions and a more rigorous outline of the proof.

Definition 8.9 (Bounded Witness Indistinguishable) (P, V) *is* bounded witness indistinguishable *over R if for any sufficiently large input x, any $w_1, w_2 \in w(x)$, and for a randomly chosen reference string σ, the ensembles which differ only in the witness that P is using, but not in x or σ, are polynomial time indistinguishable. That is, $\forall$ probabilistic polynomial time $D \, \exists N \, \forall n > N \, \forall x \in L_R \cap \{0,1\}^n \, \forall w_1, w_2 \in w(x)$,*

$$\sum_{\sigma} 2^{-|\sigma|} \cdot |\Pr[D(P(x, w_1, \sigma)] - \Pr[D(P(x, w_2, \sigma)]| < \nu(n)$$

where the probability space is over the coin flips of P and D.

The definition says that the proofs must look similar when both proofs use the same reference string. This is the definition given above, indicated by the fact that the absolute value is taken *inside* the summation. Another reasonable definition would not require that the reference string not be the same in both cases, and intuitively this is the case we are interested in (the real prover uses truly random strings and a witness to T; the simulator uses pseudo-random strings and the seed to the pseudo-random generator). This is indicated in the following definition, where the absolute value is outside the summation:

Definition 8.10 (Bounded Witness Indistinguishable (alternate)) (P, V) *is* bounded witness indistinguishable *over R if for any sufficiently large input x, any $w_1, w_2 \in w(x)$, and for a randomly chosen reference string σ, the ensembles which differ only in the witness that P is using, but not in x or σ, are polynomial time indistinguishable. That is, $\forall$ probabilistic polynomial time $D\,\exists N\,\forall n > N\,\forall x \in L_R \cap \{0,1\}^n \forall w_1, w_2 \in w(x)$:*

$$|\sum_{\sigma} 2^{-|\sigma|} \cdot (\Pr[D(P(x, w_1, \sigma)] - \Pr[D(P(x, w_2, \sigma)])| < \nu(n)$$

where the probability space is over the coin flips of P and D.

Exercise 8.4 *Prove that if P is probabilistic polynomial time bounded, then the two definitions of bounded witness indistinguishability are equivalent.*

Lemma 8.1 *Any bounded non-interactive zero knowledge proof system with polynomial time prover is also a bounded non-interactive witness indstinguishable proof system.*

Exercise 8.5 *Prove Lemma 8.1.*

Definition 8.11 (General Witness Indistinguishable) *A non-interactive proof system is* general witness indistinguishable *over R if for any positive constant c, for any $m \leq n^c$, the ensembles*

$$\{(P(x_1, w_1^1, \sigma), P(x_2, w_2^1, \sigma), \ldots P(x_m, w_m^1, \sigma))\}$$

and

$$\{(P(x_1, w_1^2, \sigma), P(x_2, w_2^2, \sigma), \ldots P(x_m, w_m^2, \sigma))\}$$

are polynomially indistinguishable, where σ is identical in the two distributions (as in Definition 8.9).

The next lemma may be proved by a hybrid argument.

Lemma 8.2 *Any bounded non-interactive witness indistinguishable proof system with probabilistic polynomial time bounded provers is also general witness indistinguishable.*

Formal Description of the Transformation Let g be a pseudo-random bit generator that on input an n-bit seed outputs a $2n$-bit seed. Let (P, V) be a bounded NIZK with polynomial time prover for the NP complete language L_R. We construct (P', V'), a general NIZK proof system for L_R, under the assumption that one-way functions exist (used in constructing the generator g).

Define the relation $R_g(y, s)$ to be true iff $g(s) = y$. L_{R_g} is the corresponding NP language (the language of strings that are output by g on some seed).

Define the relation $R_{\#}(x\#y, w)$ to be true iff either $R(x, w)$ or $R_g(y, w)$, where $\#$ is a delimiter.

P' behaves as follows on input $(x, w) \in R$ and reference string σ.

1. Divide σ into two segments $\sigma = y \cdot \sigma'$, where $\|y\| = 2n$. y is called the *reference statement.*

2. Construct the instance $x\#y \in L_{R_{\#}}$. The witness w for $x \in L_R$ is a witness for $x\#y \in L_{R_{\#}}$.

3. Reduce $x\#y$ to an instance X of the NP complete language L_R using a publicly known reduction with efficient transformation of witnesses; in the process, reduce w, the witness for $x\#y \in L_{R_x}$ to W, a witness for $X \in L_R$.

4. Send x, X, and $P(X, W, \sigma')$ (this last term is a random variable because P is randomized).

This completes the description of the transformed prover's protocol. The transformed verifier V' accepts if the reduction is correct (gives X when applied to $x\#y$) and if $V(P(X, W, \sigma'))$ accepts.

Theorem 8.6 *[38] If (P, V) is a bounded NIZK with efficient provers for an NP-complete language L_R, and g is a pseudo-random bit generator, then the transformed scheme is a general noninteractive zero knowledge proof system for L_R.*

Proof: (Sketch) Completeness (while preserving efficiency of the prover) is inherited from completeness of (P, V). Soundness follows from the fact that y is truly random and the soundness of (P, V). As suggested above, the simulation of (P', V') replaces the truly random reference statement y with a pseudo-random reference statement y' generated by selecting an n-bit seed s and setting $y' = g(s)$. Then any statement x with a witness w is transformed into a statement X which has *two* witnesses: w and s. The simulator uses s instead of w; witness indistinguishability implies that the choice of witness not detectable to probabilistic polynomial time observers, even if this is done polynomially many times.

