**CS 334 Computer Security** **Fall 2014**

# Cryptography Intro and Symmetric Ciphers

*Prof. Szajda*

As always, thanks to Vern Paxson and David Wagner. This handout was written by them (with only minor modifications, if any, by me).

## 1 Brief History of Cryptography

The word "cryptography" comes from the latin root crypt meaning secret, and graphia, meaning writing. So cryptography is literally the study of how to send secret messages. In the next few lectures we shall study encryption schemes as well as some other fundamental goals of cryptography: authentication and digital signatures.

Schemes for sending secret messages go back to antiquity - the Romans used the Caesar cipher, which consists of permuting the alphabet by simply shifting each letter forward by a fixed amount. For example, Caesar used a shift by 3 so the message `cryptography` would be encoded as `fubswrjudskb`. With the developement of the telegraph (wireless communication) at the end of the nineteenth century, the need for encryption in military and diplomatic communications became particularly important. The codes that were used during this "pen and ink" period were relatively simple since messages had to be decoded by hand. The codes were not very secure, especially given modern mathematical and statistical techniques.

The second phase of cryptography, the "mechanical era" was the result of a German project to create a mechanical device for encrypting messages in an unbreakable code. The resulting Enigma machine was a remarkable engineering feat. Even more remarkable was the massive British effort to break the code. The breaking of the Enigma code was an important event that determined the course of World War II, and according to most experts shortened the war by about a year. There were three important factors in the breaking of the Enigma code. First, the British managed to obtain a replica of a working Enigma machine from Poland, which had cracked a simpler version of the code. The second factor was the sophistication of the Allied codebreakers, first the Poles, who employed a large contingent of mathematicians to crack the structure, and then the British, whose massive effort included Alan Turing[1], one of the founding fathers of computer science. The third factor was the scale of the code-breaking effort. The Germans figured that the Enigma was well-nigh uncrackable, but what they didn't figure on was the unprecedented level of commitment the British poured into breaking the Enigma, once the codebreakers made initial progress: at its peak, the British codebreaking organization employed over 10,000 people breaking these codes, a level of effort that vastly exceeded anything the Germans had anticipated.

Modern cryptography is distinguished by its reliance on mathematics and electronic computers. It has its early roots in the work of Claude Shannon following WorldWar II. The analysis of the one-time pad later in this lecture is due to Shannon. The early seventies saw the the introduction of the

[1]Turing, a homosexual, was thanked for his efforts by being charged in 1952 with being a homosexual (a crime, at that time, in Britain). His security clearances were revoked, and he was given the choice of prison or probation with at-home hormonal treatments that reduced libido and caused, among other symptoms, breast enlargement. He chose probation, and apparently committed suicide two years later after ingesting an apple laced with cyanide.

cryptosystem DES by NIST (the National Institute for Standards in Technology). DES answered the growing need for digital encryption standards in banking and other business. The decade starting in the late seventies saw an explosion of work on a computational theory of cryptography.

The most basic problem in cryptography is one of ensuring the security of communications across an insecure medium. The two main members of the cast of characters in cryptography are Alice and Bob who wish to communicate securely as though they were in the same room or were provided with a dedicated, untappable line. In actual fact they have available a telephone line or an internet connection which can be tapped by an eavesdropping adversary, Eve. The goal is to design a scheme for scrambling the messages between Alice and Bob in such a way that Eve has no clue about the content of their exchange. In other words we wish to simulate the ideal communication channel with the available insecure channel.

We will discuss encryption in the next few lectures. Encryption is focused on ensuring the *confidentiality* of communications between Alice and Bob. In the *symmetric-key* model, Alice and Bob share a random key $K$ that is unknown to Eve. Alice encrypts her message $M$ using the key $K$, and Bob decrypts the received ciphertext (to recover the original message) using the same key $K$. The unencrypted message $M$ is sometimes known as the *plaintext*, and its encryption is sometimes called the *ciphertext*.

Let us now examine the threat model, which in this setting involves answering the question, how powerful is Eve? Consistent with Kerkhoff's principle, we will assume that Eve knows the encryption and decryption algorithms[2]. The only information she is missing is the secret key $K$. There are several possibilities about how much access Eve has to the insecure channel:

1. Eve has managed to intercept a single encrypted message and wishes to recover the plaintext (the original message).

2. Eve has intercepted an encrypted message and also has some partial information about the plaintext.

3. Eve can trick Alice to encrypt messages $M_1, M_2, \ldots, M_n$ of her choice (this might happen if Eve has access to the encryption program). At some other point in time, Alice encrypts a message $M$ that is unknown to Eve; Eve has intercepted the encryption of $M$ and would like to recover $M$.

