An Introduction to Modern Cryptography: Lecture Notes for ECS 289A

Course taught by Phillip Rogaway
Notes prepared by Kevin Murphy

12 March 1995

Contents

1 Introduction

In classical cryptography, one takes a real-word problem, such as the need to ensure privacy, proposes a solution (a protocol), attempts to break the protocol by using cryptanalysis, and fixes any bugs. This process is repeated until no more bugs are found. Unfortunately, this is not a guarantee that a bug won't be found later, and the history of cryptography has shown that many apparently sound protocols were in fact flawed. What we would like is to *prove* that the protocol is correct.

Modern cryptography, also known as provable security, started with the paper of Goldwasser and Micali [GM84]. It involves formally defining the goal, devising a protocol, and proving that the protocol satisfies the goal. Typically it is necessary to make complexity-theoretic assumptions, such as the existence of one way functions (which is equivalent to assuming $P \neq NP$).

Exact security demands that we make exact statements about how good our protocols are in terms of the "security parameter" (which controls such things as the running time and key length): asymptotic statements, such as "the probability of error is negligible", are not sufficient, since the values of the constants hidden in this way are very important in practical applications.

Most of the security literature is of the classical kind, and the comparitively few papers in the provable security tradition tend to be asymptotic analyses of asymmetric protocols. We redress the balance here by providing an introduction to exact analyses, concentrating on symmetric protocols.

1.1 Intended audience

These notes are based around a $12 \times 3 = 36$ hour lecture course taught to graduate students in the computer science department of U. C. Davis during Winter Quarter 1995. However, anyone with a basic knowledge of probability theory, computation theory and complexity theory should be able to understand them; no knowledge of number theory or applied cryptography is assumed.

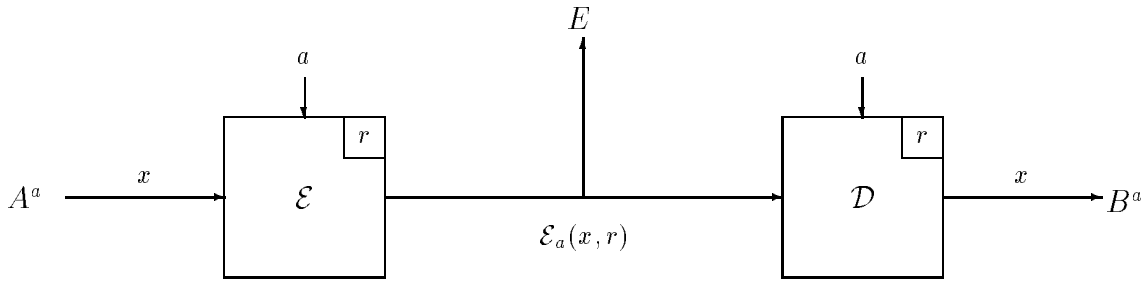

Figure 1: One view of symmetric encryption. Adversary E witnesses the encryptions of messages x. Each time a message x is encrypted, a fresh random string r is selected to encrypt it.

2 Symmetric encryption

We begin our study of cryptography by looking at its oldest problem: *symmetric encryption.* The name "symmetric" stems from the key setup: both the sender, A (for Alice), and the receiver, B (for Bob), share the *same* piece of information, a secret key a. The adversary, E (for Eve, or Enemy, or Eavesdropper), lacks this key.

A descriptive picture is show in Figure 1. We indicate the key a which A and B share by writing it as a superscript to each party's name. The encryption function is $\mathcal{E}$ and the decryption function is $\mathcal{D}$. As we shall see, it will be important that $\mathcal{E}$ be probabilistic (or stateful; you can ignore this possibility for now). Though a is chosen just once, when A and B want to communicate, each time A wants to send B some *plaintext* (or *cleartext*), x, she chooses a fresh random string, r, and then transmits the *ciphertext* $y = \mathcal{E}_a(x, r)$. The (deterministic) function $\mathcal{D}$ must be able to recover x from a and y.

The picture of Figure 1 would seem to indicate that E is a a completely passive eavesdropper. But in many real-life settings E has influence over what messages may be encrypted. Also, E may have considerable *a priori* information about what plaintext a given ciphertext corresponds to. We formalize a worst-case scenario in these regards, as depicted in Figure 2. Now E can choose each message which is to be encrypted, give it to her *oracle* $\mathcal{E}$, and get back a ciphertext which she may think about before choosing the next plaintext to encrypt. This attack scenario is sometimes called a *chosen plaintext attack.* We will define secure protocols as ones which are resistant to such asttacks.

We can think of stronger notions of security, such as resistance to *chosen ciphertext attack*, where the adversary can see the plaintext corresponding to ciphertexts of her choice. Or we might demand that an adversary always know the plaintext for any given ciphertext that she forges (see [BR94]). We shall not consider these stronger definitions in these notes. In the section on message authentication, we will consider the stronger idea of non-malleable encryption schemes, however.

2.1 A definition of secure encryption

What is a good encryption? One's first attempt might be along the lines: "given $\mathcal{E}_a(x)$, you can't recover x." Such intuition can certainly be lifted to a definition, but it would be *too weak*: maybe an adversary can't recover x, but she can recover some particular bit of x. Or, maybe she can't recover any particular bit of x, but she can recover some interesting function of them, like their XOR. And so forth. We want our

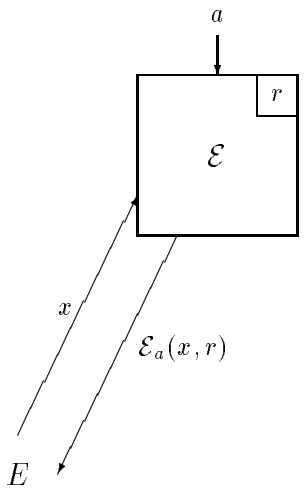

Figure 2: Another view of symmetric encryption. Adversary E is given an oracle which encrypts strings, modeling her potential ability to influence which strings are encrypted.

encryption scheme to leak no partial information about x. This idea — properly formalized — is what we will call *semantic security.*

But in this phrase "properly formalized" we hide a lot. As one immediate concern, there seems to be some information which is inherently "leaked" by $\mathcal{E}_a(x)$. One such piece of information is the existence of the message x; a second piece of information is its length; and a third piece of information is an encryption of the message under the (secret) key a. Ultimately, we deal with the first two concerns by saying that it is not the function of *encryption* to hide a message's existence or its length (that is the business of other techniques); and we will deal with the third objection by saying that $\mathcal{E}_a(x,r)$ is not really information about x — information about x is realized by some deterministic function f applied to x.

Goldwasser and Micali [GM84] first captured the intuition above (in the domain of asymmetric encryption), defining and showing how to achieve semantic security (under a complexity theoretic assumption). As they say it, an encryption scheme is "semantically secure" if *whatever is efficiently computable about the cleartext given the cyphertext, is also efficiently computable without the cyphertext.*

Though strong and compelling, there are other nice definitions for an encryption scheme to be secure, which are easier to use in proofs. We now describe one of these. We imagine the adversary E with her oracle for $\mathcal{E}_a(\cdot)$, as in Figure 2. Suppose that $\mathcal{E}$ hides the plaintext so well that for any adversary E, E can not tell if what she is getting back from the oracle is really an encryption of x, or if instead it had been an encryption of $0^{|x|}$—a fixed string of the same length. We will call this notion *security in the sense of indistinguishability,* since equal length strings have encryptions which are indistinguishable to the adversary. This notion is an adaptation of what call Goldwasser and Micali called "polynomial security."

In the remainder of this section we give formal definitions of security in the sense of indistinguishability and of semantic security. We will then show that the former implies the latter; as an **Exercise**, the reader might like to prove the converse, thus establishing that the two definitions are equivalent. We go on to exhibit some examples of secure encryption schemes, and consider the objects on which they are based (pseudo-random functions) in some detail.

2.1.1 A symmetric encryption scheme as a syntactic object

A (probabilistic, symmetric) *encryption scheme* Π is a three-tuple of string-valued functions, $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$, the first and third of which are probabilistic. Algorithm $\mathcal{E}$ is called the *encryption* algorithm; $\mathcal{D}$ is the *decryption* algorithm; and $\mathcal{K}$ is the *key generator*. With $\mathcal{E}$ understood, the domain of $\mathcal{E}$ is denoted by Dom. Where r is an (infinite) sequence of random coins, we write $\mathcal{E}_a(x, r)$ for (the string output by) $\mathcal{E}(a, x, r)$, and $\mathcal{D}_a(y)$ for (the string output by) $\mathcal{D}(a, r)$. We let $\mathcal{E}_a(x)$ be probability space corresponding to $\mathcal{E}_a(x, r)$ when r is sampled uniformly at random. We call $K =$ support $\mathcal{K}$ the *key space*. We demand the following of any encryption scheme: that for all $a \in K$, $x \in$ Dom, and $r \in \{0,1\}^\infty$,

$$\mathcal{D}_a(\mathcal{E}_a(x, r)) = x.$$

We further demand that $\mathcal{D}_a(y)$ be the distinguished value Undefined when $y \neq \mathcal{E}_a(x, r)$ for any $x \in$ Dom, $r \in \{0,1\}^\infty$.

For an encryption scheme to be useful, $\mathcal{E}$, $\mathcal{D}$, and $\mathcal{K}$ should be efficiently computable functions.

We will also consider *stateful* encryption schemes. In this case, $\mathcal{E}$ takes arguments (a, x, s, r), where $s \in \{0,1\}^*$ is called the *state*, and $\mathcal{E}$ computes from this tuple a pair of strings (y, s'), which we call the *ciphertext* and the *revised state,* respectively.

We emphasize that the above definition provides syntax—no notion of security. For example, $\mathcal{E}_a(x, r) = x$ is valid encryption function, even though it doesn't hide anything about x at all.

2.1.2 Definition of a (t, q, μ, ϵ)-break of an encryption scheme

An *adversary* E is simply an algorithm. To simplify the proofs, we will assume this algorithm can be non-uniform, that is, can be given an associated infinite string, α. Usually we will not explicitly indicate α. E is given access to an oracle: E writes a string on a special *query tape,* enters a distinguished state, and then receives an answer (which replaces the query) in unit time.

Definition 2.1 *Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an encryption scheme. Adversary E is said to (t, q, μ, ϵ)-*break *Π if*

$$\begin{aligned} \mathrm{Adv}_E^{x, 0^{|x|}} &\stackrel{\text{def}}{=} \Pr\left[a \leftarrow \mathcal{K} :\ E^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[a \leftarrow \mathcal{K} :\ E^{\mathcal{E}_a(0^{|\cdot|})} = 1\right] \\ &\geq \epsilon \end{aligned}$$

and, in the experiments above, E makes at most q queries, asking for a total of at most μ bits, and runs in at most t steps.

The notation $\Pr[A;\ B;\ \ldots :\ E]$ denotes the probability of event E occuring when we perform an experiment consisting of actions A, B, ...in order. The action $a \leftarrow \mathcal{K}$ denotes choosing a key a from the probability space induced by $\mathcal{K}$.

The quantity $\mathrm{Adv}_E^{A,B}$ is called the adversary E's *advantage* in distinguishing A from B. For all the "types" of advantage we define, we will often omit the superscript. Remember that an adversary ϵ-breaks a scheme if her advantage is *at least* ϵ.

In the above definition, we assume that E outputs 1 if she thinks she is given the "true" oracle, and 0 if she thinks she is given the "zero" oracle, so her advantage is always non-negative. If E actually adopts the reverse convention, we can specify in the hint α that we should flip the output bit. In the uniform case,

for which there is no α, it is conventional to consider the absolute value of the difference in probabilities to overcome this problem.

We emphasize the importance of keeping t and q separate: in practice, oracle queries correspond to observations or interaction with a system whose overall structure often severely limits q (e.g., the system might limit the amount of plaintext encrypted before the key is changed); but t corresponds to off-line computation by the adversary, and so is much less under the good guy's control. We introduce the parameter μ so that we can make explicit how long our queries are.

In the case that Π is stateful we must modify the definition: the oracle, currently in state s and receiving a query of x, samples $(y, s') \leftarrow \mathcal{E}_a(x, s)$, then returns y and updates the current state to s'. The initial state is taken to be the empty string.