In a little more detail, the proof is broken into two steps. Assume that P' proves membership in R_L for inputs $x_1, x_2, \ldots, x_m$ using reference string σ and respective witnesses $w_1, w_2, \ldots, w_m$. The simulator M will be as we have already discussed: it generates a pseudo-random reference statement y' and a truly random reference string σ'. For each $1 \leq j \leq m$, M reduces $x_j \# y'$ to an instance X_j of L_R and derives from s a witness S for X_j. It then uses $P(X_j, S, \sigma')$ to simulate a proof that $X_j \in L_R$.

Let Q be a machine that on inputs $(x_1, w_1), \ldots, (x_m, w_m)$ constructs y' pseudo-randomly as M does, but simulates the proofs using the witnesses $w_1, \ldots, w_m$ as P' does.

In the first part of the proof we show that P's output is indistinguishable from that of Q. The proof is by contradiction: these two ensembles of non-interactive proofs differ only in whether the reference statement is truly random or pseudo-random; any ability to distinguish them would imply an ability to distinguish pseudo-random from truly random strings.

In the second part of the proof we show that Q's output is indistinguishable from that of M. The proof appeals to witness indistinguishability. These two ensembles differ only in which choice of witness is used in the proofs; by witness indistinguishability they cannot efficiently be distinguished. ∎

Remark 8 *Note that at the time that it generates the simulated reference string, the simulator does not need to know the theorems for which it will have to simulate proofs.*

Exercise 8.6 *Complete the proof of Theorem 8.6.*

8.7.5 Security Against Adaptive Adversaries

Feige, Lapidot, and Shamir define the following *adaptive indistinguishability* test (also called an *AI* test in part because of its origins as Turing's test for artificial intelligence). An adversary $\mathcal{A}$ can interact with either a prover P or a simulator M in a black box fashion. Let B denote the party with whom $\mathcal{A}$ interacts. $\mathcal{A}$ first requests the reference string σ. If $B = P$ then P replies with a truly random string. If $B = M$ then M replies with a string of its choice. Based on everything it has seen so far, $\mathcal{A}$ generates a pair $(x, w) \in R$ and sends it to B[6]. If $B = P$ then P responds with a non-interactive proof $P(x, w, \sigma)$. If $B = M$ then M ignores w ([38] say that w is "magically" filtered away), and M must simulate a non-interactive proof that $x \in L_R$. This process continues, with new elements of R being chosen by $\mathcal{A}$ as a function of everything seen before – that is, *adaptively* – polynomially many times, at which point $\mathcal{A}$ outputs either 0 or 1. The pair (M, P) is said to pass the AI test if for all probabilistic polynomial time bounded adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ outputs 1 when $B = P$ is negligibly close to the probability that $\mathcal{A}$ outputs 1 when $B = M$.

[6] We assume there is a probabilistic polynomial time machine for sampling (perhaps not uniformly) from R.

We can modify the test to capture witness indistinguishability as follows: $\mathcal{A}$ generates triples (x, w_1, w_2), where both w_1 and w_2 are witnesses that $x \in L_R$, and sends the triplet to B. B will contain either P_1 or P_2. P_1 chooses the lexicographically smaller witness, say, w_1, and generates a proof $P(x, w_1, \sigma)$. P_2 chooses the lexicographically larger witness w_1, and generates a proof $P(x, w_2, \sigma)$.

Definition 8.12 (Adaptive Witness Indistinguishable) *A non-interactive adaptive proof system* (P, V) *is* adaptive witness indistinguishable *if* (P_1, P_2) *pass the AI test for indistinguishability.*

It is proved in [38] that the transformation described in Section 8.7.4 is adaptive witness indistinguishable, first for single statements ($\mathcal{A}$ must pass judgement after a single non-interactive proof has been issued by B) and then because single-statement adaptive witness indistinguishability implies general witness indistinguishability (via a hybrid argument). Recall that the transformation *only* applies to probabilistic polynomial time bounded provers.

A Note on Efficiency Much more efficient constructions have been developed. The current state of the art is in [61].

9 Encryption II: Beyond Chosen Plaintext Attack

We now return to chosen-ciphertext security. After sketching Naor and Yung's ss-cca-pre scheme and its proof of security [69], we formally define non-malleability of a public-key cryptosystem under chosen ciphertext attack in the post-processing mode (nm-cca-post)[7], and present a general construction, due to Dolev, Dwork, and Naor, requiring the following tools:

- a public-key cryptosystem generator semantically secure with respect to relations under chosen *plaintext* attack;
- a family of universal one-way hash functions;
- bounded non-interactive zero knowledge proofs systems;
- a one-time signature scheme generator

9.1 The Naor-Yung SS-CCA-Pre Cryptosystem

The Naor-Yung scheme makes use of the fact that the definitions of both an interactive and a non-interactive proof system permit the possibility that the simulator

[7]The definitions apply, *mutatis mutandi*, to private key cryptosystems

can generate simulated "proofs" of false statements. Intuitively, something like this must be the case for statements whose truth cannot be ascertained in probabilistic polynomial time, since otherwise we could use the simulator and the verifier together to do precisely this.

In the Naor-Yung scheme, the public key consists of a random reference string σ and two keys E_1, E_2, drawn according to a probabilistic public-key cryptosystem generator that is semantically secure against chosen plaintext attacks.

Encryption is performed bit-by-bit. To encrypt a bit b, the sender computes encryptions under both public keys and then, using the reference string σ, provides a non-interactive zero-knowledge proof that both encryptions are of the same plaintext. The ciphertext is thus a triple (α, β, p).

Decryption is performed as follows. First the proof of consistency is checked. If verification of the proof fails then the decryption procedure outputs "invalid." If the proof succeeds, then decryption is performed using either one of the private keys corresponding to E_1 and E_2 respectively.