4. Eve can trick Bob into decrypting some ciphertexts $C_1, \ldots C_n$. Eve would like to use this to learn the decryption of some other ciphertext $C$ (different from $C_1, \ldots, C_n$).

5. A combination of cases 3 and 4: Eve can trick Alice into encrypting some messages of Eve's choosing, and can trick Bob into decrypting some ciphertexts of Eve's choosing. Eve could like to learn the decryption of some other ciphertext that was sent by Alice (and in particular did not occur as a result of her trickery).

The first two cases are known as a *ciphertext-only attack* (the difference between them is that in the second case, Eve has partial information, whereas in the first case she does not). The third case is known as a *chosen-plaintext attack*, and the fourth as a *chosen-ciphertext attack*. The fifth

[2]The story of the Enigma gives one possible justification for this assumption: given how widely the Enigma was used, it was inevitable that sooner or later the Allies would get their hands on an Engima machine, and indeed they did. Moreover, to assume that Eve does not know the encryption/decryption algorithms is an example of Security through Obscurity

is known as a *chosen-plaintext/ciphertext attack*, and is the most serious threat model. Today, we usually insist that our encryption algorithms provide security against chosen-plaintext/ciphertext attacks, both because those attacks are practical in some settings, and because we can: it is feasible to provide good security even against this very powerful attack model. However, for this handout we will focus primarily on security against chosen-plaintext attack.

## 2 One Time Pad

The one time pad is a simple and idealized encryption scheme that will help illustrate some important concepts. Alice and Bob share an $n$-bit secret key $K = k_1 \dots k_n$, where the bits $k_1, \dots, k_n$ are picked at random (they are the outcomes of independent unbiased coinflips).

Suppose Alice wishes to send the $n$-bit message $M = m_1 \dots m_n$.

The desired properties of the encryption scheme are:

1. It should scramble up the message. i.e. map it to a ciphertext $C = c_1 \dots c_n$.
2. Given knowledge of the secret key $K$, it should be easy to recover $M$ from $C$.
3. Eve, who does not know $K$, should get no information about $M$.

The encryption scheme is very simple: $c_j = m_j \oplus k_j$, where $\oplus$ is the xor or exclusive-or of the two bits (0 if the two bits are the same and 1 if they are different).

Decryption is equally simple: $m_j = c_j \oplus k_j$.

To sum up, the one-time pad (or any symmetric encryption scheme) is described by specifying three procedures:

- Key generation: Alice and Bob pick a shared random key $K$.
- Encryption algorithm: $C = M \oplus K$.
- Decryption algorithm: $M = C \oplus K$.

Let us now analyze how much information Eve gets about the plaintext $M$ by intercepting the ciphertext $C$. What is the correct measure of this information gained by Eve. It might be the case that Eve had some partial information about $M$ to begin with. Perhaps she knew that the last bit of $M$ is a 0 or that 90% of the bits of $M$ are 1's, or that $M$ is one of BUY! or SELL! but we do not know which. The security property will be: intercepting the ciphertext $C$ should give Eve no additional information about the message $M$ (i.e., she should not learn any new information about $M$ beyond what she already knew before she intercepted $C$).

Let's prove that the one-time pad meets this security property. Consider the following experiment. Suppose Alice has sent one of two messages $M_0$ or $M_1$, and Eve has no idea which was sent. Eve tries to guess which was sent by looking at the ciphertext. We will show that Eve's probability of guessing correctly is 1/2, which is no different than it would be if she had not intercepted the ciphertext at all.

The proof is very simple: for a fixed choice of plaintext $M$, every possible value of the ciphertext $C$ can be achieved by an appropriate and unique choice of the shared key $K$: namely $K = M \oplus C$.

Since each such key value $K$ is equally likely, it follows that $C$ is also equally likely to be any $n$-bit string. Thus Eve sees a uniformly random $n$ bit string no matter what the plaintext message was, and thus gets no information about the plaintext.