2.1.3 Families of encryption schemes

It is often necessary to think about a parameterized *family* of encryption schemes: effectively, there is a different $(\mathcal{E}, \mathcal{D}, \mathcal{K})$ for every *security parameter*, k. We model this by providing $\mathcal{K}$ with a unary *security parameter*, k, denoted $\mathcal{K}(1^k)$. We use unary to ensure that if $\mathcal{K}$ is polynomial in its argument, it is polynomial in k; if $\mathcal{K}$ was polynomial in its argument, but its argument was the binary representation of k, $\mathcal{K}$ would be exponential in the size of k itself.

Intuitively, the security parameter allows us to "tune" the degree of security: a larger k might result in a longer key, which might result in a more secure, but slower, system. However, the main reason for introducing k is technical: to allow us to make meaningful asymptotic statements about the various functions. We now have a different domain, Dom_k, for each security parameter k.

Definition 2.2 *Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ be a parameterized encryption scheme. Adversary E is said to $(t(k), q(k), \mu(k), \epsilon(k))$-*break $\Pi[k]$ *if*

$$\begin{aligned} \mathrm{Adv}_E^{\$, 0^{|x|}} &\stackrel{\text{def}}{=} \Pr\left[a \leftarrow \mathcal{K}(1^k) : E^{\mathcal{E}_a(\cdot)}(1^k) = 1\right] - \Pr\left[a \leftarrow \mathcal{K}(1^k) : E^{\mathcal{E}_a(0^{|\cdot|})}(1^k) = 1\right] \\ &\geq \epsilon \end{aligned}$$

and, in the experiments above, E makes at most $q(k)$ queries, asking for a total of at most $\mu(k)$ bits, and runs in at most $t(k)$ steps.

For notational simplicity, we usually omit explicit indication of the security parameter k.

2.1.4 A complexity-theoretic notion of the security of a symmetric encryption scheme

A function $\epsilon(k)$ is *negligible* if for all $c > 0$ there exists an N such that for all $n \geq N$, $\epsilon(n) \leq n^{-c}$. This can be written more succinctly as "$\epsilon(k) \in k^{-\omega(1)}$."

We can make a complexity-theoretic definition of security as follows: A (parameterized) encryption scheme $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ is said to be *secure* if $\mathcal{E}$, $\mathcal{D}$ and $\mathcal{K}$ are Probabilistic Polynomial Time (PPT) algorithms; and for any adversary E which $(t(k), q(k), \mu(k), \epsilon(k))$-breaks Π_k, if $t(k)$ is a polynomial then $\epsilon(k)$ is negligible.

Although this definition is widely used in the literature, we shall usually be interested in more precise statements of a scheme's security. This is because, for practical systems, the values of the constants c and N are very important, and should not be "swept under the asymptotic rug".

2.1.5 Definition of a (t, q, μ, ϵ)-break of semantic security

For a function $f : \{0,1\}^* \to \{0,1\}^*$ and a distribution on the messages $\mathcal{M}$, let $P^*_{\mathcal{M},f}$ be $\max_y\{\Pr[x \leftarrow \mathcal{M} : f(x) = y]\}$ That is, $P^*_{\mathcal{M},f}$ is the probability of the most likely image of f, which you can think of as the most likely ciphertext to be output. A (nonuniform) adversary can, when $x \leftarrow \mathcal{M}$, guess $f(x)$ without looking at *any* information associated to x and still be right with probability $P^*_{\mathcal{M},f}$. We want to say that, in a "good" encryption scheme, the adversary can not do much better. We formalize this as follows:

Definition 2.3 *Let $\Pi = (\mathcal{E}, \mathcal{D}, \mathcal{K})$ be an encryption scheme, let $f : \{0,1\}^* \to \{0,1\}^*$ be a function, and let $\mathcal{M}$ be a distribution on messages such that $u, v \in$ support $\mathcal{M} \Rightarrow |u| = |v|$. Adversary E is said to* (t, q, μ, ϵ)-break the semantic security of Π with respect to $\mathcal{M}$ and f *if*

$$\begin{aligned} \mathrm{Adv}_E^{\mathcal{M},f} &\stackrel{\text{def}}{=} \Pr\left[a \leftarrow \mathcal{K};\ x \leftarrow \mathcal{M};\ r \leftarrow \{0,1\}^\infty;\ y \leftarrow \mathcal{E}_a(x,r) :\ E^{\mathcal{E}_a(\cdot)}(y) = f(x)\right] - P^*_{\mathcal{M},f} \\ &\geq \epsilon \end{aligned}$$

and, in the experiment above, E runs in at most t steps and makes at most q queries, their total length being at most μ.

2.1.6 A complexity-theoretic notion of semantic security

As usual, we can define the complexity-theoretic analog of the definition above, saying that Π is *semantically secure* if: $\mathcal{E}$ and $\mathcal{K}(1^k)$ are PPT; $\mathcal{D}$ is polynomial time; and for any function f, and family of distributions $\mathcal{M}_k$ having polynomial support of some fixed length $poly(k)$, if E $(t(k), q(k), \epsilon(k))$-breaks the semantic security of $\Pi[k]$ with respect to $\mathcal{M}$ and f for some polynomial $t(k)$, then $\epsilon(k)$ is negligible.

2.1.7 Security in the sense of indistinguishability implies semantic security

Theorem 2.4 *Let Π be an encryption scheme, let $f : \{0,1\}^* \to \{0,1\}^*$ be a function, and let $\mathcal{M}$ be a distribution on messages with* support $\mathcal{M} \subseteq$ Dom $\subseteq \{0,1\}^n$. *Suppose an adversary can $(t,\ q,\ \mu, \epsilon)$-break the semantic security of Π with respect to $\mathcal{M}$ and f. Then there an adversary which can $(t + \lambda_1(\mu + n + \max_{x \in \{0,1\}^n}\{|f(x)|\}) + \lambda_0,\ q + 1,\ \mu, \epsilon)$-break Π (for absolute constants λ_1 and λ_0 which depend only on the model of computation that we use).*

Proof: Suppose E_* can $(t,\ q,\ \mu, \epsilon)$-break the semantic security of Π with respect to $\mathcal{M}$ and f. Imagine for the moment that it is possible to sample in $\mathcal{M}$ and compute f. (We will not need these abilities; at the end, we will remove this assumption using nonuniformity.) We construct adversary E as follows. Adversary E will begin by choosing a random sample x from $\mathcal{M}$ and then applying its oracle h to x, thereby computing a string $y = h(x)$. Now E will simulate the behavior of E_* running on input y. Any time E_* makes an oracle query of some string u we have E make a query of the *same* string u, getting some answer w in return. We have E return to E_* this string w. When, finally, E_* would output a string, z, and halt, adversary E checks if $z = f(x)$, and if so, E outputs 1; otherwise, E outputs 0.

Suppose that E's oracle is answered according to $\mathcal{E}_a(\cdot)$. Then the environment which E provides for $\mathcal{E}_*$ is *identical* to the one defining how well E_* finds $f(x)$:

$$\Pr\left[x \leftarrow \mathcal{M};\ y \leftarrow \mathcal{E}_a(x) :\ E_*^{\mathcal{E}_a(\cdot)}(y) = f(x)\right] \quad = \quad \Pr\left[E^{\mathcal{E}_a(\cdot)} = 1\right]. \tag{1}$$

On the other hand, if E's oracle is answered according to $\mathcal{E}_a(0^{|\cdot|})$, then

$$\begin{aligned}\Pr\left[x \leftarrow \mathcal{M};\ y \leftarrow \mathcal{E}_a(0^{|x|}) :\ E_*^{\mathcal{E}_a(0^{|\cdot|})}(y) = f(x)\right] &= \Pr\left[E^{\mathcal{E}_a(0^{|\cdot|})} = 1\right] \\ &\leq P^*_{\mathcal{M},f}\end{aligned}$$

because the information E_* receives (both y and the answers from its oracle) is uncorrelated with x. Multiplying by -1,

$$-\Pr\left[E^{\mathcal{E}_a(0^{|\cdot|})} = 1\right] \geq -P^*_{\mathcal{M},f}\ . \tag{2}$$

Now we are given that

$$\Pr\left[x \leftarrow \mathcal{M};\ y \leftarrow \mathcal{E}_a(x) :\ E_*^{\mathcal{E}_a(\cdot)}(y) = f(x)\right] \geq \epsilon + P^*_{\mathcal{M},f}$$

so, from, (1),

$$\Pr\left[E^{\mathcal{E}_a(\cdot)} = 1\right] \geq \epsilon + P^*_{\mathcal{M},f}\ .$$

Combining this with (2) we have

$$\Pr\left[E^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_a(0^{|\cdot|})} = 1\right] \geq \epsilon + P^*_{\mathcal{M},f} - P^*_{\mathcal{M},f}\ ,$$

that is, $\mathrm{Adv}_E^{x,0^{|x|}} \geq \epsilon$.

To remove the technical conditions we assumed about E, realize that $\mathrm{Adv}_E^{x,0^{|x|}}$ is actually the $\mathcal{M}$-weighted sum of the advantages for particular x-values, $\mathrm{Adv}_E = \sum_{x \leftarrow \mathcal{M}} \Pr[x]\mathrm{Adv}_{E,x}$. Thus we can choose a particular x which achieves at least the mean of the advantages. Then, use the nonuniformity of our adversary to "hardwire" into her the particular x that we have selected, as well as its associated $f(x)$-value. ■

2.2 Symmetric encryption using a one-time pad

How can we achieve secure symmetric encryption? One famous method, which was invented by Vernam in 1917, is as follows: A performs a bit-wise xor (exclusive-or) of x with a and sends the result to B; to recover the plaintext, B performs the same operation (since xor is self-inverse). The advantage of this scheme is that it is secure in an *information theoretic* sense; that is, without needing to make any complexity assumptions such as $P \neq NP$, it is possible to show that an adversary cannot recover anything about x given $y = \mathcal{E}_a(x)$. The proof is simple: any message x could have yielded y if the key was equal to $x \oplus y$. The disadvantage is that A and B must share a key a which is as long as the message they want to send; and each key can only be used for one message. In the rest of these notes, we shall therefore content ourselves with encryption schemes which are secure in the computational complexity sense defined above.

2.3 Symmetric encryption using a block cipher

Most practical encryption schemes are based around a block cipher f such as DES (these and other terms will be explained presently). If we make certain assumptions about how good f is, we can then ask how good our encryption scheme built on f is. In the following sections, we will show that, if you can't break f better than ϵ', then you can't break encryption schemes based on f better than $\epsilon > \epsilon'$. (By "breaking f", we mean distinguishing it from a random function.) We will do this by proving theorems in their contrapositive form, namely: "If you can break an encryption scheme based on f better than ϵ, then you can distinguish f from a random function better than $\epsilon' < \epsilon$."

2.3.1 Block ciphers

A *block cipher* with key length κ and block length ℓ is a function $f : \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^\ell$. We write $f_a(x)$ for $f(a,x)$ and we demand that, for each a, f_a is a permutation on $\{0,1\}^\ell$. For a block cipher to be useful, both f_a and f_a^{-1} should be efficiently computable functions. We shall also call a block cipher a (finite) *pseudorandom permutation* (PRP).

An example of a block ciper is DES, the Data Encryption Standard defined by the US government. It is a function mapping 64-bit strings to 64-bit strings under control of a 56-bit key. Now the number of possible permutations on 64-bit strings is $2^{64}!$ which is much greater than the number of permutations specifiable by DES, namely 2^{56}. The question naturally arises as to how "good" a block cipher DES actually is.

By a "good" block cipher we mean one that is indistinguishable from a random permutation. We quantize this as follows. Let $\mathcal{P}_{\ell\to\ell}$ be the set of all permutations on $\{0,1\}^\ell$. The success of an adversary in attacking a block cipher is quantized in the following way:

Definition 2.5 *Let f be a PRP with key length κ and block length ℓ. Adversary D is said to* (t,q,ϵ)-distinguish f from a random permutation *if*

$$\begin{aligned}\mathrm{Adv}_D^{f,\pi} &\stackrel{\text{def}}{=} \Pr\left[a \leftarrow \{0,1\}^\kappa : D^{f_a(\cdot)} = 1\right] - \Pr\left[\pi \leftarrow \mathcal{P}_{\ell\to\ell} : D^{\pi(\cdot)} = 1\right] \\ &\geq \epsilon\end{aligned}$$

and, in the experiments above, D makes at most q queries and runs in at most t steps.