Let $\mathcal{A}$ be a probabilistic polynomial time bounded chosen ciphertext pre-processing attacker. Assume that after mounting a chosen ciphertext attack on a Naor-Yung public key, $\mathcal{A}$ has non-negligible probability of correctly destinguishing encryptions of 0 from encryptions of 1. We will show how to construct $\mathcal{B}$, a chosen *plaintext* attacker on the underlying ss-cpa cryptosystem.

We first begin with a *gedanken* experiment.

Let $\mathcal{T}$ (for "thought") be a machine that receives as input: 1^n, $E, E' \in_R GP(1^n)$, $\alpha \in_R E(b_1)$ and $\beta \in_R E'(b_2)$ for $b_1, b_2 \in \{0,1\}$, and one of the decryption keys D or D'. $\mathcal{T}$ runs the simulator to generate a reference string σ and a (possibly false) proof π that α and β encrypt the same bit. $\mathcal{B}$ gives to a cca-pre attacker $\mathcal{A}$ the public key (E, E', σ), and lets $\mathcal{A}$ run a cca-pre attack on this public key. For every ciphertext received from $\mathcal{A}$, $\mathcal{T}$ checks the proof of consistency to ensure that both ciphertexts (under E and E' respectively) encrypt the same plaintext, and, if so, replies with the plaintext (computed using the one decryption key). Since the reference string σ is *not* random but was chosen to accomodate the "proof" of a possibly false theorem, care must be taken in designing the NIZK argument system to ensure that $\mathcal{A}$'s proofs are really proofs (if σ were truly random this would be immediate).

After $\mathcal{A}$ completes its preprocessing chosen-ciphertext attack, it is given a ciphertext (a, b, p). Let $a \in E(b_1)$ and $b \in E(b_2)$ for some $b_1, b_2 \in \{0,1\}$. Define P_{00}, P_{01} and P_{11} as the probabilities that the output of $\mathcal{A}$ is $t = 1$, given that $b_1 = b_2 = 0$, $0 = b_1 \neq b_2 = 1$, and $b_1 = b_2 = 1$ respectively. The probabilities are taken over the choices of E and E', the random choices made in the encryption of b_1 and b_2, and the coin flips of the machines $\mathcal{S}$ and $\mathcal{A}$. These probabilities are independent of which of D or D' is known to $\mathcal{T}$.

By assumption, if the reference string were truly random then $\mathcal{A}$ distinguishes double encryptions of 0 from double encryptions of 1 (all accompanied by valid proofs of consistency). [69] argue this implies that $|P_{00} - P_{11}|$ is also non-negligible, and therefore so is at least one of $|P_{00} - P_{01}|$ or $|P_{01} - P_{11}|$. We assume without loss of generality that $|P_{01} - P_{11}|$ is non-neglible (otherwise, in the description of $\mathcal{B}$ below we must choose $\beta \in E'(0)$).

Using the assumed existence of the successful chosen-ciphertext-preprocessing attacker $\mathcal{A}$, we now construct $\mathcal{B}$, a chosen *plaintext* attacker on the underlying cryptosystem generator GP.

Let $E \in_R GP(1^n)$ and let and let $\alpha \in_R E(x)$ be a challenge ciphertext to $\mathcal{B}$, where $x \in \{0,1\}$. $\mathcal{B}$ runs $GP(1^n)$ to obtain (E', D'), and creates an encryption $\beta \in E'(1)$. $\mathcal{B}$ gives to $\mathcal{T}$ the unordered inputs $\{\alpha, \beta\}$, $\{E, E'\}$, and D'. $\mathcal{T}$ runs the experiment above and allows $\mathcal{A}$ to run a pre-processing chosen-ciphertext attack on the public key generated by $\mathcal{T}$. $\mathcal{B}$ gives to $\mathcal{A}$ the ciphertext (α, β, π) and outputs t, the value produced by $\mathcal{A}$. Since $Pr[t = 1|b = 0] = P_{01}$ and $Pr[t = 1|b = 1] = P_{11}$, $|Pr[t = 1|b = 0] - Pr[t = 1|b = 1]|$ is non-negligible, and thus we have a distinguisher for E. In other words E has been broken by a plaintext attack only, contradicting our assumption.

The proof of consistency allows $\mathcal{B}$ (via $\mathcal{T}$) to respond to queries from $\mathcal{A}$ during $\mathcal{A}$'s chosen-ciphertext preprocessing attack and is therefore indispensible to the reduction. The same idea will be used below in the full DDN scheme.

9.2 Definition of NM-CCA-Post

As noted in Section 3, the definition of security of a cryptosystem must specify both the type of attack considered and what it means to break the cryptosystem. Recall that in the chosen-ciphertext-post-processing attack 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.

Malleability, as defined informally in Section 3.4 and formally in Section 3.6, specifies what it means to "break" the cryptosystem. Recall that, 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 $R(\alpha, \beta) = 1$.

We now revisit these definitions and 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 auxiliary information about m in the form of hist(m), where hist is some polynomially computable function. $\mathcal{A}$ then engages in a second adaptive process of 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 $\mathcal{M}$[8]. Let $\beta = (\beta_1, \beta_2, \ldots)$. $\mathcal{A}$ is considered to have succeeded with respect to R if $R(m, \beta, \sigma)$. Let $\pi(\mathcal{A}, R)$ be the probability that $\mathcal{A}$ succeeds 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 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 $\mathcal{M}'$.

Note that $\mathcal{A}'$ is considerably weaker 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 9.1 *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.*

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).

[8] 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. The possibility of including plaintexts p in σ is included so that the definition can apply to symmetric, or private key, encryption.

9.3 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 σ for providing non-interactive zero-knowledge proofs of consistency (NIZKs proofs that various ciphertexts are encryptions of the same plaintext), and a universal one-way hash function.

