YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

CPSC 467a: Cryptography and Computer Security *Notes 3 (rev. 2)*
Professor M. J. Fischer *September 10, 2008*

Lecture Notes 3

13 Caveats with perfect secrecy

In Section 12, we saw how a seemingly minor change in a cryptosystem changed it from one that had perfect secrecy to one that leaked a considerable amount of information to Eve. Perfect secrecy, while perhaps difficult to obtain, might seem like the ideal that one should strive for. We make two observations to show that even a scheme with perfect secrecy is not without defects and must still be used carefully.

First of all, our simplified Caesar cipher succumbs immediately to a known plaintext attack, since if one knows even a single plaintext-ciphertext pair (m_1, c_1), one can easily solve the equation $c_1 = E_k(m_1) = (m_1 + k) \bmod 3$ to find the key $k = (c_1 - m_1) \bmod 3$. Hence, any subsequent ciphertext $c = E_k(m)$ is immediately decrypted using $D_k()$ and the system is completely broken.

Second, a system with perfect secrecy can be subject to a modification attack whereby an attacker who can both read and alter messages en route can modify the contents of a message in specific semantically-meaningful ways even though he has no idea what the message actually is. We refer to such an active attacker as "Mallory", and we call such an attack a *man-in-the-middle* attack.

Here's what Mallory could do to the one-letter Caesar cipher (where we now return to the original version that works over the full 26-letter alphabet). Suppose Alice sends c to Bob. Mallory intercepts it and changes c to $(c + 5) \bmod 26$. Even though he doesn't know the key and cannot read m, he knows that his change will alter m in a similar way, changing m to $(m + 5) \bmod 26$. Why? Let's do the calculations, where all arithmetic is done modulo 26:

$$D_k(c') = D_k(c + 5) = c + 5 - k = D_k(c) + 5 = m + 5.$$

Depending on the application, this could be a devastating attack. Suppose Alice were a financial institution that was making a direct deposit of m thousand dollars to Mallory's bank account at the Bob bank. By this attack, Mallory could get an extra 5 thousand dollars put into his account each month.

For another application, note that the English vowels are all represented by even numbers in our encoding scheme where we number the letters beginning with 0. Hence, $A = 0$, $E = 4$, $I = 8$, $O = 14$, and $U = 20$. Hence, if m is a vowel, then the altered message m' is guaranteed to not be a vowel. Suppose Alice were a general sending an order to a field commander whether or not to attack. If an attack is to her advantage, she orders an attack by sending a vowel; otherwise, she withholds the attack by sending a consonant. She thinks she is adding yet another layer of security by encoding the attack bit in such a non-obvious way. However, Mallory's $c + 5$ transformation changes every attack message to "don't attack" (and some "don't attack messages to "attack"). This effectively prevents Alice from attacking in those situations where she could win. The fact that she was using a cryptosystem for which perfect secrecy is known did not protect her.

This illustrates once again that the security of a system in practice depends critically on the kinds of attacks available to an attacker. In this case, the cryptosystem that is provably perfectly secure against a passive eavesdropper failed miserably against an active attacker.

14 One-time pad

Before we leave the topic of perfect secrecy, I want to present a well-known information-theoretically secure cryptosystem known as a one-time pad. This is important, both because it is sometimes used in practice, and also because it is the basis for many stream ciphers, where the truly random key is replaced by a pseudo-random bit string.

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 Caesar 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 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.

15 Security of the Caesar cipher

We return now to the full Caesar cipher, extended to encrypt string of letters, not just single letters. Recall that if m is an r-letter message $m_1 \ldots m_r$, then

$$E_k^r(m_1 \ldots m_r) = E_k(m_1) \ldots E_k(m_r)$$

is an r-letter ciphertext $c_1 \ldots c_r$, and

$$D_k^r(c_1 \ldots c_r) = D_k(c_1) \ldots D_k(c_r).$$

That is, we extend the basic 1-letter Caesar cipher by applying it repeatedly to each letter of the message, using the same key each time.

Although the Caesar cipher is not practical because of its small key space, many practical ciphers share the basic Caesar cipher's property that they are build to work only on fixed-sized blocks of data (typically bitstrings of some convenient length such as 64 or 128). To send longer messages, the cipher is used repeatedly according to some rule. The simplest is to break the long message into blocks and encrypt each block separately, as we're doing in the full Caesar cipher. Using a block cipher in this way is called "Electronic Code Book" mode, or ECB.

Consider now the problem of breaking the Caesar 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 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.

Thus, we see that the Caesar cipher can be information-theoretically secure when $r = 1$, but for longer r it is only partially secure or completely breakable, depending on the length of the message to which it is applied and the redundancy present in that message.[1]

16 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 Caesar cipher is easily broken by brute force—one simply tries all 26 possible keys and sees which leads to a sensible plaintext. The Caesar 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

[1]There is a whole theory of redundancy of natural language that allows one to calculate a number called the "unicity distance" for a given cryptosystem. If a message is longer than the unicity distance, there is a high probability that it is the only meaningful message with a given ciphertext and hence can be recovered uniquely, as we were able to recover "BUY GUN" from the ciphertext "EXB JXW" in the example. See [Sti06, section 2.6] for more information on this interesting topic.