How good is DES according to the above definition? According to [BS93, MAT93, MAT94], no one currently knows how to break DES better than $q = 2^{20}$, $\epsilon = 4 * t/2^{56} + 2^{-20}$.

2.3.2 A complexity-theoretic notion for the security of block ciphers

One may imagine a family of PRPs: for each security parameter k there is a block cipher $f[k]$ which is a permutation on key length $\kappa(k)$ and block length $\ell(k)$. The functions κ and ℓ are polynomials associated to the the PRP. The definition of the success of an adversary in attacking a block cipher function immediately "lifts" to take account of the family of functions involved:

Definition 2.6 *Let f be a finite PRP family with key length $\kappa(k)$ and block length $\ell(k)$. Adversary D is said to* $(t(k),q(k),\epsilon(k))$-distinguish $f[k]$ from a random permutation *if*

$$\begin{aligned}\mathrm{Adv}_D^{f,\pi}(k)) &\stackrel{\text{def}}{=} \Pr\left[a \leftarrow \{0,1\}^{\kappa(k)} : D^{f_a(\cdot)}(1^k) = 1\right] - \Pr\left[\pi \leftarrow \mathcal{P}_{\ell(k)\to\ell(k)} : D^{\pi(\cdot)}(1^k) = 1\right] \\ &\geq \epsilon(k)\end{aligned}$$

and, in the experiments above, D makes at most $q(k)$ queries and runs in at most $t(k)$ steps.

As before, one can lift the definition to a complexity-theoretic one. A finite PRP family f is said to be *secure* if some polynomial time algorithm M computes $f[1^k]_a(x)$ on input $(1^k,a,x)$; some polynomial time algorithm M^{-1} computes $f^{-1}[1^k]_a(x)$ on input $(1^k,a,x)$; and, for every polynomial time adversary E which $(t(k),q(k),\epsilon(k))$-distinguishes $f[k]$ from a random permutation, if $t(k)$ is polynomial then $\epsilon(k)$ is negligible.

2.3.3 Pseudorandom functions versus pseudorandom permutations

Sometimes we will think of a block cipher like DES as a general (finite) *pseudorandom function* (PRF) $f : \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^\ell$.[1] This is slightly inaccurate, since the space of permutations is much smaller than the space of functions. However, as long as we encrypt less than $O(2^{k/2})$ messages, the chance of an adversary discovering that we are using a PRP and not a general PRF is negligible, since the chance of a collision is small. (A collision occurs when two points in the domain are mapped to the same image; this cannot happen with a permutation.) Calculating the probability of such a collision is similar to the birthday paradox (which states that it is only necessary to have as few as 23 people before the probability that they share a birthday (i.e., their birthdays collide) is greater than 0.5). The following sections formalise these ideas.

We extend the notion of a "good" PRP family to the case of a PRF family in the obvious way: a good PRF is one which is indistinguishable from a random *function* $\rho \in \mathcal{R}_{\ell\to\ell}$ (as opposed to a random permutation $\pi \in \mathcal{P}_{\ell\to\ell}$). Similarly, a "secure" PRF is one for which the chance of distinguishing, $\epsilon(k)$, is negligible.

Theorem 2.7 *Suppose you can (t,q,ϵ)-distinguish $f : \{0,1\}^\kappa \times \{0,1\}^\ell \to \{0,1\}^\ell$ from a random function. Then you can $(t,q,\epsilon - q^2 2^{-\ell-1})$-distinguish f from a random permutation.*

Proof: We assume there is an algorithm D' running in time t and making q queries such that

$$\epsilon \leq \Pr\left[D'^f = 1\right] - \Pr\left[D'^\rho = 1\right] = \mathrm{Adv}_{D'}(k).$$

To keep the inequalities pointing the same way (in particular, because Equation 5 is an upper bound), it will be useful to consider an algorithm D which inverts the output of D'.

$$\begin{aligned}\mathrm{Adv}_D(k) &= \Pr\left[D^f = 1\right] - \Pr\left[D^\rho = 1\right] \\ &= (1 - \Pr\left[D'^f = 1\right]) - (1 - \Pr\left[D'^\rho = 1\right]) \\ &= \Pr\left[D^\rho = 1\right] - \Pr\left[D^f = 1\right]\end{aligned}$$

In the experiment of the left hand side, where D is running with random function ρ, let B be the ("bad") event that D makes two distinct queries, x_1 and x_2, which return the same value, $\rho(x_1) = \rho(x_2)$. We have then

$$\begin{aligned}\epsilon &\leq \Pr\left[D^\rho = 1|\overline{\mathrm{B}}\right]\Pr\left[\overline{\mathrm{B}}\right] + \Pr\left[D^\rho = 1|\mathrm{B}\right]\Pr\left[\mathrm{B}\right] - \Pr\left[D^f = 1\right] \\ &\leq \Pr\left[D^\rho = 1|\overline{\mathrm{B}}\right] + \Pr \mathrm{B} - \Pr\left[D^f = 1\right]. \qquad (3)\end{aligned}$$

Now it is easy to verify that

$$\Pr\left[D^\rho = 1|\overline{\mathrm{B}}\right] = \Pr\left[D^\pi = 1\right], \qquad (4)$$

because D "sees" an identical distribution in both cases: a random distinct response in answer to each query. Also,

$$\begin{aligned}\Pr\left[\mathrm{B}\right] &\leq \sum_{i=1}^{q} \frac{i-1}{2^\ell} \\ &\leq \frac{q(q-1)}{2} 2^{-\ell}, \qquad (5)\end{aligned}$$

[1] Note that, for a PRF to be useful, f_a should be an efficiently computable function; however, we do not require that it have an efficiently computable inverse (consider how UNIX encrypts passwords).

because $\Pr[\mathrm{B}]$ is upper-bounded by the likelihood that q randomly selected numbers drawn from $\{0, \ldots, 2^\ell - 1\}$ are not all distinct. Putting (4) and (5) into (3) gives

$$\epsilon \quad \leq \quad \Pr[D^\pi = 1] + q^2 2^{-\ell-1} - \Pr[D^f = 1],$$

so

$$\epsilon - q^2 2^{\ell-1} \quad \leq \quad \Pr[D^\pi = 1] - \Pr[D^f = 1].$$

Forming D' by reversing when D outputs "0" and "1" (to put the distinguisher in line with our conventions) we have that $\mathrm{Adv}_{D'} \geq \epsilon - q^2 2^{-\ell-1}$, as desired. ■

2.3.4 There is no secure encryption which is deterministic and stateless

Suppose we want to encrypt a message $x \in \{0,1\}^\ell$ and we have a block cipher f acting on ℓ-bit blocks and using a κ-bit key. Is $\mathcal{E}_a(x) = f_a(x)$ a secure encryption of x? The answer is certainly *no*: with two distinct queries, x_1 and x_2, we can tell if we are getting back the encryption of 0's or the encryptions of what we ask for: just look to see if the answers are the same. By the same reasoning, there is, in general, no stateless deterministic encryption.

(Nonetheless, one could come up with a weaker notion tailored to deterministic encryption: intuitively, all that should be leaked is whether or not the enciphered string is a repetition of an earlier enciphered string and, if so, which one.)

2.3.5 Two Vernam ciphers based on a block cipher

Recall that the problem with the one-time pad was the need to share such a long key. Suppose instead that A and B share a PRF (or rather, a key a which will index into a family of PRFs $\{f_a\}$). Then we can *generate* a one-time pad as follows: choose a random ℓ-bit r and use $f_a(r \oplus \langle 1 \rangle), f_a(r \oplus \langle 2 \rangle), \ldots$ as our pad (in which case A needs to send r to B as well). So the encryption of $x = x_1 \cdots x_m$, where $|x_i| = \ell$ and $m \leq 2^\ell$, is

$$(r,\ f_a(r \oplus \langle 1 \rangle) \oplus x_1) \ \|\ \cdots \ \|\ f_a(r \oplus \langle m \rangle) \oplus x_m)).$$

Here we are letting $\langle i \rangle$ denote the binary representation of the number i mod 2^ℓ, treated as an ℓ-bit string. We denote the scheme above by $\Pi^{\mathrm{V}}_{\mathrm{PRF,rnd}}[f, \kappa, \ell]$.

A related but stateful scheme is as follows. Let $\mathrm{ctr} = 0$. Suppose we want to encrypt the string $x = x_1 \cdots x_m$, where $|x_i| = \ell$ and $\mathrm{ctr} + m \leq 2^\ell$. Then we can set the ciphertext to be

$$(\mathrm{ctr},\ f_a(ctr \oplus \langle 1 \rangle) \oplus x_1) \ \|\ \cdots \ \|\ f_a(ctr \oplus \langle m \rangle) \oplus x_m))$$

and then replace ctr by $\mathrm{ctr} + m$. We denote the scheme by $\Pi^{\mathrm{V}}_{\mathrm{PRF,ctr}}[f, \kappa, \ell]$. (Note that we are limited to a total of 2^ℓ messages, since ctr must be an ℓ-bit number.)

Both of these schemes are secure. Intuitively, however, the stateful scheme is better, since the probabilistic scheme might choose the same r twice. We will prove these claims in the following sections. (The bounds we obtain, using the triangle inequality, are not very tight. As an **Exercise**, you might consider ways to improve these schemes and/or the analysis.)

2.3.6 Proof that the stateful Vernam block cipher is secure

Theorem 2.8 *Suppose you can (t, q, μ, ϵ)-break $\Pi^{\mathrm{V}}_{\mathrm{PRF,ctr}}[f, \kappa, \ell]$. Then you can $(t + \lambda_1\mu + \lambda_0,\ q,\ \mu,\ \epsilon/2)$-distinguish f from a random function (for absolute constants λ_1 and λ_0).*

Proof: Let E be an adversary which (t, q, μ, ϵ)-breaks $\Pi^{\mathrm{V}}_{\mathrm{PRF,ctr}}[f, \kappa, \ell]$. For brevity, let us assume that E makes oracle queries of M strings, each consisting of m blocks (so $\mu = Mm\ell$). By definition of E's success we know that

$$\Pr\left[E^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_a(0^{|\cdot|})} = 1\right] \quad \geq \quad \epsilon.$$

Thus

$$\left(\Pr\left[E^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(\cdot)} = 1\right]\right) - \left(\Pr\left[E^{\mathcal{E}_a(0^{|\cdot|})} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1\right]\right) \geq$$
$$\epsilon - \left(\Pr\left[E^{\mathcal{E}_\rho(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1\right]\right)$$

By the triangle inequality either

$$\Pr\left[E^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(\cdot)} = 1\right] \quad \geq \quad \frac{\epsilon}{2} - \frac{\Pr\left[E^{\mathcal{E}_\rho(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1\right]}{2} \tag{6}$$

or

$$\Pr\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1\right] - \Pr\left[E^{\mathcal{E}_a(0^{|\cdot|})} = 1\right] \quad \geq \quad \frac{\epsilon}{2} - \frac{\Pr\left[E^{\mathcal{E}_\rho(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1\right]}{2} \tag{7}$$

(The triangle inequality states that if $X + Y \geq Z$, then either $X \geq Z/2$ or $Y \geq Z/2$.) Without Loss of Generality (WLOG), suppose Equation 6 holds. (This is a non-uniform choice.) We construct a distinguishing test D as follows. D has access to an oracle h computing a function $h : \{0,1\}^\ell \to \{0,1\}^\ell$. D initializes counter ctr to 0^ℓ. Then, D behaves just as E behaves, but when E makes an oracle query of some string $x_1 \cdots x_m$, where $|x_i| = \ell$, algorithm D must answer this query on E's behalf. It answers the query by computing

$$y = h(\mathrm{ctr} \oplus 1) \oplus x_1 \,\|\, \cdots \,\|\, h(\mathrm{ctr} \oplus m) \oplus x_m)$$

and returning (ctr, y); then, D updates ctr, replacing it by $\mathrm{ctr} \oplus m$.

