YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CPSC 467a: Cryptography and Computer Security

Notes 2 (rev. 1)

Professor M. J. Fischer *September 6, 2005*

Lecture Notes 2

1 Classical cryptography

A *symmetric cryptosystem* consists of a set $\mathcal{M}$ of *plaintext messages*, a set $\mathcal{C}$ of *ciphertexts*, a set $\mathcal{K}$ of keys, and two functions, an *encryption* function $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ and a *decryption* function $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$. We often write $E_k(m) = E(k, m)$ and $D_k(c) = D(k, c)$ and may talk about a cryptosystem consisting of a family of encryption and decryption functions indexed by k, but this should not obscure the fact that in order to be useful, it must be feasible to evaluate the functions given an arbitrary key k.

We require that D_k be the (one-sided) inverse of E_k, that is, for all keys $k \in \mathcal{K}$ and all plaintext messages $m \in \mathcal{M}$, we have $D_k(E_k(m)) = m$.

To be useful, it should certainly be difficult to find k given only $c = E_k(m)$. But we generally want much more. The goal is to hide m, not k, so a procedure that found m given c would compromise the cryptosystem even if k remained well hidden. Thus, we want the system to resist a *ciphertext only* attack: the attacker knows c and tries to find m.

In practice, this also might not be enough. The attacker might know a number of pairs plaintext-ciphertext pairs $(m_1, c_1), \ldots, (m_r, c_r)$ where $c_i = E_k(m_i)$ for $i = 1, \ldots, r$, in addition to knowing c. We want it to be difficult for the attacker to find m in this case also. Thus, we want the system to resist a *known plaintext* attack.

In many contexts, one needs even stronger forms of attack resistance, where the attacker not only knows a number of plaintext-ciphertext pairs but also gets to choose the plaintext or the cipher text, or both. Where the attacker gets to experiment with the encryption system, it should still be difficult to find the plaintext corresponding to any ciphertext c that differs from those in the known plaintext-ciphertext pairs. These so-called *chosen plaintext*, *chosen ciphertext*, and *chosen text* attacks are much stronger than the simple ciphertext only attack and much harder to defend against. For example, in a chosen ciphertext attack, the attacker would be allowed to obtain the decryption of $c + 1$ when trying to decrypt c.

1.1 Caeser cipher

The Caeser cipher is said to go back to Roman times. Let the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ consist of strings of uppercase letters from the Roman alphabet and spaces. Number the letters of the alphabet starting at 0, so $A = 0$, $B = 1, \ldots, Z = 25$. The key space $\mathcal{K} = \{0, \ldots, 25\}$. To encrypt, E_k replaces each letter m by the letter $(m + k) \bmod 26$.[1] To decrypt, D_k replaces each letter c by the letter $(c - k) \bmod 26$.

Consider the problem of breaking the Caeser cipher. Suppose you intercept the ciphertext JXQ. You quickly discover that $E_3(\text{GUN}) = \text{JXQ}$. But can you conclude that $k = 3$ and GUN is the corresponding plaintext? Upon further investigation, you discover that $E_{23}(\text{MAT}) = \text{JXQ}$. So now you are in a quandary. You have two plausible decryptions and no way to tell for sure which is correct. Have you broken the system or haven't you? According to the "definition" above, you

[1] $x \bmod 26$ is the remainder of x divided by 26.

haven't since you haven't found the plaintext with certainty. But you've certainly inflicted serious damage. You've reduced the possible set of 3-letter words that the message might have been down to a small set of possibilities. In many contexts, this is nearly as good as having completely broken the system. This suggests that a formal definition of security must take into account the possibility of "partially breaking" a system, as we did here.

The longer the message, the more likely that only one key will lead to a sensible message. For example, if the ciphertext were "EXB JXQ", you would know that $k = 3$, since $D_3(\text{EXB JXQ}) =$ BUY GUN, whereas $D_{23}(\text{EXB JXQ}) =$ HAE MAT is nonsense.

At the opposite extreme, consider the Caeser cipher with a one-letter message. If I tell you that the ciphertext is V, you learn absolutely nothing about the plaintext. You reason as follows: If $k = 0$, then $m =$ V. If $k = 1$, then $m =$ U. If $k = 2$, then $m =$ T. And so on through the alphabet. For every possible plaintext message m, there is exactly one key for which m encrypts to V.[2] So knowing that $c =$ V doesn't give the attacker any information about the plaintext m.