Heres another way to see that Eve's probability of guessing successfully is $1/2$. Suppose Eve observes the ciphertext $C$, and she knows that the message $M$ is either $M_0$ or $M_1$ but she does not know which. The probability space here has size $2^{n+1}$: it represents the $2^n$ choices for the $n$-bit key $K$, as well as Alice's choice of whether to send $M_0$ or $M_1$. All $2^{n+1}$ choices are equally likely. We can imagine that Alice and Bob randomly and uniformly choose the key $K$; then Alice randomly chooses a bit $b \in \{0,1\}$, and Alice sends the encryption of $M_b$. So, if Eve observes the ciphertext is some specific value $C$, what is the conditional probability that $b = 0$ given her observation? It is

$$Pr[b = 0 | \text{ciphertext} = C] = \frac{Pr[b = 0 \wedge \text{ciphertext} = C]}{Pr[\text{ciphertext} = C]} = \frac{Pr[b = 0 \wedge K = M^0 \oplus C]}{Pr[\text{ciphertext} = C]} = \frac{1/2^{n+1}}{1/2^n} = \frac{1}{2}.$$

The one time pad has a major drawback. As its name suggests, the shared key cannot be reused to transmit a new message $M'$. If the key $K$ is reused, then Eve can take the xor of the two ciphertexts $C = M \oplus K$ and $C' = M' \oplus K$ to obtain $M \oplus M'$. This gives partial information about the two messages. In particular, if Eve happens to learn $M$, then she can deduce the other message $M'$. Actually in this case she can reconstruct the key $K$. Moreover, there are well-known methods for reconstructing two messages given their xor (see e.g., *FISH and I*, by W. J. Tutte, available on the web).

# 3 Block Ciphers

In symmetric encryption schemes, Alice and Bob share a random key and use this single key to repeatedly exchange information securely despite the existence of an eavesdropping adversary, Eve. The block cipher is a fundamental building block in implementing a symmetric encryption scheme.

In a block cipher, Alice and Bob share a $k$ bit random key $K$, and use this to encrypt an n bit message into an $n$ bit ciphertext. In mathematical notation this can be said as follows. There is an encryption function $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. Once we fix the key $K$, we get a function mapping $n$ bits to $n$ bits: $E_K : \{0,1\}^n \to \{0,1\}^n$ defined by $E_K(M) = E(K, M)$. $E_K$ is required to be a permutation on the $n$ bit strings. The inverse mapping of this permutation is the decryption algorithm $D_K$. Decryption is the reverse of encruption: $D_K(E_K(M)) = M$.

The Advanced Encryption Standard (AES) is an example of a block cipher. It was designed in 1998 by Joan Daemen and Vincent Rijmen, two researchers from Belgium, in response to a competition organized by NIST.

AES uses a block length of $n = 128$ bits and a key length of $k = 128$ bits (it can also support $k = 192$ or $k = 256$ bit keys, as well as various block sizes). It was designed to be extremely fast in both hardware and software. In terms of security, AES has proved to be an impressively strong algorithm. After all these years, the best practical attack known is exhaustive key search, where the attacker systematically tries decrypting some ciphertext with each possible key to see which one gives intelligible plaintext. In the case of AES, exhaustive key search requires $2^{128}$ computations in the worst case ($2^{127}$ on average). This is a large enough number that even the fastest current supercomputers couldn't possibly mount an exhaustive keysearch attack against AES within the lifetime of our Solar system. Thus AES behaves very differently than the one-time pad. Even given a very large number of plaintext, ciphertext pairs, there appears to be no effective way to decrypt

any new ciphertexts. We will formalize these security properties of AES by saying that the function $E_K$ for a randomly chosen key $K$ "behaves like" a random permutation on the $n$-bit strings.

Formally, we shall measure the security of the block cipher by performing the following experiment: suppose that the adversary, Eve, is given a box which contains either (I) the block cipher with a random key or (II) a uniformly random permutation $\pi$ on $n$ bits chosen uniformly at random when the box was created. Eve is allowed $T$ steps in which to play with the box. In each step, Eve can supply an input $x$ to the box and receive a corresponding output $y$ from the box (namely, $y = E_K(x)$ for a type-I box, or $y = (x)$ for a type-II box). After playing with the box, Eve must guess whether the box is type I or type II. The advantage of the adversary Eve is $Adv(\text{Eve}) = |pq|$, where $p$ is the probability that Eve guesses I when the box she is given is actually of type I, and $q$ is the probability that Eve guesses I when the box she is given is actually of type II. Informally the advantage of Eve is the chance that she can distinguish between the block cipher and a truly random permutation. If Eve's advantage is at most $\epsilon$ then we say that the block cipher is $(T, \epsilon)$-secure. For AES, the above discussion says that if Eve wants advantage $\epsilon = 1$, she needs $T \geq 2^{128}$ steps. In general there is a tradeoff between $T$ and $\epsilon$, and so we could expect that $T/\epsilon \geq 2^{128}$ for all successful attacks. In some sense $\log(T/\epsilon)$ is the effective key length of the block cipher.