Observe that if $h \leftarrow \mathcal{R}_{\ell \to \ell}$, then D^h provides the (virtual) E with an identical environment to E running in the presence of $\mathcal{E}_\rho(\cdot)$ (that is, E's view is the same in both cases). Similarly, if $h \leftarrow [a \leftarrow \mathcal{K} : f_a]$, then D^h provides the (virtual) E with an identical environment to E running in the presence of $\mathcal{E}_a(\cdot)$. Thus

$$\begin{aligned} \Pr\left[E^{\mathcal{E}_a(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(\cdot)} = 1\right] &= \Pr\left[D^f = 1\right] - \Pr\left[D^\rho = 1\right] \\ &= \mathrm{Adv}_D \end{aligned} \tag{8}$$

and so, from Equations 6 and 8,

$$\mathrm{Adv}_D \quad \geq \quad \frac{\epsilon}{2} - \frac{\Pr\left[E^{\mathcal{E}_\rho(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1\right]}{2} \tag{9}$$

A similar argument constructs a distinguishing algorithm D' for which

$$\mathrm{Adv}_{D'} \quad \geq \quad \frac{\epsilon}{2} - \frac{\Pr\left[E^{\mathcal{E}_\rho(\cdot)} = 1\right] - \Pr\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1\right]}{2}\,. \tag{10}$$

Theorem 2.8 then follows from the following Lemma:

Lemma 2.9 $\Pr\left[E^{\mathcal{E}_\rho(\cdot)}=1\right] = \Pr\left[E^{\mathcal{E}_\rho(0^l|\cdot)}=1\right]$.

Let Experiment 1 be defined by choosing a random function ρ from ℓ bits to ℓ bits and answering each query $x_1\cdots x_m$ by $(\mathrm{ctr}, \rho(\mathrm{ctr}\oplus\langle 1\rangle)\oplus x_1 \parallel \cdots \parallel \rho(\mathrm{ctr}\oplus\langle m\rangle\oplus x_m)$ Let Experiment 0 be defined by choosing a random function ρ from ℓ bits to ℓ bits and answering each query $x_1\cdots x_m$ by $(\mathrm{ctr}, \rho(\mathrm{ctr}\oplus\langle 1\rangle) \parallel \cdots \parallel \rho(\mathrm{ctr}\oplus\langle m\rangle)$ and updating ctr. Let $\Pr_0[\cdot]$ and $\Pr_1[\cdot]$ be the corresponding probability measures. We are trying to show that

$$\Pr_1\left[E^h \text{ outputs } 1\right] = \Pr_0\left[E^h \text{ outputs } 1\right]$$

Let $\mathsf{X}_1,\cdots,\mathsf{X}_{mq}$ be the random variables which comprise all of E's queries, let $\mathsf{R}_1,\cdots\mathsf{R}_{mq}$ be the random variables which comprise all of the output values of ρ (that is, $\mathsf{R}_i=\rho(i)$), and let $\mathsf{Y}_i=\mathsf{R}_i\oplus\mathsf{X}_i$. Now clearly $\mathsf{R}=\mathsf{R}_1\cdots\mathsf{R}_{mq}$ is a Random Variable which is uniformly distributed over $\{0,1\}^{mq}$ (which we shall denote by $\mathsf{R}\sim U_{mq}$), and is independent of $\mathsf{X}=\mathsf{X}_1\cdots\mathsf{X}_{mq}$. But also $\mathsf{Y}=\mathsf{Y}_1\cdots\mathsf{Y}_{mq}$ is a U_{mq}-distributed random variable independent of X since each X_i is independent of R_i. So E the oracle E has access to behaves identically in both Experiments 1 and 2. This concludes the lemma and the theorem both. ■

2.3.7 Proof that the stateless Vernam block cipher is secure

Theorem 2.10 *Suppose you can (t,q,μ,ϵ)-break $\Pi^{\mathrm{V}}_{\mathrm{PRF},\mathrm{rnd}}[f,\kappa,\ell]$. Then you can $(t+\lambda_1\mu+\lambda_0,\ q,\ \mu,\ \epsilon/2-(\mu/\ell)^2 2^{-\ell-2})$-distinguish f from a random function (for absolute constants λ_1 and λ_0).*

Proof: Let E (t,q,μ,ϵ)-break $\Pi^{\mathrm{V}}_{\mathrm{PRF},\mathrm{rnd}}[f,\kappa,\ell]$. For brevity, once again assume that E asks queries of m-blocks. Let Experiment 1 denote running E with an oracle h, where h is determined by choosing $\rho\leftarrow\mathcal{R}_{\ell\to\ell}$ and then answering any query $x_1\ldots x_m$ by choosing a random $r\leftarrow\{0,1\}^l$ and then returning $(r,\ \rho(r\oplus\langle 1\rangle)\oplus x_1 \parallel \cdots \parallel \rho(r\oplus\langle m\rangle)\oplus x_m)$ and then updating ctr. Let Experiment 0 denote running E with an oracle h, where h is determined by choosing $\rho\leftarrow\mathcal{R}_{\ell\to\ell}$ and then answering any query $x_1\ldots x_m$ by choosing a random $r\leftarrow\{0,1\}^l$ and then returning $(r,\ \rho(r\oplus\langle 1\rangle) \parallel \cdots \parallel \rho(r\oplus\langle m\rangle))$. Let $\Pr_0[\cdot]$ and $\Pr_1[\cdot]$ denote the corresponding probability measures. In either experiment 1 or 0 we define random variables $\mathsf{R}_1,\cdots,\mathsf{R}_{mq}$ where R_i is the i-th query to ρ. Let B_i be the event that some pair of $\{\mathsf{R}_1,\ldots,\mathsf{R}_{i-1}\}$ are equal. Let $\mathrm{B}=\mathrm{B}_{mq}$. We will need use the following two lemmas:

Lemma 2.11 $\Pr_1\left[E^{\mathcal{E}_\rho(\cdot)}|\overline{\mathrm{B}}=1\right] = \Pr_0\left[E^{\mathcal{E}_\rho(0^l|\cdot)}|\overline{B}=1\right]$.

Proof: We leave this as an **Exercise**. ■

We will denote the common value, above, by β.

Lemma 2.12 $\Pr_0[\mathrm{B}],\ \Pr_1[\mathrm{B}] \le q^2m^2 2^{-\ell-1}$.

Proof: We leave this as an **Exercise**, too. ■

Now from the Proof of Theorem 2.8 it suffices to show that

$$\mathrm{Adv}_E = \Pr_1\left[E^{\mathcal{E}_\rho(\cdot)}=1\right] - \Pr_0\left[E^{\mathcal{E}_\rho(0^l|\cdot)}=1\right]$$

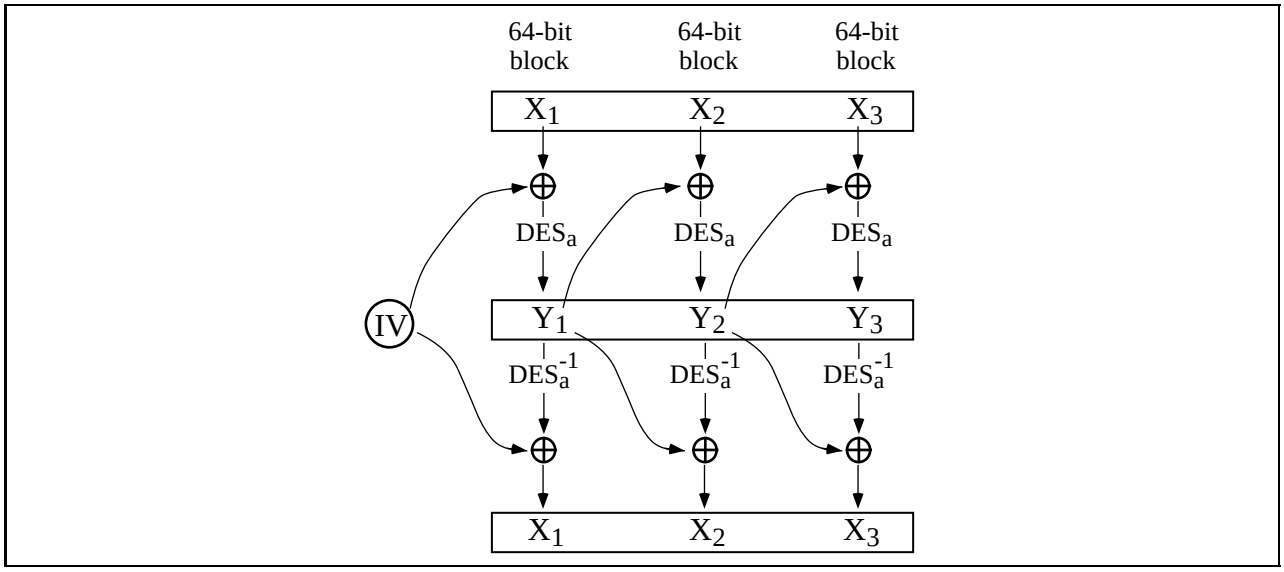

Figure 3: Cipher Block Chaining with DES.

is at most $q^2 m^2 2^{\ell-1}$. To show this we bound the above quantity by

$$\begin{aligned}
\mathrm{Adv}_E^{\rho} &= \Pr_1\left[E^{\mathcal{E}_\rho(\cdot)} = 1 \mid \overline{B}\right]\Pr_1\left[\overline{B}\right] + \Pr_1\left[E^{\mathcal{E}_\rho(\cdot)} = 1 \mid \mathrm{B}\right]\Pr_1[\mathrm{B}] - \\
&\qquad \Pr_0\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1 \mid \overline{B}\right]\Pr_0\left[\overline{\mathrm{B}}\right] - \Pr_0\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1 \mid \mathrm{B}\right]\Pr_0[\mathrm{B}] \\
&\leq \Pr_1\left[E^{\mathcal{E}_\rho(\cdot)} = 1 \mid \overline{B}\right]\Pr_1\left[\overline{B}\right] + \Pr_1[\mathrm{B}] - \Pr_0\left[E^{\mathcal{E}_\rho(0^{|\cdot|})} = 1 \mid \overline{B}\right]\Pr_0\left[\overline{\mathrm{B}}\right] \\
&\leq \beta\left(\Pr_1\left[\overline{\mathrm{B}}\right] - \Pr_0\left[\overline{\mathrm{B}}\right]\right) + \Pr_1[\mathrm{B}] \qquad \text{(By lemma 2.11)} \\
&= \beta(1 - \Pr_1[\mathrm{B}] - 1 + \Pr_0[\mathrm{B}]) + \Pr_1[\mathrm{B}] \\
&= \beta(\Pr_0[\mathrm{B}]) + (1-\beta)\Pr_1[\mathrm{B}] \qquad \text{(By lemma 2.12)} \\
&\leq q^2 m^2 2^{\ell-1}\ .
\end{aligned}$$

This completes the proof. ■

2.3.8 Cipher block chaining

A widely used encryption scheme is Cipher Block Chaining, defined as follows. The CBC encryption of $x = x_1 \cdots x_m$, where $|x_i| = \ell$, using key $a \in \{0,1\}^\kappa$, initialization vector $\mathrm{IV} \in \{0,1\}^\ell$, and PRF f, is

$$f^m_{\mathrm{IV},a}(x_1 \ldots x_m) \quad = \quad (\mathrm{IV}, y_1 \ldots y_m)$$

where $y_i = f(y_{i-1} \oplus x_i)$ for $1 \leq i \leq m$, and $y_0 = \mathrm{IV}$. (See Figure 3.) This is a secure encryption scheme. (**Exercise**: prove this claim.)

3 Message authentication

Suppose A sends a message x to B, along with a "tag" called a Message Authentication Code MAC(x). The purpose of the MAC is to prove that x is an authentic message from A, so that B will believe that A (and

no one else) sent x, and also so that A cannot later deny having sent x. Thus MACs act just like signatures in paper-based communication. In this section, we will assume the "trust architecture" is such that A and B share a private key a.