The encryption process 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 NIZK 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.

It is instructive to compare this to the DDN Lite NM-CCA scheme described in Section 3.6 and to the Naor-Yung SS-CCA-Pre scheme [69]. In DDN Lite and here the plaintext is encrypted under multiple public keys of an ordinary SS-CPA cryptosystem; the method of choosing which keys to use for the encryptions is the same in both cases. The selection of keys is to force the attempted "mauler" to encrypt β under a key for which no encryption of α exists. In DDN Lite there are no non-interactive proofs of consistency, so only the string of ciphertexts is signed. Here for each bit b_i of the plaintext the string of (ciphertexts of b_i, proof of consistency of ciphertexts of b_i) pairs is signed.

In the Naor-Yung ss-chosen ciphertext pre-processing scheme the plaintext is encrypted under both keys, and a NIZK of consistency is given. [69] does not provide non-malleability, and so no selection among a set of keys is performed in that scheme.

As in [69], 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.

9.4 The Tools

We require a probabilistic public key cryptosystem that is semantically secure (Sections 3.2). Recall that GP denotes the key generator, e and d denote the public and private keys, respectively, and E and D denote, respectively, the encryption and decryption algorithms.

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

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

for some $b \in \{0,1\}$ and $r_1, r_2, \ldots, r_n \in \{0,1\}^{p(n)}$, for some polynomial p. The language of consistent encryptions $L = \{e_1, e_2, \ldots e_n, w | w \text{ is a consistent encryption}\}$ is in NP. For a given word $w = E_{e_1}(b, r_1), E_{e_2}(b, r_2), \ldots, E_{e_n}(b, r_n)$, the sequence $r_1, r_2, \ldots, r_n$ is a witness for its membership in L. In order to prove consistency we will use non-interactive zero-knowledge proof system for L with the property that the simulator need not know, when generating the simulated reference string, for which theorem it will have to generate a simulated proof (the [37] scheme described above enjoys this property; it is not crucial to the construction but it simplifies the proof). For completeness, we summarize exactly the properties needed for the construction.

Definition 9.2 *A triple $(\mathcal{P}, \mathcal{V}, \mathcal{U})$, where $\mathcal{P}$ is a probabilistic polynomial time machine, $\mathcal{V}$ is polynomial time bounded, 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 $z \in w(x)$, 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 $z \in w(x)$ then,*

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

for all polynomials q and sufficiently large n. Here $\mathcal{CONV}(x, z)$ is the probability distribution on conversations (U, p), where $U \in \mathcal{U}$ is a reference string and $p \in \mathcal{P}(x, z, U)$ is a proof[9].

Note that the length of the reference string 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 (Section 4.1). Recall that 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 should compress from any polynomial in n bits to n bits.

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 [7, 53, 68].

9.5 The Full DDN Scheme

We now describe the scheme $\mathcal{S}$ [32].

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 $\langle d_1^0, d_1^1, d_2^0, d_2^1, \ldots, d_n^0, d_n^1 \rangle$.

[9] 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 [14, 38] 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 the proof on non-malleability under chosen ciphertext post-processing does assume that the simulator has this property.

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.

9.6 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 9.1 *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 $\mathcal{A}$, $\mathcal{A}'$, etc. are all adversaries against the DDN scheme $\mathcal{S}$. Adversaries $\mathcal{B}$, $\mathcal{B}'$, etc. are all 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 $\mathrm{hist}(m)$ 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 $\mathrm{hist}(m)$. 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 $\mathcal{B}$.

Recall that in the Naor-Yung scheme, the proof of consistency allows $\mathcal{B}$ (via $\mathcal{T}$) to respond to queries from $\mathcal{A}$ during $\mathcal{A}$'s chosen-ciphertext preprocessing attack. The same idea is used here, combined with the idea in DDN Lite for controlling the choice of encryption keys, to show that the simulation phase runs smoothly.

Lemma 9.2 *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 $\zeta \neq c$. ■

Note that in Step 3 of the simulation 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 9.2, $\mathcal{B}$ can decrypt these values, and so outputs β_i, $i = 1, 2, \ldots$, together with σ.

Lemma 9.3 *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) and in the latter case they are authentic. The lemma therefore follows immediately from the definition of non-interactive zero knowledge (Definition 9.2): any difference between the two probabilities can be translated into an ability to distinguish a simulated proof from a true proof. ■

Theorem 9.1 *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 malleabilty 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 9.2, $\mathcal{B}$ can extract the β_i. Let $\beta = (\beta_1, \beta_2, \ldots)$. Let R be any probabilistic polynomial time computable relation. By Lemma 9.3, 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 $|\Pr[R(m,\beta,\sigma)] - \Pr[R(m',\beta',\sigma')]| \leq \nu(n)$, 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 $\text{hist}(m')$, 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 9.3 and the fact that $\pi'(\mathcal{A}', R) = \pi'(\mathcal{B}', R)$, implies that $|\pi(\mathcal{A}, R) - \pi'(\mathcal{A}', R)| \leq \nu(n)$. Therefore $(\mathcal{A}, R)$ cannot witness the malleability of $\mathcal{S}$. ■

Corollary 9.1 *If there exists a public-key cryptosystem semantically secure against chosen plaintext attack and if non-interactive zero-knowledge satisfying the requirements of Definition 9.2 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 [2]).

9.6.1 Remarks

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 counterexample 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 $R(m, m')$.

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 [8].

On Having $\mathcal{A}'$ 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. The full DDN scheme satisfies this strong requirement.