## 4 Sidebar: Advantage

In the previous section, we introduced the notion of the advantage of an adversary. The *advantage* is a general way of measuring how similar one setting is to another. This concept is often used in formalizing what it means for cryptographic algorithms to be secure, so let's look at it in general.

Suppose that Trevor tosses a fair coin, and Gloria claims that she has some method to predict what outcome will come up, slightly only better than chance. (Maybe Gloria has really sharp eyes and can sometimes spot the head or tail on the coin just before it falls on Trevor's wrist, or something like that.) How can we quantify how much better Gloria is at guessing than just blind guessing?

The obvious way is to compute the probability that Gloria guesses correctly. There's nothing wrong with that measure. But in some cases it's a bit easier to work with the advantage, which we define as $Adv(\text{Gloria}) = |pq|$, where $p =$ the probability that Gloria guesses Heads when the coin actually lands Heads, and $q =$ the probability that Gloria guesses Heads when the coin actually lands Tails. Note that the advantage is a number between 0 and 1; if the advantage is 0, it means Gloria is not doing any better than blind guessing, and if the advantage is 1, it means that Gloria always gets it right. The advantage is a way of measuring Gloria's edge: if Gloria always guesses Heads, her advantage is zero, and if she always guesses Tails, her advantage is zero, but if her guess is correlated to the true outcome, her advantage will be greater than zero.

How does the advantage relate to the probability that Gloria guesses correctly? It turns out the two measures are interchangeable:

$$Adv(\text{Gloria}) = 2[Pr[\text{Gloria guesses correctly}] - 1/2]$$

and

$$Pr[\text{Gloria guesses correctly}] = \frac{1}{2} \pm \frac{Adv(\text{Gloria})}{2}.$$

So the advantage is just a rescaled version of the probability that Gloria guesses correctly. There's no fundamental reason why we need both measures—but sometimes it is a little more convenient to work with the advantage, so it's useful to be familiar with the concept.

We often use the advantage to measure how "similar" (or "distinguishable") two things are. If Eve has advantage at most $\epsilon$ of guessing whether she is interacting with a type-I or type-II box, for some small $\epsilon$, then it follows that type-I boxes are basically indistinguishable from type-II boxes: e.g., any place where a type-II box is secure, a type-I box will also be secure. We will often compare a real scheme to an ideal (but unimplementable) model of what the scheme ought achieve; if Eve's advantage at distinguishing these two is guaranteed to be small, then the real scheme is just as good as the idealization.

# 5 Symmetric Encryption Schemes

A symmetric encryption scheme allows Alice and Bob to privately exchange a sequence of messages in the presence of an eavesdropper Eve. We will assume that Alice and Bob share a random secret key K. How Alice and Bob managed to share a key without the adversary's knowledge is not going to be our concern here. The encryption scheme consists of an encryption algorithm $\mathcal{E}$ that takes as input the key $K$ and the plaintext message $M \in \{0,1\}^*$, and outputs the ciphertext. The decryption algorithm $\mathcal{D}$ takes as input the key and the ciphertext and reconstructs the plaintext message $M$. In general the encryption algorithm builds upon a block cipher to accomplish two goals: one is to show how to encrypt arbitrarily long messages using a fixed length block cipher. The other is to make sure that if the same message is sent twice, the ciphertext in the two transmissions is not the same. The encryption algorithm to achieve these goals can either be randomized or stateful — it either flips coins during its execution, or its operation depends upon some state information. The decryption algorithm is neither randomized nor stateful.

ECB Mode (Electronic Code Book): In this mode the plaintext $M$ is simply broken into $n$ bit blocks $M_1, M_2, \ldots, M_l$, and each block is encoded using the block cipher: $C_i = E_K(M_i)$. The ciphertext is just a concatenation of these individual blocks: $C = C_1 \cdot C_2 \cdots C_l$. This scheme is adequate for simple tasks such as encrypting PINs for cash machine systems. However any redundancy in the blocks will show through and allow the eavesdropper to deduce information about the plaintext. We will discuss this in more detail after formalizing the notion of the security of a symmetric encryption scheme below.