We say such a system is *information-theoretically secure*. This means that there is no statistical correlation between the plaintext and the ciphertext. To make this definition precise, one must assume that the key k is chosen uniformly at random and is unknown to the attacker. What the attacker sees is a random variable $c = E_k(m)$, where m is a constant and k a uniformly distributed random variable. c is identically distributed for each choice of m; hence, the distribution of c does not depend on m, so knowing c gives no information about m.

Thus, we see that the same Caeser cipher can be information-theoretically secure, or partially secure, or completely breakable, depending on the length of the message to which it is applied.

2 Unbreakable Codes

A code can be hard for Eve to break for two different reasons: (i) She may not have enough information to uniquely determine the plaintext, or (ii) she may have enough information, but it takes her too much time to find the plaintext. In the first case, we say the code is *information-theoretically secure*. In the second, it is *computationally secure*. The first notion is based on statistical inference; the second on computational complexity.

2.1 Statistical inference

Statistics is about guessing underlying parameter values from samples drawn from a random distribution. For example, given a biased coin with probability p of heads, one wants to determine p by observing the outcomes of several coin flips. Of course, one can never determine p exactly, so one tries instead to guess a value p' that is "likely" to be "close" to p.

Eve's job in attempting to break a cryptosystem is analogous to that of the statistician. The analog of a coin flip is for Alice to choose a random key k and compute $c = E_k(m)$. Eve tries to guess a value m' that is "likely" to be "close" to m. Unlike the coin-flip example, Eve generally will only get one sample for a given message m. Neverless, the same notions apply, and the closer Eve can get to finding the true message m', and the more likely she is to be correct, the weaker our cryptosystem is.

The strongest case of security is where Eve can get no information at all about m. This case obtains when c is completely independent of m, that is, when the distribution of c does not depend on m. Recall that a (discrete) probability distribution is a function that assigns a probability $p(x)$ to each point x in the sample space. In our case, each value of m results in a potentially different

[2] Why does it matter that there is exactly one?

probability distribution $p_m(c)$, where $p_m(c)$ is the probability of choosing a key for which $E_k(m) = c$. To say that the distribution of c does not depend on m is the same as saying that all of the functions $p_m()$ are the same.[3]

For the Caeser cipher with one-letter messages, we have $p_m(c) = 1/26$ for each possible ciphertext letter c. This is because there are 26 equiprobable keys, and each key maps m to a different letter. In this case, c is independent of m, and therefore knowledge of c gives no information about m. We say the system is *information-theoretically secure*.

For the Caeser cipher with two-letter messages, we lose this independence. For example, let $c = \texttt{VV}$. Then $p_{\texttt{AA}}(\texttt{VV}) = 1/26$, but $p_{\texttt{BC}}(\texttt{VV}) = 0$. As functions, $p_{\texttt{AA}}() \neq p_{\texttt{BC}}()$, and Eve can get information about m from observing c. In particular, Eve knows that m is one of the messages in the set $\{\texttt{AA}, \texttt{BB}, \ldots \texttt{ZZ}\}$.

2.2 One-time pad

A *one-time pad* is a simple information-theoretically secure cryptosystem based on the bitwise *exclusive-or* operator (XOR), which we write as $\oplus$. Recall for Boolean values x and y that $x \oplus y$ is true when exactly one of x and y is true, but is false if x and y are either both false or both true. Exclusive-or is therefore equivalent to sum modulo two, when true is represented by 1 and false by 0. In symbols,

$$x \oplus y = (x + y) \bmod 2.$$

With a one-time pad, the plaintext, ciphertext, and key are all assumed to be binary strings of the same length. The encryption function is $E_k(m) = k \oplus m$, where $\oplus$ is applied componentwise to corresponding bits of k and m. It's a basic fact of mod 2 addition that addition and subtraction are the same. This implies that E_k is its own inverse; hence $D_k(c) = E_k(c)$ is the decryption function.

$$D_k(E_k(m)) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = 0 \oplus m = m.$$

Like the one-letter Caeser cipher, the simple XOR cryptosystem has the property that for every ciphertext string c and every plaintext m, there is exactly one key k such that $E_k(m) = c$ (namely, $m \oplus c$). Hence, every ciphertext is equally likely no matter what m is, so the one-time pad is information-theoretically secure.