Practicality. For seven years after the [32] scheme was developed, no practical construction of a nm-cca-post cryptosystem was known. Finally, Cramer and Shoup [26] obtained a practical construction based on the Decisional Diffie-Hellman assumption; roughly speaking, this assumption says that the following two distributions are polynomial-time indistinguishable:

$$\{g^a \bmod p, g^b \bmod p, g^c \bmod p\}$$

and

$$\{g^a \bmod p, g^b \bmod p, g^{ab} \bmod p\}$$

where g has order q for a large prime q and $a, b, c \in_R \mathbb{Z}_q$ (see [67] for a discussion of the assumption). Unlike the two chosen-ciphertext (pre- and post-) schemes described in these notes, the Cramer-Shoup system *does not* have the property that *anyone* can determine if a ciphertext is valid – there is no NIZK proof of consistency. However, Cramer and Shoup are able to show that the information obtained by learning that a ciphertext is invalid is in some quantifiable sense "useless."

9.6.2 Public-Key Authentication

As we saw in Exercise 3.3, semantic security under cca-pre attacks is inadequate for an identification scheme based on arbitrary challenges. The following scheme is proved secure in [32].

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.

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.

10 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. In this section we present a non-malleable string commitment scheme $\mathcal{S}$ due to Dolev, Dwork, and Naor [32].

Strategic Overview. Intuitively, in order to guarantee that α and β are unrelated it is sufficient to require that that C know the value to which it is committing. To see this, suppose C knows β and C also knows that $R(\alpha, \beta)$ holds. Then C knows "something about" α, violating the semantic security of the (A, B) string commitment. Still speaking intuitively, proving possession of knowledge requires specifying a knowledge extractor, which, given access to the internal state of C, outputs β. We will define an extractor that will interact with C in order to obtain β. As in the case of proofs of knowledge, the extractor must be able to rewind C in order to perform the extraction. This introduces some technical difficulties, since C, through its "accomplice" B, has access to the (A, B) interaction: rewinding C would therefore seem to entail rewinding A. This is unacceptable: if the extractor can re-wind A then the extraction becomes a proof only that *someone* – perhaps A! – knows β, without being a proof that C knows β. To overcome this obstacle, we introduce a second, possibly malleable, commitment scheme, that is still semantically secure, with the property that there exists an *adaptor* such that

- the adversary $\mathcal{A}$ cannot distinguish between the scenario described above ((A, B) and (C, D) are running the commitment protocol $\mathcal{S}$, with A committing in conversation with B and C committing in conversation with D, and B, C are under control of the adversary) and the scenario in which:
    1. A commits using $\mathcal{S}'$
    2. B, controlled by $\mathcal{A}$, runs $\mathcal{S}$ (B may deviate from its protocol, but it "thinks" that A is running the sender part of $\mathcal{S}$, and B is "supposed" to be running the receiver part of $\mathcal{S}$)
    3. communication between A and B is manipulated by the adaptor so that A cannot tell that B is running the receiver part of $\mathcal{S}$ instead of the receiver part of $\mathcal{S}'$
- extraction can be performed on the value committed to by C in the $\mathcal{S}'$-adaptor-$\mathcal{S}$ system, *without rewinding A*.

Identities. The [32] non-malleable string commitment scheme has the property that if the players have names (from a possibly unbounded universe), then for all polynomial time computable relations R the 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. 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. One delicate issue is the question of identities. Let α and β be as above. If the players have names, then the commitment protocol guarantees that β is independent of α. The names may come from an unbounded universe. Note that there are many possibilities for names: timestamps, locations, message histories, and so on. If the players are anonymous, or the names they claim cannot be verified, then it is impossible to solve the *transparent intermedieary* problem, in which the adversary essentially acts as a clear channel between a legitimate committer and a legitimate receiver. However, the faulty sender must be completely transparent: if $\beta \neq \alpha$ then the two strings are unrelated by any relation R. Instead of committing to a value α, let A commit to $A \circ \alpha$ (read, "A says α"). In this case the only dependent value that C can commit to is again $A \circ \alpha$, which contains the "wrong" name. Note that authenticated channels are *not* assumed.

Agents and Protocols. Let (A, B) be an interactive protocol. So far, we have not distinguished between the protocols A and B and the agents or parties supposedly running these protocols. Henceforth, we distinguish between the algorithm A and the agent $\psi(A)$ that (purportedly) executes it. Thus $\psi(A)$ denotes a possibly faulty agent that is "supposed" to be running the protocol A (that is, that the non-faulty participants expect it to be running A).

Definition 10.1 (Adversarially Coordinated System) *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 commitment protocol.

Intended Values. In any string commitment protocol (A, B) for primitive $\mathcal{P}$, party $\psi(A)$ has an *intended value*: the string to which $\psi(A)$ commits. 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 a default value. The term *intended value* covers cases like this. Consider the adversarially coordinated system $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$. In an execution of this system, $\psi(A)$ has intended value $\alpha \in_R \mathcal{D}$ in its conversation with $\psi(B)$, and $\psi(C)$ has intended value β in its conversation with $\psi(D)$. If $\psi(C)$ fails to do commit – *e.g.*, fails to respond to a query, is caught cheating, or produces invalid ciphertexts – we take β to be all zeros.

On "Copying". We treat "copying" slightly differently in the context of commitment, which is interactive, than we did in the context of encryption, which is non-interactive. In the interactive setting we only ensure that if $\alpha \neq \beta$, then the two values are unrelated. We use the identities 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 exactly the same string as A does, even if it gives a different commitment (to the same value).

Interactive Non-Malleability. We now formally define the non-malleability guarantee in the interactive setting. We state the definition in terms of commitment, but it is in fact quite general (see [32]). A *relation approximator* R is a probabilistic polynomial time Turing machine taking two inputs[10] 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 setting, 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, given the use of identities, this is not the case in which the adversary "succeeds."