Now suppose there is an adversary F (for Forger or Felicia[2]) capable of performing an Adaptive Chosen Message Attack; that is, she can choose a plaintext x, ask her oracle for its MAC, think about it for a while, choose another plaintext, and so on. We say that F "wins" in such an experiment if she can produce the MAC for a message x which she has not previously seen. A secure MAC is one which cannot be defeated by an adversary with high probability.

The following sections formalise these ideas. We then study how to construct secure MACs, and show why some naive schemes are inadequate.

3.1 Formal definitions of a MAC

A message authentication code (MAC) is a 3-tuple

$$\mathcal{M} = (\mathcal{G}, \mathcal{V}, \mathcal{K})$$

where

$$\mathcal{G} : \underbrace{\Sigma^*}_{\text{key}} \times \underbrace{\Sigma^*}_{\text{message}} \times \underbrace{\Sigma^\infty}_{\text{coins}} \to \underbrace{\Sigma^*}_{\text{tag}}$$

is a MAC generation (or signing) algorithm such that $t = \mathcal{G}(a, x, r)$ is the tag for message x using key a and random coins r (we can define a stateful MAC generator in a similar way); we will often write $t \leftarrow \mathrm{MAC}_a(x)$ for brevity.

$$\mathcal{V} : \underbrace{\Sigma^*}_{\text{key}} \times \underbrace{\Sigma^*}_{\text{message}} \times \underbrace{\Sigma^*}_{\text{purported tag}} \to \underbrace{\Sigma^*}_{\text{YES/NO}}$$

is the MAC verifier, defined such that $\mathcal{V}(x, t)$ returns "YES" if $t \in \mathrm{MAC}_a(x)$, and "NO" otherwise, and

$$\mathcal{K} : \underbrace{1^*}_{\text{security parameter}} \times \underbrace{\Sigma^*}_{\text{coins}} \to \underbrace{\Sigma^*}_{\text{key}}$$

is the key generator.

Definition 3.1 *Let $\mathcal{M} = (\mathcal{G}, \mathcal{V}, \mathcal{K})$ be a MAC. An adversary* F *is said to (t, q, μ, ϵ)-break $\mathcal{M}$ if*

$$\begin{aligned} \mathrm{Adv}_F &\stackrel{\text{def}}{=} \Pr[a \leftarrow \mathcal{K}; ((x_1, \ldots, x_q), (x^*, t^*)) \leftarrow F(\mathcal{G}_a) : \\ &\qquad \mathcal{V}(x^*, t^*) = 1 \text{ and } x^* \notin \{x_1, \ldots, x_q\}] \\ &\geq \epsilon \end{aligned}$$

and F asks at most q queries of total length μ bits and runs in at most t steps.

As usual, we can make all the parameters be functions of the security parameter k; this allows us to make the following complexity-theoretic definition.

Definition 3.2 *A MAC $\mathcal{M} = (\mathcal{G}, \mathcal{V}, \mathcal{K})$ is secure if $\mathcal{G}$, $\mathcal{V}$ and $\mathcal{V}$ are PPT, and if for all F, if F $(t(k), q(k), \mu(k), \epsilon(k))$-breaks $\mathcal{M}$ then if $t(k)$ is polynomial then $\epsilon(k)$ is negligible.*

[2] Ever since the time someone called the Eavesdropper Eve for short, it has been traditional to assume the enemy is female.

3.2 The interaction between message authentication and encryption

It is important that the message be encrypted before it is tagged. For example, suppose A sends $(x, \mathrm{MAC}(x))$. If the $\mathrm{MAC}(x) = x \oplus a$, where a is a one-time pad, then F can simply compute $x \oplus \mathrm{MAC}(x)$ to recover $|x|$ bits of the pad; then she can authenticate any message she pleases (of the same length).

Now suppose A sends $(\mathcal{E}_a(x), \mathrm{MAC}_a(x))$. There is no guarantee that $\mathrm{MAC}(x)$ doesn't leak useful information about x, thus undoing the benefits of encrypting it.

So what A should send is $(y, \mathrm{MAC}_{a2}(y))$, where $y = \mathcal{E}_{a1}(x)$. Note that we need to use different keys for the encryption and the MAC, since it is possible, for example, that $\mathcal{E}(x)$ leaks the first few bits of x, but that $\mathrm{MAC}(x)$ "ignores" these very same bits, which can therefore be changed at will. (We can use the following *key separation* method to generate two keys from one: let f_a be a PRF; then define $a1 = f_a(0)$ and $a2 = f_a(1)$.)

In the following sections, we shall assume the appropriate encryption has taken place, and concentrate on authentication.

3.3 How *not* to achieve secure message authentication

3.3.1 Using a one-time pad

We have seen that a xor'ing with a one-time pad will achieve encryption which is secure in the strongest possible sense (namely information-theoretically). Can we use a one-time pad to implement message authentication? Suppose parties A, B share a long key a, which we can divide up into consecutive blocks of k bits. Then the MAC for message $x = x_1 x_2 \ldots x_n$ is $m = m_1 m_2 \ldots m_n$ where m_i is block number $2i - 1$ of a if $x_i = 0$ and block $2i$ otherwise; that is, we take an odd or even numbered block depending on the parity of the corresponding bit of x. (Of course, we also need to send an index specifying where A is in its pad, since some messages might not get through, and hence B might not have "kept up with" A.)

One drawback of this scheme is its inefficiency: if $|x| = n$, then $|m| = nk$, which is unacceptably long. More importantly, it is not secure: although the chance of correctly guessing the MAC for any particular bit is only 2^{-k}, this scheme is very *malleable*, for we can easily omit trailing blocks of m, to generate the MAC for any prefix of x. We can even combine MACs from separate messages to authenticate composite messages which were never actually sent; for example, suppose A sends to B $(0, m(0))$ followed by $(1, m(1))$; then an adversary F can intercept these messages, and send the message $(01, m(0) \cdot m(1))$.

3.3.2 Using CBC-IV

This scheme is also malleable. Suppose A sends (x, y) and (x', y'), where $x = x_1 x_2 x_3$, $y = \mathrm{MAC}(x) = y_1 y_2 y_3$, and similarly for x', y'. Then F can intercept these messages and send the following forgery: $(x_1 x_2 z, y_1 y_2 y'_3)$, where $z = x'_3 \oplus y'_2 \oplus y_2$. This works since the tag for z is $z \oplus y_2 = x'_3 \oplus y'_2 = y'_3$. By intercepting enough messages, she may be able to forge any message she chooses. (Note that Todd incorrectly assumes in his notes that F has access to the private key a, which would of course allow her to forge anything.)

3.4 How to achieve secure message authentication

3.4.1 Using CBC MAC

The problem with the previous CBC method was that each bit of the tag was independent. What we need to do is to make the tag depend on the whole message. One way to do this is to define MAC(x) as a prefix of

$$f^{(n)}(x_1 \cdots x_n) \stackrel{\text{def}}{=} f(f(\cdots f(f(x_1 \oplus x_2) \oplus \cdots \oplus x_{n-1}) \oplus x_n)$$

where f is some underlying block cipher. This method is a pervasively used international and U.S. standard. In [BKR], it is proved secure.

3.4.2 The GGM Scheme

Here is another way to make the MAC be a function of all the bits of x, due to Goldreich, Goldwasser, and Micali (ref?). Suppose A, B share a key a, which can be used to specify a member of a set of PRFs $\{f_a : \Sigma^k \to \Sigma^k\}$. Then we can use MAC($x$) $= f_a(x_1 \ldots x_k)$. Of course, this allows us to tamper with the remaining $n - k$ bits. What we really need is a function $\Sigma^* \to \Sigma^k$. So suppose we have a generator $g : \Sigma^k \to \Sigma^{2k}$. (We can make one from a PRF by setting $g(a) = f_a(0) \cdot f_a(1)$.) Then construct a complete binary tree as follows: at each node, with label α of length k, form $g(\alpha)$, and label the left child with the first k bits of $g(\alpha)$ and the right child with the second k bits. The root of the tree is 0^k. To form MAC(x), walk down the tree taking a left or right branch according as $x_i = 0$ or 1 respectively, until you reach a leaf; its label will be the desired MAC. Unfortunately, this scheme takes $O(n)$ time (where each application of $g(\cdot)$ takes $O(1)$ time), which is unacceptable.

3.4.3 Steven's method

Suppose we enclose the message x inside unique delimiters, say \$ and #, and then encode it into a binary alphabet. Call the resulting string x'. For example,

```
011 -> $011# -> 01 00 11 11 10
```

(Actually, there is no need to have distinct starting and finishing delimters, since they must always be balanced in any well-formed message.) If we encrypt the result with a one-time pad (i.e., MAC(x) $= \mathcal{E}_a(x')$), we have a secure MAC. (**Exercise**: prove this claim.) However, the resulting MAC has length proportional to n, which is unacceptable.

3.4.4 The WC scheme

The general idea, due to Wegman and Carter [WC80], is that we use MAC(x) $= \mathcal{E}_a(h(x))$, where $h(x)$ is the hash value of x, which has fixed length. Clearly if we use any particular hash function h, an adversary could construct the MACs for all x' such that $h(x) = h(x')$. However, if we choose h at random from a family of *universal* hash functions, the chance of an adversary finding those x' which collide with x under h can be made suitably small, and hence the MAC will be secure. (It will also be efficient: we can use the first few bits of the shared key a to specify h, and then we exhaust a at a rate proportional to the *number* of messages we send, irrespective of their lengths.) The following sections formalise these ideas.

3.5 Secure message authentication using hash functions

3.5.1 Definitions of universal hash functions

Definition 3.3 *Let* $\mathcal{H} = \{h : A \rightarrow B\}$. $\mathcal{H}$ *is* universal-2 *if*

$$\forall x \neq y,\ x, y \in A.\ \Pr[h(x) = h(y)] \leq 1/|B|$$

$\mathcal{H}$ *is* ϵ-almost universal-2 *($\epsilon - \mathrm{AU}_2$) if the bound is ϵ instead of $1/|B|$.*

(The 2 refers to the fact that we are only considering pairs of points.)

Note that the bound $1/|B|$ is the best we can do if every function h uniformly distributes each member of A over the range B.

It is possible for h to be universal-2 and yet $h(x+1) = y+1$ where $h(x) = y$. Functions which avoid this kind of correlation earn the honor of being called *strongly universal*:

Definition 3.4 $\mathcal{H}$ *is* strongly universal-2 *if*

$$\forall x, y.\ \Pr[h(x) = y] \leq 1/|B|$$

and

$$\forall x_0 \neq x_1,\ y_0, y_1.\ \Pr[h(x_1) = y_1 | h(x_0) = y_0] \leq 1/|B|$$

$\mathcal{H}$ *is* ϵ-almost strongly universal-2 *($\epsilon - \mathrm{ASU}_2$) if the second bound is replaced by ϵ.*

The first part of this definition says that h is equally likely to map x to every element in the range. The second part says that, knowing where one element got mapped to won't help you predict where another, distinct, element will be mapped to.

To prove the WC scheme secure, we actually need a somewhat weaker notion, as follows.

Definition 3.5 *Let* $\mathcal{H} = \{h : A \rightarrow B\}$. $\mathcal{H}$ *is* XOR universal-2 *if*

$$\forall x \neq y, z.\ \Pr[h(x) \oplus h(y) = z] \leq 1/|B|$$

$\mathcal{H}$ *is* ϵ-almost XOR universal-2 *($\epsilon - \mathrm{AXU}_2$) if the bound is ϵ instead of $1/|B|$.*

The following lemmas show how these notions relate to one another.

Lemma 3.6 *If* $\mathcal{H}$ *is* $\epsilon - \mathrm{ASU}_2$, *then* $\mathcal{H}$ *is* $\epsilon - \mathrm{AXU}_2$.

Proof:

$$\begin{aligned}
\forall x_0 \neq x_1,\ y_0, y_1.\ \Pr[h(x_1) = y_1 | h(x_0) = y_0] &\leq \epsilon \quad \text{since } h \text{ is } \epsilon - \mathrm{ASU}_2 \\
\Pr[h(x_1) = b \oplus h(x_0) | h(x_0) = y_0] &\leq \epsilon \quad \text{setting } y_1 = b \oplus h(x_0) \\
\Pr[h(x_1) = b \oplus h(x_0)] &\leq \epsilon \quad * \\
\Pr[h(x_1) \oplus h(x_0) = b] &\leq \epsilon
\end{aligned}$$

The justification for step * is that the conditioning event is irrelevant. ■

Lemma 3.7 *If* $\mathcal{H}$ *is* $\epsilon - \mathrm{AXU}_2$, *then* $\mathcal{H}$ *is* $\epsilon - \mathrm{AU}_2$.