CBC Mode (Cipher Block Chaining): This is a popular mode for commercial applications. A random $n$ bit string, the initial vector or IV is selected. Define $C(0) = E_K(IV)$. The $i$th encrypted block $C_i = E_K(C_{i-1} \oplus M_i)$. The ciphertext is the concatenation of the initial vector and these individual blocks: $C = IV \cdot C_1 \cdot C_2 \cdots C_l$ . The CBC mode does provide strong security guarantees on the privacy of the plaintext message. This is formalized later in this handout.

OFB Mode (Output Feedback Mode): In this mode, the initial vector IV is repeatedly encrypted to obtain a set of values $Z_i$ as follows: $Z_0 = \text{IV}$ and $Z_i = E_K(Z_{i1})$. These values $Z_i$ are now used as keys for a one-time pad, so that $C_i = Z_i \oplus M_i$ . The ciphertext is the concatenation of the initial vector and these individual blocks: $C = \text{IV} \cdot C_1 \cdot C_2 \cdots C_l$. In OFB mode, it is very easy to tamper with ciphertexts. For instance, suppose that the adversary happens to know that the jth block of the message, $M_j$, specifies the amount of money being transferred to his account from the bank, and suppose she also knows that $M_j = 100$. Since she knows both $M_j$ and $C_j$, she can determine $Z_j$. She can then substitute any $n$-bit block in place of $M_j$ and get a new ciphertext $C'_j$ where the 100 is replaced by any amount of her choice. This kind of tampering is also possible with other modes of operation as well (so don't be fooled into thinking that CBC mode is safe from tampering); its just easier to illustrate on OFB mode.

Counter Encryption: One drawback of the CBC mode is that successive blocks must be encrypted sequentially. For high speed applications it is useful to parallelize these computations. This is easily achieved by encrypting a counter initialized to IV to obtain a set of keys for a one-time pad: namely $Z_i = E_K(IV + i)$ and $C_i = Z_i \oplus M_i$.

# 6 Notions of Security

CBC mode can be shown to be secure against chosen-plaintext attacks. Let us first define what it means for a symmetric encryption scheme to be secure against chosen-plaintext attack. We consider the following experiment: Eve, the adversary, has access to a box which takes as input a pair of same-length plaintext messages $(M, M')$; if the box is of type I, it outputs the CBC encryption of $M$, and if it is of type II, it outputs the CBC encryption of $M'$. The box has a key $K$ hardcoded in it that was chosen randomly and secretly when the box was manufactured. Eve is allowed to play with the box for the available time and must guess which type it is. As before, Eve's advantage is defined to be

$$Adv(\text{Eve}) =$$
$$Pr[\text{Eve guesses I}|\text{Eve is given a type-I box}]Pr[\text{Eve guesses I}|\text{Eve is given a type-II box}].$$

If we can show that Eve's advantage is negligibly small, for every possible algorithm or strategy that Eve might use[3], then we will have shown that the encryption scheme is secure.

ECB mode encryption is not secure in this sense. Let $x$ and $y$ be any two distinct $n$-bit strings. Let $M = x \cdot x$ and $M' = x \cdot y$. Then Eve can use the following attack strategy: submit the pair $(M, M')$ to her box, and then check whether the output from the box is of the form $a \cdot a$ or $a \cdot b$. In the former case (where the box outputs a $2n$-bit ciphertext whose first $n$ bits are the same as the last $n$ bits), then Eve can infer she is interacting with a type-I box; in the latter case, Eve can infer she is interacting with a type-II box. This is an attack that gives advantage 1, and thus demonstrates that ECB is not secure against chosen-plaintext attacks.

It can be proven that CBC mode encryption is secure under chosen plaintext attack. If the block cipher is invoked a total of $j$ times by the box during the $T$ units of time that Eve plays with it, and if the block cipher is $(T, \epsilon)$-secure, then $Adv(\text{Eve}) \leq \epsilon + 2j^2/(2^n j)$.

[3]To be precise, we only consider attack algorithms that terminate in some finite time. For instance, we might ignore attacks that require 2200 steps of computation as being totally unreasonable—they will not finish within the lifetime of the Solar system.