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 committed to without access to the commitment to α (although as always we assume that $\psi(C)$ has access to $\mathcal{D}$ and $\mathrm{hist}(\alpha)$). An *adversary simulator* for a commitment 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.

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, $\mathrm{hist}(\alpha)$, and participation in the (A, B) execution in which $\psi(A)$ commits to α, causes $\psi(C)$ to have intended value 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 $\mathrm{hist}(\alpha)$, and let $\mathcal{A}'$ have intended value 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 10.2 (Non-Malleable String Commitment) *A commitment scheme $\mathcal{S}$ is non-malleable with respect to itself under a given type of attack G, if for all*

[10] Sometimes, as in the case of encryption, we need R to take three inputs, the third being in plaintext.

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.*

10.1 The Non-Malleable String Commitment Scheme $\mathcal{S}$

It is instructive at this point to consider the protocol of Chor and Rabin [24]. They considered the "traditional" 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.

$\mathcal{S}$ uses as a building block a (possibly malleable) string commitment scheme, such as Naor's scheme (Section 6.2). 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, string commitment is considered to be a one-phase procedure. In the sequel, when we refer to the string commitment in [65], we consider only the second stage of that protocol. The scheme also requires zero-knowledge proofs.

Protocol $\mathcal{S}$ consists of two general stages. The first is a string commitment as in [65]. 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 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 [65].
- **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 **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 Step **BCK**3 is a proof of consistency; it need not be proof of knowledge in the sense of [37].

Remark 9 *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).

2. A simulation argument shows that Protocol $\mathbf{BCK}(\alpha)$ *is semantically secure. The semantic security of the committed value α in the protocol has implications for the security of selective decommitment. For example, if the x_i's are uniformly distributed, then the selective decommitment does not yield information about whether the majority of the* unopened *values have more zeros than ones; otherwise it would be possible to distinguish* $\mathbf{BCK}(0^k)$ *from* $\mathbf{BCK}(1^k)$.

It is helpful to consider each step **BCK**i as single operation. (**BCK**1 and **BCK**2 are indeed "instantaneous," in that each requires a single send, while **BCK**3, due to its interactive nature, requires more time to carry out.) We refer to an instance of **BCK** as a *triple.*

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 3 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. We say such a triple is *exposed.* 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. This is the key idea in the construction in [32].

Pursuant to Remark 9(1.), from the replies to two *different* queries in any triple one can extract the committed value. If in addition the triple is exposed, then the extraction is indeed from C alone. This is the key idea in the proof.

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

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 the stages of the two distinct instances of the **BCK** protocol are interleaved.

Using the 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. The existence of the exposed triple follows from 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 [24]. 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 [65]).
- For $j = 1$ to $|I|$
  Execute an I_j-sixplet
  Execute a $(1 - I_j)$-sixplet
  End

For simplicity, 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, the non-malleable string commitment requires invoking each **BCK**i a total of $4n^2$ times.

Suppose that two pairs of sender and receiver $(\psi(A), \psi(B))$ and $(\psi(C), \psi(D))$ are conducting $\mathcal{S}$, 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.)

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.

An examination of the possible interleavings of the two executions of $\mathcal{S}$ shows yields the following lemma (see [32] for details):

Lemma 10.1 *For any $x \neq y$ and for any two sequences σ_x and σ_y, there exists an exposed triple in σ_y.*

10.2 The Knowledge Extractor

Consider an adversarially coordinated system $\langle (A, B), (C, D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C) \rangle$ where (A, B) and (C, D) are both instances of $\mathcal{S}$. Intuitively, if $\psi(C)$ succeeds in committing to a string β, then our goal is to extract β. To achieve this we devise a somewhat different, possibly malleable, protocol called $\mathcal{S}'$ and, as stated above, we extract β from the $\mathcal{S}$'-adaptor-$\mathcal{S}$ system. 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.

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

- Commit to α (*e.g.*, using the protocol in [65]).
- 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 **BCK3** 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 10.2 *For any strategy of choosing the sixplets and for any receiver Q', the string commitment protocol $\mathcal{S}'$ is semantically secure.* ■

The adaptor will allow us to emulate to $\mathcal{A}$ (and its controlled machines) a player $\psi(A)$ 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 $\mathcal{S}$.) The adaptor that operates between P and B as follows (see Figure 4): whenever the (A, B) protocol 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. The distribution of conversations that $\psi(B)$ sees when it participates in $\mathcal{S}$ and the distribution of conversations it seeswhen it participates through the adaptor in $\mathcal{S}'$ are identical.

The extractor operates on $\psi(C)$ and $\psi(B)$ by running the $\mathcal{S}$'-adaptor-$\mathcal{S}$ system, rewinding these machines as needed in order to get answers to two different queries in an exposed triple. The extractor *never* rewinds P. The entire procedure will require P to run $m = |I|\frac{16}{(\epsilon^2 \log \epsilon)}$ sixplets. The semantic security of α (the value committed to by $\psi(P)$) is ensured by the semantic security of the protocol $\mathcal{S}'$ run by $\mathcal{P}$: this protocol is secure regardless of how the sixplets are selected. Thus, the extractor will extract the value β (if any) committed to by $\psi(C)$, while keeping α secure. Thus, intuitively, β must be idependent of α. For further details, see [32].