Proof: The proof is similar to the above, and omitted. ■

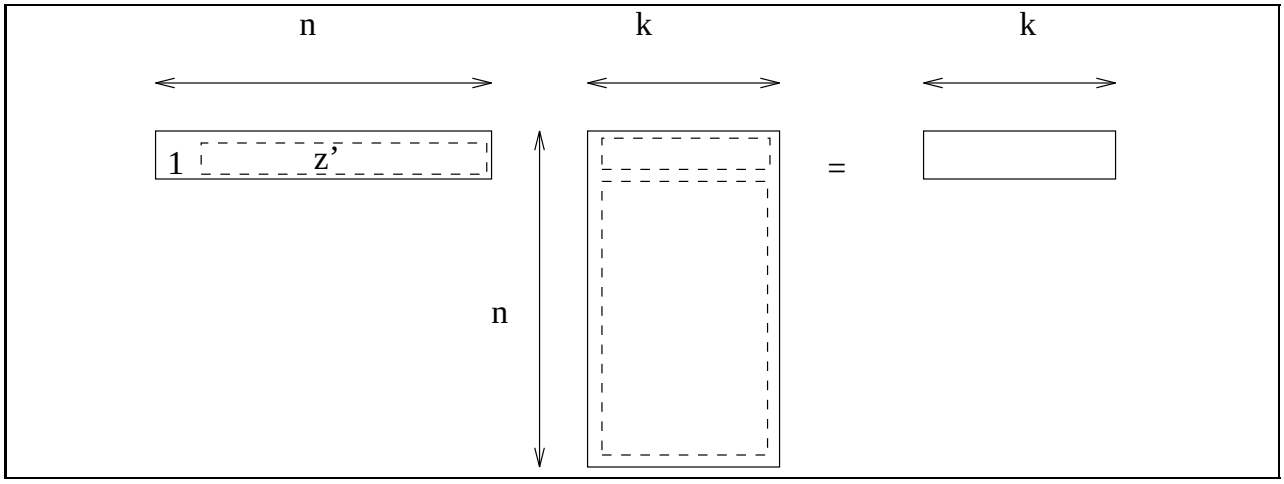

Figure 4: A hash function implemented as matrix multiplication.

3.5.2 Proving the WC MAC scheme secure

Theorem 3.8 *If $\mathcal{H}$ is $\epsilon - \mathrm{AXU}_2$, then $\mathrm{MAC}(x) = (i, h(x) \oplus a_{i \ldots i+n-1})$ is a secure MAC, where $h \in \mathcal{H}$, $|x| = n$ and i is the index into the shared private key a.*

Proof: Unfortunately, I missed this lecture, and Michael Dilger's proof is not convincing. ■

3.5.3 Constructing universal hash functions

We can construct a $\epsilon - \mathrm{AXU}_2$ hash function using matrix multiplication as follows. Let x be an n-element row vector and H be an $n \times k$ matrix whose elements are U_1-distributed. Then $t = h(x) \stackrel{\mathrm{def}}{=} xH \pmod 2$ is a k-element row vector, whose ith element is given by $t_i = x \odot H^{(i)}$, where $H^{(i)} = (H_{1i}, H_{2i}, \ldots, H_{ni})$ is the ith column of H and $\odot$ denotes inner product.

Theorem 3.9 *The above scheme is $\epsilon - \mathrm{AXU}_2$, with $\epsilon = 2^{-k}$.*

Proof: Required To Prove:

$$\forall x \neq y, z.\ \Pr[h(x) \oplus h(y) = z] \leq \epsilon$$

Now,

$$\begin{aligned} h(x) \oplus h(y) &= (x \oplus y)\ H \quad \text{by linearity} \\ &\stackrel{\mathrm{def}}{=} zH \\ &= (z \odot H^{(1)}, z \odot H^{(2)}, \ldots, z \odot H^{(k)}). \end{aligned}$$

Since $x \neq y$, let us assume Without Loss Of Generality that x, y differ in their first bit, so $z = (1, z')$ with $|z'| = n - 1$. We multiply z' by all the rows of H except the first, and then xor in the first row of H (see Figure 4.) Now the first row of H is U_k-distributed, so by the following lemma, so also is $h(x) \oplus h(y)$.

Lemma 3.10 *If X and Y are two random binary variables, X is uniformly distributed over $\{0,1\}^{|X|}$, and Y has an arbitary distribution, then $X \oplus Y$ is uniformly distributed over $\{0,1\}^{|X|}$.*

Proof: Left as an **Exercise**. ■

This completes the proof of the theorem. ■

Corollary 3.11 *$h(x) = xH \oplus \rho$, where $\rho \sim U_n$, is $\epsilon - \mathrm{ASU}_2$.*

Proof: (Sketch.) The previous scheme had the property that $h(0) = 0$, but by adding (mod 2) a random vector to the result, we get a stronger scheme. ■

3.5.4 Efficiency of the matrix multiplication hash function

How efficient is the matrix multiplication scheme? Typically we wish to authenticate a whole NFS block at a time, implying that $|x| = 2^{12}$. An acceptable degree of security might be achieved with $k = 32$ (so $\epsilon(k) \sim 2 \times 10^{-10}$). Hence the matrix is very large, and the computation very slow. There are three ways to speed things up. (1) Perform operations one word at a time (thus utilising the intrinsic parallelism of the hardware). (2) Use x as an index into a lookup-table which store the results of all possible inner product calculations for each column of the matrix. (3) Use a different hashing scheme. We will now briefly study this last option.

3.5.5 Bucket hashing

The following scheme, due to Rogaway (ref?), is the fastest known MAC scheme, and has been patented by IBM. Suppose we have N "buckets" $B_1, \ldots, B_N$, and we wish to hash n messages, $x_1, \ldots, x_n$. For each message x_i, choose three distinct buckets at random, with indices i_1, i_2 and i_3, and set $h(x_i) = (i_1, i_2, i_3)$. The hash of the whole set of messages is then $(h(x_1), \ldots, h(x_n), z)$, where $z = (\oplus B_1, \ldots, \oplus B_N)$ and $\oplus B_i$ denotes the result of xor'ing every element in B_i.

Claim 3.12 *In the above bucket hashing scheme,*

$$\Pr_{x \neq x'}[h(x) = h(x')] \leq \left(\frac{1}{1 - n/\binom{N}{3}} \right) \left(\frac{1}{3348 N^6} + O(N^{-7}) \right)$$

Corollary 3.13 *Bucket hashing is $\epsilon - \mathrm{AU}_2$.*

This is an interesting result, but it seems to be too weak: the hash function is universal, but the WC scheme requires an XOR-universal hash function. However, it turns out that we can "glue together" different kinds of hash functions to achieve the desired results. This will be proved in the next section.

3.5.6 Combining different hash functions

We can use the following result to extend the domain of any hash function.

Theorem 3.14 *Let $\mathcal{H} = \{h : \{0,1\}^a \rightarrow \{0,1\}^b\}$ be $\epsilon - \mathrm{AU}_2$. Then $\mathcal{H}' = \{h' : \{0,1\}^{2a} \rightarrow \{0,1\}^{2b}\}$ is $\epsilon - \mathrm{AU}_2$, where $h'(xy) = h(x) \cdot h(y)$ and $|x| = |y| = a$.*

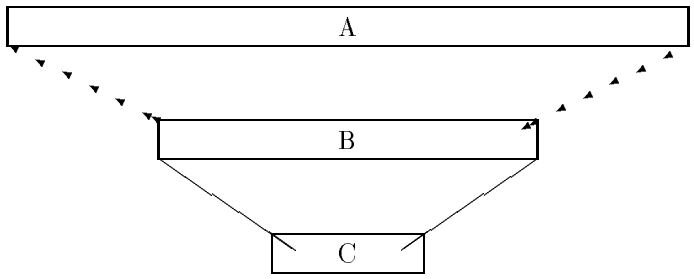

Figure 5: Composition of hash functions mapping A to B and B to C

Proof: RTP $\Pr[h'(xy) = h'(x'y')] \leq \epsilon$. In the worst case, we may assume WLOG $y = y'$; then $h'(xy) = h'(x'y')$ iff $h(x) = h(x')$, which occurs with probability at most ϵ, since h is $\epsilon - \mathrm{AU}_2$. ■

For efficiency reasons, we might want to reduce the size of the resulting hash value. We can do this without too much loss of accuracy, as the following theorem shows.

Theorem 3.15 *If $\mathcal{H}_1 : A \to B$ is $\epsilon_1 - \mathrm{AU}_2$ and $\mathcal{H}_2 : B \to C$ is $\epsilon_2 - \mathrm{AU}_2$, then $\mathcal{H}_2 \circ \mathcal{H}_1$ is $\epsilon_1 + \epsilon_2 - \mathrm{AU}_2$. (See Figure 5.)*

Proof: Two points collide under $h' \in \mathcal{H}_2 \circ \mathcal{H}_1$ if they either collide under $h_1 \in \mathcal{H}_1$ or they collide under $h_2 \in \mathcal{H}_2$. This occurs with probability $\epsilon_1 + \epsilon_2 - \epsilon_1\epsilon_2$, since the events are independent but not mutually exclusive. We usually ignore the $\epsilon_1\epsilon_2$ term since it is negligible. ■

Finally, we show how to achieve a strong hash function from weaker and/or smaller parts.

Theorem 3.16 *If $\mathcal{H}_1 : A \to B$ is $\epsilon_1 - \mathrm{AU}_2$ and $\mathcal{H}_2 : B \to C$ is $\epsilon_2 - \mathrm{ASU}_2$, then $\mathcal{H}_2 \circ \mathcal{H}_1$ is $\epsilon_1 + \epsilon_2 - \mathrm{ASU}_2$.*

Proof: Intuitively, if $x \neq y$ collide under h_1 (with probability ϵ_1), then we know $h'(x)$ and $h'(y)$ map to the same point. If x, y don't collide under h_1, we still have an ϵ_2 chance of predicting $h'(y)$ from $h'(x)$, since h_2 is still a "weak link". Formally, we have to show that h' satisfies both parts of the definition of a $\epsilon - \mathrm{ASU}_2$ function.

1. Since h_2 is $\epsilon - \mathrm{ASU}_2$ (using part 1 of the definition), we have the following:

$$\forall x \in A, y \in C.\ \Pr[h'(x) = y] \leq 1/|C|$$

2. Let P_1 be the event $h_2(h_1(x_1)) = y_1$ and P_0 the event $h_2(h_1((x_0)) = y_0$, where $x_0 \neq x_1$. Let C be the event that two distinct points collide under h_1, i.e., $h_1(x_1) = h_1(x_0)$.

$$\begin{aligned} \Pr[P_1|P_0] &= \Pr[P_1|P_0 \wedge \neg C] \Pr[P_0 \wedge \neg C] + \Pr[P_1|P_0 \wedge C] \Pr[P_0 \wedge C] \\ &\leq \epsilon_2 \cdot 1 + 1 \cdot \epsilon_1 \end{aligned}$$ ■

4 Entity authentication

Entity authentication is the process by which an agent in a distributed system gains confidence in the identity of a communication partner. (This should not be confused with message authentication, which can,

however, be used as a tool for achieving entity authentication.) There may be two parties involved or more; the authentication may be unilateral or mutual; and parties might (the symmetric case) or might not (the asymmetric case) share a secret key. We shall study symmetric mutual authentication (MA) between two parties, A and B.

Merely knowing that it really was party B that you just spoke to is not often very useful (possible applications include network monitoring). In the secret key exchange problem, the parties also want to distribute a "fresh" and "secret" *session key*, α. This might seem redundant, since A and B already share "long lived" secret key a. However, we wish to prevent messages which are sent during one session and then replayed in another session from being considered authentic. One way would be to have both parties share state (e.g., a message counter) across sessions. However, we prefer not to have to make this assumption.