The simple XOR cryptosystem would seem to be the perfect cryptosystem. It works for messages of any length (by choosing a key of the same length). It is simple, easy to encrypt and decrypt, and information-theoretically secure. In fact, it is sometimes used for highly sensitive data. However, it suffers from two major weaknesses that make it not so useful in practice:

1. The key k must be as long as the message to be encrypted. Key management and key distribution are both difficult problems which we will be discussing later in the course. The longer the keys, the more difficult these problems become.

2. The same key must never be used more than once. (Hence the term "one-time".) The reason is that the simple XOR cryptosystem immediately succumbs to a known plaintext attack. If Eve knows just one plaintext-ciphertext pair (m_1, c_1), then she can easily compute $k = m_1 \oplus c_1$, totally breaking the system and allowing her to decrypt all future messages sent with that key. Even in a ciphertext-only situation, if Eve has two ciphertexts c_1 and c_2 encrypted by the

[3] One often assumes that the parameters themselves are also drawn from some distribution. This enables one to talk, for example, about the probability that the ciphertext = "QBA". But it seems somewhat counterintuitive to assume that Alice merely sends random message according to some fixed (but unknown) probability distribution. Therefore, we prefer to avoid such assumptions whenever possible.

same key k, she can gain significant partial information about the corresponding plaintexts m_1 and m_2. In particular, she can compute $m_1 \oplus m_2$, since

$$m_1 \oplus m_2 = (c_1 \oplus k) \oplus (c_2 \oplus k) = c_1 \oplus c_2.$$

That information, together with other information she might have about the likely content of the messages, may be enough for her to seriously compromise the secrecy of the data.

3 Theoretically Breakable Codes

For any fixed message m, the number of possible ciphertexts is at most the size of the key space. (It will be even less if two or more keys map m to the same ciphertext, which is perfectly possible.) Often the message space and the ciphertext space are the same, in which case $E_k(m)$ is a bijection for each fixed value of k. From these two facts, it follows that for each fixed ciphertext c, the number of possible plaintext messages is at most $|\mathcal{K}|$. Hence, knowing the ciphertext c immediately reduces the uncertainty about m from being a member of a set of size $|\mathcal{M}|$ to a member of a set of size $|\mathcal{K}|$. The key spaces of most codes used in practice are much smaller than the space of possible messages and ciphertext. Hence, information-theoretically, Eve gains a lot of information about m just knowing c. Moreover, if the set of meaningful messages is sparse in $\mathcal{M}$ and not correlated with the structure of E_k (the usual case), then it highly likely that the only meaningful message that maps to c is Alice's original message m.

3.1 Brute force attacks

A *brute force attack* is one that tries all possible keys k. For each k, Eve computes $m_k = D_k(c)$ and tests if m_k is meaningful. If so, then m_k is a plausible decryption of c. If exactly one m_k is found that is meaningful, then Eve knows that $m_k = m$.

Given long enough messages, the Caeser cipher is easily broken by brute force—one simply tries all 26 possible keys and sees which leads to a sensible plaintext. The Caeser cipher succumbs because the key space is so small.

With modern computers, it is quite feasible for an attacker to try millions ($\sim 2^{20}$) or billions ($\sim 2^{30}$) of keys. Of course, the attacker also needs some automated test to determine when she has a likely candidate for the real key, but such tests are often easy to produce given a little knowledge of the probable message space.

How big is big enough? The DES (Data Encryption Standard) cryptosystem (which we will talk about shortly) has only 56-bit keys for a key space of size 2^{56}. A special DES Key Search Machine was built as a collaborative project by Cryptography Research, Advanced Wireless Technologies, and EFF. (See http://www.cryptography.com/resources/whitepapers/DES.html for details.) This machine was capable of searching 90 billion keys/second and discovered the RSA DES Challenge key on July 15, 1998, after searching for 56 hours. The entire project cost was under $250,000.

Today, 80-bit keys are probably still safe, but only barely so. 2^{80} is only about 16 million times as large as the DES key space, and tens of thousand of machine on the internet could conceivably be deployed in breaking a code. If not quite feasible using today's PC's (and I'm not saying it isn't), it's not that far-fetched to imagine searching an 80-bit key space in the foreseeable future. Much better are systems like triple DES (with 112-bit keys) and AES (with 128-bit keys), which will probably always be safe from brute-force attacks (but not necessarily from other kinds of attacks).