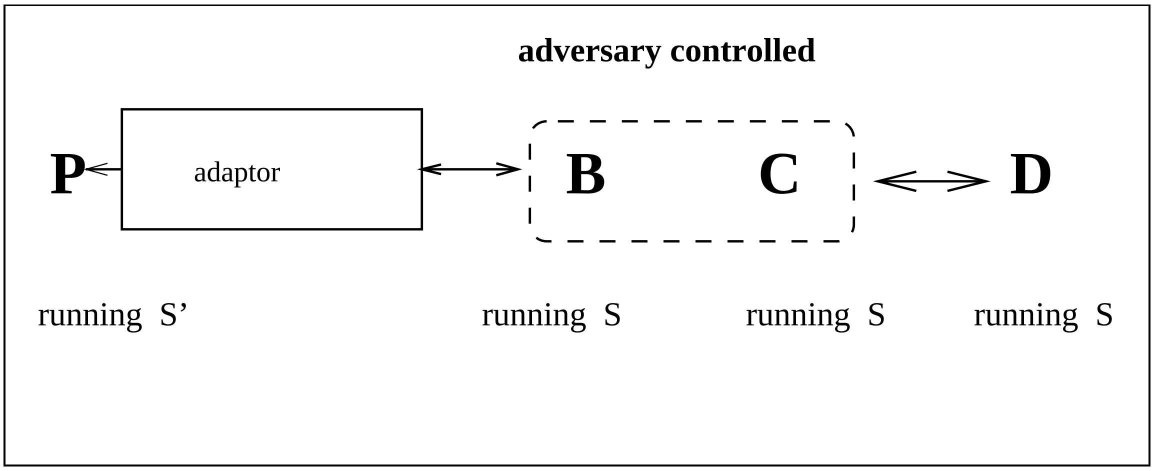

Figure 4: The $\mathcal{S}'$-adaptor-$\mathcal{S}$ system used in constructing the Extractor

Lemma 10.3 *Consider any adversarially coordinated system $\langle (A,B),(C,D),\ \mathcal{A} : \psi(B) \leftrightarrow \psi(C)\rangle$ in which $(A,B) = (C,D) = \mathcal{S}$, and suppose that the probability that $\psi(C)$ completes its part successfully is ρ. Then 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.* ■

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

Claim 10.1 *Let $\alpha \in_R \mathcal{D}$ and let β be obtained by the extractor. Then for every relation approximator R, either (1) the probability that $R(\alpha,\beta)$ outputs 1, where the probability space is over the choice of α and the internal coin flips of the machines involved, is larger than $\pi(\mathcal{A},R)$ or (2) these two probabilities are arbitrarily close.*

10.3 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 $\mathcal{A}'$ on input hist(α); we assume $\mathcal{A}'$ has 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 hist(α), 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 hist(α). 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 hist(α). 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 hist(α). Then $\pi'(\mathcal{A}',R) = \Pr[R(\alpha,\beta'')]$ since this is exactly the setting of the variables when $\mathcal{A}'$ receives as input hist(α) (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 Claim 10.1, 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 10.4 *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 (equivalent to semantic security [42, 51, 64]).*

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 $\text{hist}(\alpha_1)$, and extract ζ, the value committed to by $\psi(C)$ in the emulation.
3. Output $R(\alpha_1, \zeta)$.

If, in the emulation, $i = 1$, 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 10.1 *$|\tilde{\pi}(\mathcal{A}, R) - \pi'(\mathcal{A}', R)|$ is subpolynomial.* ∎

Theorem 10.1 *The string commitment scheme $\mathcal{S}$ is non-malleable.*

Proof: By Claim 10.1, $\pi(\mathcal{A}, R) < \tilde{\pi}(\mathcal{A}, R)$ or the two are subpolynomially close. Thus $\mathcal{A}$ is not polynomially more likely to successfully commit to a value related by R to the value committed to by $\psi(A)$ than $\mathcal{A}'$ is able to commit to a value related by R to the value committed to by $\psi(P)$. However, by Lemma 10.4, $\pi'(\mathcal{A}', R)$ 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 $\psi(P)$. ∎

Remark 10 *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 [24] 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 11 *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 9.6.1, this is shown by a hybrid argument.*

2) There is a lack of symmetry between the 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 10.3). 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 10 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* i*th bit of the identity of* $\psi(C_i)$ *equals the* i*th 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 10 cannot be iterated to obtain a constant round protocol.*

Recently, Di Crescenzo, Ishai, and Ostrovsky have constructed a *non-interactive* commitment scheme in the model of computation in which there is a common random string available to all parties (analagous to the reference string in a NIZK) [28].

References

[1] M. Ajtai, *Generating Hard Instances of Lattice Problems*, STOC'96

[2] 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.

[3] 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.

[4] L. Babai, *Trading Group Theory for Randomness*, *Proc. 17 ACM Symposium on Theory of Computing*, pp. 421–429, 1985.

[5] L. Babai and S. Moran, *Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes*, *JCSS 36*, pp. 254–276, 1988

[6] 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.

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

[8] 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.

[9] M. Blum, *How to Prove a Theorem So No One Else Can Claim It*, Proc. of the International Congress of Mathematicians, Berkeley, California, USA, 1986, pp. 1444-1451.

[10] L. Blum, M. Blum, and M. Shub, A Simple Secure Pseudo-Random Number Generator, *SIAM J. Computing*

[11] M. Blum, P. Feldman, and S. Micali, *Noninteractve Zero Knowledge and its Applications*, *Proc. 20 ACM Symposium on Theory of Computing*, 1988, pp. 103–112.

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

[13] 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.

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

[15] 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.

[16] R. Boppana and R. Hirschfeld, Pseudorandom Generators and Complexity Classes

[17] G. Brassard and C. Crepeau, *Zero-Knowledge Simulation of Boolean Circuits*, Proc. CRYPTO'86

[18] G. Brassard and C. Crepeau, *Non-Transitive Transfer of Confidence: A Perfect Zero-Knolwedge Interactive Protocol for SAT and Beyond, Proc. 27 IEEE Symposium on Foundations of Computer Science*, 1986

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

[20] J.W.S. Cassels, *An Introduction to the Geometry of Numbers*, Springer, 1959

[21] D. Chaum, *Demonsrating that a Public Predicate can be Satisfied Without Revealing Any Information About How*, Proc. CRYPTO'86.