In the following sections, we shall define what it means for a MA protocol to be secure, and then study ways to achieve it, and ways *not* to achieve it. We will be somewhat less formal than in other sections (no proofs!), since [BR93] provides a very lucid exposition of the details.

4.1 How *not* to achieve secure mutual authentication

The simplest MA protocol is for A to send B a challenge, such as an encrypted random number $\mathcal{E}_a(R_A)$, and then ask B to send back its decryption R_A (or vice versa). (A number such as R_A which is only used once is called a *nonce*, and is used to guarantee freshness. It must be non-predictable.) It is clear that if F is just a passive eavesdropper, she can break this scheme iff she can break the underlying encryption scheme.

But in reality, adversaries are more powerful. In particular, they may be able to invoke different sessions or versions of a party, and interact with them (think of a daemon starting multiple threads on demand), before attempting a forgery. Let us denote the session number of a party by a superscript. Then the following *interleaved attack*, in which F "talks to" A^t, causes A^s, $s \neq t$, to falsely believe it has communicated with B (since we assume that B is the only party besides A which knows a).

$$\begin{array}{ccccc} A^s & \xrightarrow{\mathcal{E}_a(R_A)} & E & \xrightarrow{\mathcal{E}_a(R_A)} & A^t \\ A^s & \xleftarrow{R_A} & E & \xleftarrow{R_A} & A^t \end{array}$$

(In the above diagram, the order of messages is obtained by reading left to right and top to bottom. Hence first A^s sends $\mathcal{E}_a(R_A)$, then E sends $\mathcal{E}_a(R_A)$, and so on.)

Not all attacks are of this form, however. For example, suppose A sends $f(R_A)$ for some polynomial f of degree k. Then after witnessing k $(f(R_A), R_A)$ pairs, F can interpolate this polynomial and then forge any identity she pleases.

In the following section, we shall give a definition which will prevent schemes such as the above from being considered secure.

4.2 Definition of secure mutual authentication

We give to F a series of (probabilistic) oracles Π_{ij}^s, where s denotes the session number, i denotes the sender and j the receiver. F will have a conversation with each oracle, which may then go into an accepting state (and notify F of this fact). For example, consider the following exchange of messages between parties A and B.

$$A \xrightarrow{\alpha} B$$

$$A \xleftarrow{\beta} B$$
$$A \xrightarrow{\gamma} B$$

A "faithful" or *AB-benign* adversary, which acts just like a wire, would simulate this exchange by the following conversation with its oracles:

$$\begin{array}{lcccl}
\Pi^s_{AB} & \overset{\lambda}{\underset{\alpha}{\leftrightarrows}} & E & & \Pi^t_{BA} \\
\Pi^s_{AB} & & E & \overset{\alpha}{\underset{\beta}{\leftrightarrows}} & \Pi^t_{BA} \\
\Pi^s_{AB} & \overset{\beta}{\underset{\gamma}{\leftrightarrows}} & E & & \Pi^t_{BA} \\
\Pi^s_{AB} & & E & \overset{\gamma}{\underset{\lambda}{\leftrightarrows}} & \Pi^t_{BA}
\end{array}$$

In the above, λ denotes the empty string.

Each question-response pair occurs during successive time intervals, call them $\tau_0, \tau_1, \ldots$. Thus the conversation between E and Π^s_{AB} can be denoted by $((\tau_0, \lambda, \alpha), (\tau_2, \beta, \gamma))$, and the conversation between E and Π^t_{BA} is $((\tau_1, \alpha, \beta), (\tau_3, \gamma, \lambda))$.

If every message that Π^s_{AB} sent out (except the last) is delivered to Π^t_{BA}, with the response to this message being returned to Π^s_{AB} as its own next message, then we say that the conversation of Π^t_{BA} matches that of Π^s_{AB}. Similarly, if every message that Π^t_{BA} receives was previously generated by Π^s_{AB}, and each message that Π^t_{BA} sends out is subsequently delivered to Π^s_{AB}, with the response that this message generates being returned to Π^t_{BA} as its own next message, then we say that the conversation of Π^s_{AB} matches the one of Π^t_{BA}. This second condition is easily seen to imply the first. We will say that oracle Π^s_{AB} has a matching conversation with Π^t_{BA} if the first has conversation κ', the second has conversation κ, and κ' matches κ. (Hence either party can be the initiator.)

Let $\mathsf{No-Matching}^E(k)$ be the event that there exist i, j, s such that $\Pi^s_{i,j}$ accepted and yet there is no oracle $\Pi^t_{j,i}$ which engaged in a matching conversation. Now we can finally give our definition of secure mutual authentication.

Definition 4.1 *We say that* Π *is a secure mutual authentication protocol is for any polynomial time adversary E,*

1. *(Matching conversation* $\Rightarrow$*acceptance.) If oracles* Π^s_{AB} *and* Π^t_{BA} *have matching conversations, then both oracles accept.*
2. *(Acceptance* $\Rightarrow$*matching conversations.) The probability of* $\mathsf{No-Matching}^E(k)$ *is negligible.*

Note that this is a very strong definition. Suppose, for example, that A always ignores the last bit of the authentication. Even if E only changes that bit, she has won, according to the above definition. This means, for example, that it would not suffice to send $[x]_a \stackrel{\text{def}}{=} (x, \mathrm{MAC}_a(x))$ as part of a protocol, for what if $\mathrm{MAC}_a(x) = \mathrm{MAC}'_a(x) \cdot b$ for some arbitrary bit b? Clearly F could change b at will, and hence "win", even without affecting A's behavior. Nevertheless, in the next section, we study a simple protocol that satisfies this definition.

4.3 Mutual Authentication Protocol 1

Let us define $[x]_a = (x, f_a(x))$ as the authentication of message x, where f is a PRF. We define the MAP1 protocol as follows.

$$
\begin{array}{lcl}
A^a & \xrightarrow{R_a} & B^a \\
A^a & \xleftarrow{[B\cdot A\cdot R_a\cdot R_B]_a} & B^a \\
A^a & \xrightarrow{[A\cdot R_B]_a} & B^a
\end{array}
$$

In words, this says: A sends B a random challenge R_A of length k. B responds by making up a random challenge R_B of length k and returning $[B\cdot A\cdot R_A\cdot R_B]_a$. A checks that B's message is of the right form and is correctly tagged as coming from B. If it is, A sends B the message $[A\cdot R_B]_a$ and accepts. B checks that this message is of the right form and is correctly tagged as coming from A, and if it is, accepts. (Checking that the message is of the right form includes checking that the nonce you get back is the one you sent out.)

Theorem 4.2 *If f is a PRF, then MAP1 is secure.*

Proof: See [BR93]. ■

5 Asymmetric encryption

Asymmetric encryption is also known as public key cryptography. It assumes that each party has a public key, which is known to all, and a private key, which is kept secret (thus introducing asymmetry). We will denote A's public and secret keys by P_A and S_A respectively. Associated with each key is a function of the same name, which is a permutation of the message space. We require that $P_A(S_A(M)) = M$ and $S_A(P_A(M)) = M$. We also require that it be hard to compute S_A from P_A.

Diffie and Hellman first introduced this idea in 1976 [DH76], and showed how to use it to achieve asymmetric encryption and asymmetric message authentication ("digital signatures"). Rivest, Shamir and Adleman, in 1978, were the first to demonstrate the construction of functions with the desired properties. We shall not go into the details here, since they have been well described elsewhere (see e.g., [CLR90][Chapter 33]), and since they require more background in number theory than we are assuming. Instead, we shall consider RSA as a "black box".

The RSA scheme is normally used in the following way. Choose a random bits string r, and define the *embedding* of message x by $r_x = r\cdot x$. (Typically $|x| = 64$ bits and $|r_x| = 512$ bits.) Then $\mathcal{E}(x) = f(r_x)$, where f is the RSA function. (We need r to ensure the scheme is non-deterministic. See Section 2.3.4.) Unfortunately, this does not guarantee semantic security: we have no guarantee that, just because it is hard to compute r_x from $f(r_x)$, it must also be hard to compute x from $f(r_x)$; also, it is possible that, even if we can't compute x itself from $f(r_x)$, we can compute some properties of x, such as its Least Significant Bit. Intuitively, if every bit of x is hard to compute from $\mathcal{E}(x)$, then knowing $\mathcal{E}(x)$ would give us no advantage in computing some function of x. In other words, if every bit is "hard core", then $\mathcal{E}$ is "secure". In the following sections we will give formal definitions for these terms, and then study some protocols.

5.1 Definition of secure asymmetric encryption

Definition 5.1 *An* encryption scheme (generator) *is a probabilistic algorithm $\mathcal{G}$ such that $\mathcal{G}(1^k) = (\mathcal{E}, \mathcal{D})$, where $\mathcal{E}$ is a probabilistic encryption algorithm and $\mathcal{D}$ is its associated (deterministic) decryption algorithm, i.e., $\mathcal{D}(\mathcal{E}(x, r)) = x$, where $x \in \{0,1\}^{n(k)}$, $r \in \{0,1\}^\infty$, and $n(k)$ is some polynomial. We will usually assume $n(k) = k$.*

We shall now consider the weakest interesting definition of secure encryption. An adversary E can choose any two plaintexts she chooses; when given the encryption of one of them, she wins if she can decide which one it was. A secure encryption scheme is one in which she cannot win with high probability. (This formulation is provably equivalent to semantic security.)

Definition 5.2 *An adversary* (t, ϵ)-breaks $\mathcal{G}(1^k)$ *if*

$$\mathrm{Adv}_E(k) \stackrel{\mathrm{def}}{=}$$
$$2\left(\Pr\left[(\mathcal{E},\mathcal{D}) \leftarrow \mathcal{G}(1^k);\ (x_0,x_1,c) \leftarrow E(\mathcal{E},\mathsf{find});\ y_0 \leftarrow \mathcal{E}(x_0);\ y_1 \leftarrow \mathcal{E}(x_1);\ b \leftarrow \{0,1\} :\ E(c,y_b) = b\right] - \frac{1}{2}\right)$$
$$\geq \epsilon$$

and E *runs in at most* t *steps.*

In the above definition, we subtract $\frac{1}{2}$ since we can always do that well by guessing b, and we multiply by 2 to ensure the advantage lies in $[0,1]$. The variable c is used to store any state which the adversary might find helpful in her later attempt to guess the answer, and can include such things as x_0, x_1, and $\mathcal{E}$.

Definition 5.3 $\mathcal{G}$ *is a* secure encryption scheme *if* $\mathcal{G}(1^k)$, $\mathcal{E}$ *and* $\mathcal{D}$ *run in poly(k) time, and if for all non-uniform PPT adversaries* E, *if* E $(t(k),\epsilon(k))$*-breaks* $\mathcal{G}(1^k)$ *then if* $t(k)$ *is polynomial then* $\epsilon(k)$ *is negligible.*

As usual, we might want to consider stronger notions of security, such as security against chosen-ciphertext attacks, non-malleability, and known-plaintext security. However, in view of the nature of some public key encryption systems, we might also be interested in a *weaker* definition of security, in which we require that $\mathcal{D}(\mathcal{E}(x)) = x$ only with some specified probability (not necessarily 1).

To see why this could be useful, it is necessary to know a little bit about how RSA works. RSA needs to use two primes p, q. It then defines $\mathcal{E}(x) = x^e \pmod N$ where $N = pq$ and e is some fixed constant. For this to be hard to invert, N must be hard to factor, so p, q must be large (say 100 digits). The usual method of generating large primes is to use the probabilistic primality testing algorithm of Miller and Rabin on large integers. If this algorithm makes a mistake, and p, q are in fact composite, then it is possible that $\mathcal{E}(x) = \mathcal{E}(y)$ for $x \neq y$, so we might compute the wrong pre-image when inverting $\mathcal{E}$.

5.2 Trapdoor permutations and one-way functions

The building blocks for achieving secure encryption schemes are trapdoor permutations. The idea behind them is simple: computing $f(x)$ from x should be easy, but the other way round should be hard — unless you know the secret "trapdoor" (see Figure 6).

Definition 5.4 *A* trapdoor permutation generator $\mathcal{F}$ *is a probabilistic algorithm such that* $(f, f^{-1}) = \mathcal{F}(1^k, r)$, *where* f *is a permutation on* $\{0,1\}^{n(k)}$ *and* f^{-1} *is its inverse.*

Note that RSA does not satisfy this definition since it is defined over $\mathbb{Z}_N^*$, which contains "holes" (i.e., not every integer in $[0, N]$); also, N must be prime and hence cannot be a power of 2. However, there are ways of fixing this, which we shall not go into.

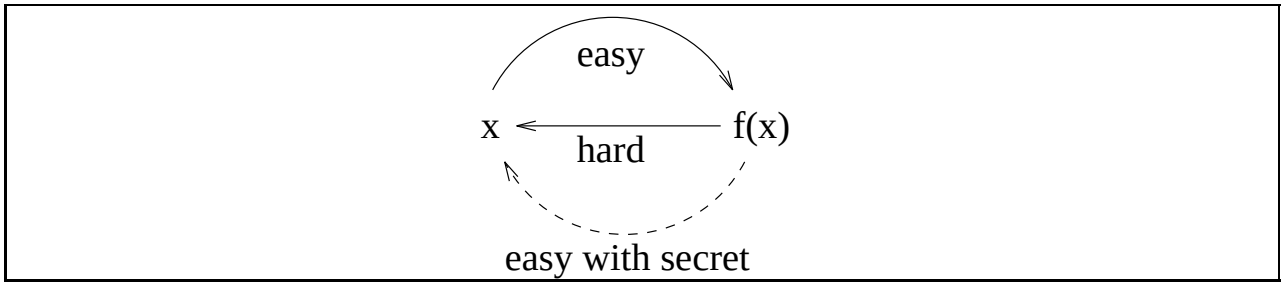

Figure 6: A trapdoor permutation.

Definition 5.5 *An adversary $\mathcal{I}$ (for Inverter or Ingrid)* (t, ϵ)-inverts $\mathcal{F}(1^k)$ *if*

$$\begin{aligned} \mathrm{Adv}_I(k) &\stackrel{\text{def}}{=} \Pr\left[(f, f^{-1}) \leftarrow \mathcal{F}(1^k);\ x \leftarrow \{0,1\}^k;\ y = f(x) : \mathcal{I}(f, y) = x\right] \\ &\geq \epsilon \end{aligned}$$

Definition 5.6 *$\mathcal{F}$ is a* secure trapdoor permutation *if $\mathcal{F}(1^k)$, f and f^{-1} run in poly(k) time, and for all non-uniform adversaries $\mathcal{I}$, if $\mathcal{I}$ (t, ϵ)-inverts $\mathcal{F}(1^k)$, then if $t(k)$ is polynomial then $\epsilon(k)$ is negligible.*

Note that the security of a trapdoor permutation rests solely on the inability of an adversary to invert the encryption function. To simplify proofs, we shall usually just assume the existence of a decryption function, and focus on the essential requirement of the encryption function, which is that it be hard to invert. This notion is captured in the following definition.

Definition 5.7 *$f : \{0,1\}^* \to \{0,1\}^*$ is a* (strong) one-way function *if for all PPT adversaries $\mathcal{I}$*

$$\mathrm{Adv}_I(k) \stackrel{\text{def}}{=} \Pr\left[x \leftarrow \{0,1\}^k : \mathcal{I}(f(x)) \in f^{-1}(f(x))\right]$$

is negligible.

Note that this definition allows for the fact that f many be one-to-many.

We can make a one-way function g from a trapdoor permutation $\mathcal{F}$ as follows. Use the first few bits of the argument to g (call it x) as the coins to be given to $\mathcal{F}$, and then apply the resulting function f to the remaining bits of x. (We can always find a security parameter which is small enough so that the number of coins needed by $\mathcal{F}$ is much less than $|x|$.)

5.3 Hard core bits

As was mentioned in the introduction to this section, we would like to ensure that it is hard to compute every bit of x given $\mathcal{E}(x)$. (See Figure 7.) This notion is formalised as follows.

Definition 5.8 *B_f is a* hard-core bit *of a one-way function f if for all non-uniform adversaries A_B*

$$\mathrm{Adv}_{A_B}(k) \stackrel{\text{def}}{=} \Pr\left[x \leftarrow \{0,1\}^k;\ y = f(x) :\ A_B(f, y) = B(x)\right] - \frac{1}{2}$$

is negligible.

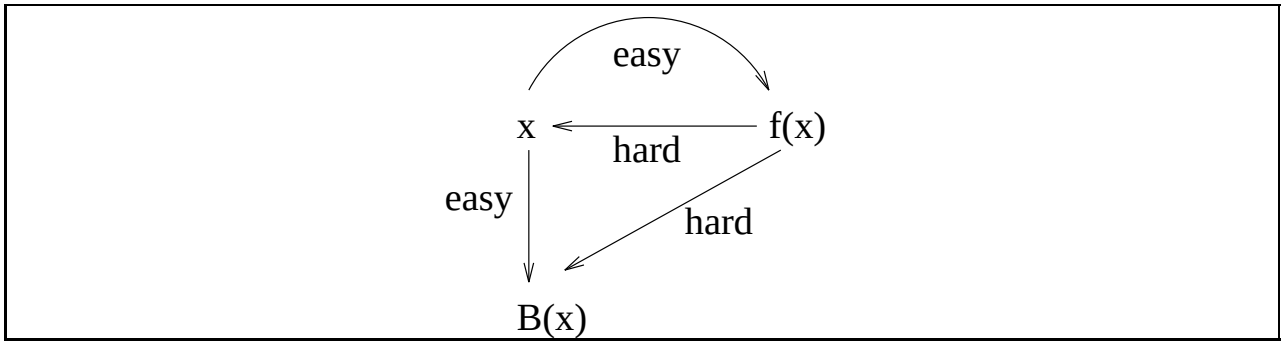

Figure 7: A hardcore bit.

The following theorem, due to Goldreich and Levin (1989), shows that if $f_1(x)$ is a strong one-way function, then the parity of a random subset of bits of x is a hard-core bit.

Theorem 5.9 *Let f_1 be a strong one-way function. Let $f_2(x,p) \stackrel{\text{def}}{=} (f_1(x),p)$, where $|x| = |p| = n$. Then*

$$B(x,p) \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i p_i \pmod 2$$

is hard-core for f_2.

The simplified proof of this, due to Venkatesan and Rackoff, as presented by Ostrovsky in his Berkeley cryptography class, is included in the appendix.

5.4 The GM scheme for secure encryption

Goldwasser and Micali (ref?) proposed the following scheme for secure encryption. $\mathcal{E}_f(x_1 \ldots x_n) = f(r_1) \cdot \ldots \cdot f(r_n)$, where each r_i is randomly chosen from the domain of f subject to $B_f(r_i) = x_i$. That is, to encrypt a bit b, we perform the following algorithm:

```
                    repeat
r ← {0,1}^k
until B(r) = b
E_f(b) = f(r)
```

Using the formula

$$\sum_{k=0}^{\infty} k x^k = \frac{x}{(1-x)^2}$$

with $x = \frac{1}{2}$ we find that the expected number of iterations of this loop is 1. The corresponding decryption function is to compute the ith bit of $B(f^{-1}(f(r_i)))$ for each i. As usual, we shall focus our attention solely on the encryption function.

Mark came up with the following scheme, which will only take 1 time step even in the worst case (at the cost of sending one extra bit). Let $r \leftarrow \{0,1\}^k$. Then $\mathcal{E}_f(b) = (f(r), B(r) \oplus b)$. Intuitively, this is a secure scheme since to infer bit b, we need to know $B(r)$, which is hard to compute even though we know $f(r)$. Let us now prove this.

Theorem 5.10 *If f is a one-way function, and B is a hard-core bit for f, then the GM scheme is a secure encryption scheme.*

Proof: We shall only prove this for the case that $n(k) = 1$, i.e., for encrypting a single bit. The proof of the more general case involves a probability walk method, and is omitted.

We will prove the contrapositive. That is, we assume that GM is not a secure encryption scheme, and show that B is not a hard-core bit for f.

To say that GM is not a secure encryption scheme means that there exists an adversary A_{enc} whose advantage in defeating GM is not negligible. Substituting into Definition 5.2, this means

$$\text{Adv}_{A_{\text{enc}}} \stackrel{\text{def}}{=} \Pr\left[(x_0, x_1, c) \leftarrow A_{\text{enc}}(f, \mathsf{find});\ b \leftarrow \{0,1\};\ r \leftarrow \{0,1\}^k;\ y \leftarrow (f(r), B(r) \oplus x_b) :\ A_{\text{enc}}(c, y) = x_b\right] - \frac{1}{2}$$

is not negligible. (A function g is not negligible if we can find an infinite set of points K and a constant c such that $g(k) > k^{-c}$ for all $k \in K$.)

Given A_{enc}, we must construct an adversary A_B whose advantage in defeating B is not negligible. So suppose A_B is given (f, y), where $y = f(r)$ for some arbitrary $r \in \{0,1\}^k$. We want to find $B(r)$. Let A_B invoke oracle A_{enc} and tell it to generate two distinct plaintext bits, x_0, x_1, along with some state, c. WLOG assume $x_0 = 0$ and $x_1 = 1$. Now let A_{enc} flip a bit, $\beta \leftarrow \{0,1\}$, and compute $x_b = A_B(c, (f(r), x_\beta))$. Suppose $\beta = 0$. If A_{enc} answers $x_b = 0$, then we have $B(r) \oplus x_b = x_\beta = 0$, so A_B should say $B(r) = 0$ (since we trust the A_{enc} oracle). If A_{enc} answers $x_b = 1$, then $B(r) \oplus 1 = 0$, so A_B should say $B(r) = 1$. The following table considers all possible cases, and gives the correct value for $B(r)$.

(x_0, x_1)	$\beta = 0$		$\beta = 1$	
	$x_b = 0$	$x_b = 1$	$x_b = 0$	$x_b = 1$
(0,1)	0	1	1	0
(1,0)	1	0	0	1

The general rule is that A_B should answer $B(r) = x_b \oplus \beta$ if $(x_0, x_1) = (0, 1)$ and $B(r) = \neg x_b \oplus \beta$ otherwise. Thus our advantage against the hard-core bit is the same as our advantage against the encryption scheme, which, by assumption, is not negligible. ■

5.5 More efficient encryption schemes

The GM encryption scheme requires that we encrypt each bit of x separately, and hence make $|x|$ calls to f, which is usually a slow function to compute. There are fasters schemes, which are not only provably secure in the semantic sense, but also in the known-plaintext sense: see [BR94].

However, in practice, public key crytosystems are only used on small texts, such as a key, which is then used for subsequent symmetric encryption.

Proposition 187.
Proof: By appeal to xenophobic instincts. ■

A Proof of Theorem 5.10

Insert Rafi's notes here.

References

[BKR] Bellare, M., J. Kilian and P. Rogaway. "The security of cipher block chaining". Manuscript.

[BR93] Bellare, M. and P. Rogaway. "Entity Authentication and Key Distribution". *Advances in Cryptology — Proc. Crypto '93.*. Springer Verlag.

[BR94] Bellare, M. and P. Rogaway (1994). "A new suggestion for how to encrypt with RSA". *Advances in Cryptology — Eurocrypt '94*. Springer Verlag.

[BS93] Biham and Shamir (1993). *Differential Cryptanalysis of DES*. Springer Verlag.

[CLR90] Cormen, T., C. Leiserson and R. Rivest (1990). *An introduction to algorithms*. MIT Press.

[DH76] Diffie, W. and Hellman, M (1976). "New directions in cryptography". *IEEE Trans. on Info. Theory*, IT-22(6): 644–654.

[GM84] Goldwasser, S. and S. Micali (1984). "Probabilistic Encryption". *J. Computer and System Sciences*, 28.2:270–299.

[MAT93] Matsui. *Eurocrypt* 93.

[MAT94] Matsui. *Crypto* 94.

[Rivest] Rivest, R. (1990). "Cryptography". Chapter 13 of *Handbook of Theoretical Computer Science*, J. v. Leeuwen (ed). Elsevier.

[WC80] Wegman & Carter (1980), "New hash functions and their use in authentication and set equality", *J. of Computer and System Sciences*, 22: 265–279.