[22] D. Chaum, J.H. Evertse, J. van de Graaf, and R. Peralta, *Demonstrating Possession of a Discrete Logarithm without Revealing It*, Proc. CRYPTO'86

[23] D. Chaum, E. van Heijst and B. Pfitzmann, *Cryptographically strong undeniable signatures, unconditionally secure for the signer*, Advances in Cryptology - Crypto '91, Lecture Notes in Computer Science 576, Springer Verlag, 1992, pp. 470–484.

[24] 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.

[25] D. Coppersmith, *Finding a Small Root of a Univariate Modular Equation, Proc. EUROCRYPT'96*, 1996.

[26] 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.

[27] 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.

[28] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky, *Non-Interactive and Non-Malleable Commitment, Proc. 30 ACM Symposium on Theory of Computing*, 1998.

[29] 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.

[30] W. Diffie and M. Hellman, *New Directions in Cryptography* , IEEE Trans. on Information Theory, vol. IT-22, 6 (1976), pp. 644-654.

[31]

[32] D. Dolev, C. Dwork, and M. Naor, *Non-Malleable Cryptography*, to appear, *SiComp*. Preliminary version in STOC'91, full version to appear, *SIAM J. Computing*.

[33] C. Dwork, *Positive Applications of Lattices to Cryptography*, Proc. Mathematical Foundations of Computer Science '97, LNCS.

[34] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer, *Magic Functions, manuscript in preparation*, 1999

[35] *Zero Knowledge Proofs of Knowledge in 2 Rounds*, Advances in Cryptology – *CRYPTO'89 Proceedings, Springer-Verlag 435*, pp. 526–544.

[36] 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.

[37] 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).

[38] 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.

[39] A. Fiat and A. Shamir, *How to Prove ourself: Practical Solutions to Identification and Signature Problems*, *Proceedings of CRYPTO'86, Springer-Verlag LNCS 263* 1987, pp. 186–194.

[40] O. Goldreich, Foundations of Cryptography, Class Notes, 1989.

[41] O. Goldreich, Fragments of a Book, http://theory.lcs.mit.edu/ oded/frag.html,

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

[43] 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).

[44] O. Goldreich, S. Goldwasser, and S. Halevi, Collision-Free Hashing from Lattice Problems, *Electronic Colloquium on Computational Complexity TR96-042*, http://www.eccc.uni-trier.de/eccc-local/Lists/TR-1996.html

[45] O. Goldreich, S. Goldwasser, and S. Micali, How to Construct Random Functions, *Journal of the Association for Computing Machinery*3(4), pp. 792–807, 1986.

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

[47] 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.

[48] O. Goldreich and S. Micali, The weakest pseudo-random bit generator implies the strongest one, manuscript in preparation, 1984

[49] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that Yield Nothing but Their Validity or All Languages in NP have Zero-Knowledge Proof Systems*, *JACM 38*(1), pp. 691–729, 1991

[50] O. Goldreich and Y. Oren, *Definitions and Properties of Zero-Knolwedge Proof Systems*, *J. Cryptology 7*(1), pp. 1–32, 1994

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

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

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

[54] M. Grötschel, Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, Algorithms and Combinatorics 2, 1988

[55] P.M. Gruber, C.G. Lekkerkerker, *Geometry of Numbers*, North-Holland, 1987

[56]

[57] R. Impagliazzo, L. Levin, and L. Luby, *Pseudo-random generation from One-Way Functions*, *Proc. 21 ACM Symposium on Theory of Computing*, 1989

[58] R. Impagliazzo and M. Naor, *Efficient Cryptographic Schemes Provably as Secure as Subset Sum*, *J. Cryptology 9*, pp. 199–216, 1996

[59] R. Impagliazzo and M. Yung, *Direct Minimum-Knowledge Computations*, Proc. *CRYPTO'87*, LNCS 293, Springer Verlag, pp. 40–51, 1988

[60] R. Impagliazzo and D. Zuckerman, *How to Recycle Random Bits*, *Proc. 30 IEEE Symposium on Foundations of Computer Science*, pp. 248 – 253, 1989

[61] J. Kilian and E. Petrank, *An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions*, Ellectronic Colloquium on Computational Complexity (ECCC)(038), 1995

[62] J. Kilian, E. Petrank, and C. Rackoff STOC'98

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

[64] 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.

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

[66] M. Naor and O. Reingold, Synthesizers and Their Application to the Parallel Construction of Pseudo-random Functions, JCSS

[67] 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.

[68] 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.

[69] 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.

[70] M. O. Rabin *Fingerprinting by Random Polynomials* , Harvard University, TR-15-81, 1981.

[71] C. Rackoff, unpublished.

[72] 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.

[73] 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.

[74] M. Santha and U. V. Vazirani, *Generating Quasi-random Sequences from Slightly-random Sources*, Proc. 25th Symposium on the Theory of Computing, 1984, pp. 434-440.

[75] A. Shamir, *IP=PSPACE, Proc. 22 ACM Symposium on Theory of Computing*, pp. 11–15, 1990

[76] S. Trilling, Some Implications of Complexity Theory on Pseudo-Random Bit Generation, S.M. Thesis, Massachusetts Institute of Technology, 1985.

[77] M. N. Wegman and J. L. Carter, *New Hash Functions and Their Use in Authentication and Set Equality*, Journal of Computer and System Sciences 22, pp. 265-279 (1981).

[78] A. Yao, Theory and Applications of Trapdoor Functions, Proc. *Proc. 23 IEEE Symposium on Foundations of Computer Science*, pp. 80–91, 1982