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 machines 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).

17 Cryptanalysis of the Caesar Cipher

In the previous lecture, we discussed what a brute force attack on the Caesar cipher would look like: Decrypt the ciphertext using each of the 26 possible keys, and see which decryption "looks like" a valid message. With English-language messages, this works quite well in practice—generally all of the decryptions but the correct one will be immediately recognized as garbled nonsense. The only sensible-looking decryption is therefore the correct one.

A cryptanalyst might wish to automate the whole process. The question is, how does one write a program to distinguish valid English sentences from gibberish? One could imagine applying all sorts of complicated natural language processing techniques to this task. However, much simpler techniques can be nearly as effective.

One simplification is to consider random messages whose letter frequencies are similar to that of valid English sentences. That is, for each letter k, let p_k be the probability (relative frequency) of that letter in normal English text. Then a message $m = m_1 m_2 \ldots m_r$ has probability $p_{m_1} \cdot p_{m_2} \cdots p_{m_k}$. This is the probability of m being generated by the simple process that chooses r letters one at a time according to the probability distribution $(p_0, \ldots, p_{25})$.

Now, assume that Eve obtains a ciphertext $c = E_k(m)$, where she knows that m was chosen randomly as described above and k is a uniformly distributed Caesar cipher key. Eve easily computes the 26 possible plaintext messages $D_0(c), ..., D_{25}(c)$, one of which is correct. To choose which, she computes the conditional probability of each message given c, then picks the message with the greatest probability. This guess will not always be correct, but for letter distributions that are not too close to uniform and sufficiently long messages, it works correctly with very high probability.

18 Monoalphabetic ciphers

The Caesar cipher uses only the 26 rotations out of the 26! permutations on the alphabet. The *monoalphabetic cipher* uses them all. A key k is an arbitrary permutation of the alphabet. $E_k(m)$ replaces each letter a of m by $k(a)$ to yield c. To decrypt, $D_k(c)$ replaces each letter b of c by $k^{-1}(b)$.

The size of the key space is $|\mathcal{K}| = 26! > 2^{74}$, which is too large for a successful brute force attack. However, monoalphabetic ciphers can be readily broken using letter frequency analysis, given a long enough message. Because each occurrence of a letter a in the message is replaced by the same letter $k(a)$, the most frequently-occurring letter of m will correspond to the most frequently-occurring letter of c. While Eve might not know what the most frequently-occurring letter of m is, if the message is long enough and she knows that it is English, then it is quite likely that the most frequently-occurring letter in m is one of the most frequently-occurring letters in

English, i.e., 'e' or maybe 't'. She can then assume that the most frequent letter b_1 in c is 'e', the next most frequent letter b_2 is 't', and so forth. Of course, not all of these guesses will be correct, but the number of likely candidates for each ciphertext letter is greatly reduced. Moreover, many wrong guesses can be quickly discarded even without constructing the entire trial key because they lead to unlikely letter combinations.

19 Playfair cipher

A cipher that encrypts two letters at a time is much harder to break than a monoalphabetic cipher since it tends to mask the letter frequencies. The *Playfair* cipher, invented by Charles Wheatstone in 1854 but popularized by Lord Lyon Playfair, is one such example [MvOV96, chapter 7, pp. 239-240] and [Wik]. Here, the key is a passphrase from which one constructs a a 5×5 matrix of letters as described below. Pairs of plaintext letters are then located in the matrix and used to produce a corresponding pair of ciphertext letters.

The matrix is constructed by writing the passphrase into the matrix cells from left to right and top to bottom, omitting any letters that have previously been used. It is filled out with the letters of the alphabet that do not occur in the passphrase, in alphabetical order. (In carrying out this process, "I" and "J" are identified, so we are effectively working over a 25-character alphabet.) Thus, each letter of the 25-character alphabet occurs exactly once in the resulting matrix.

For example, the key matrix that results from the passphrase, "CRYPTOGRAPHY REQUIRES STRONG KEYS" is:

$$\begin{array}{ccccc} \mathrm{C} & \mathrm{R} & \mathrm{Y} & \mathrm{P} & \mathrm{T} \\ \mathrm{O} & \mathrm{G} & \mathrm{A} & \mathrm{H} & \mathrm{E} \\ \mathrm{Q} & \mathrm{U} & \mathrm{I/J} & \mathrm{S} & \mathrm{N} \\ \mathrm{K} & \mathrm{B} & \mathrm{D} & \mathrm{F} & \mathrm{L} \\ \mathrm{M} & \mathrm{V} & \mathrm{W} & \mathrm{X} & \mathrm{Z} \end{array}$$

The first 7 letters of "CRYPTOGRAPHY" appear in order. The 8th letter, "R", is omitted since it duplicates the 2nd letter. The 9th letter, "A", is copied, but the 10th letter is again a duplicate. Continuing this way through the passphrase, we see that the last letter used is "K". The matrix is completed with the remaining letters "BDFLMVWXZ" of the alphabet.

A message to be encrypted is first broken up into pairs of letters. For example, the message "MEET ME AT THE SUBWAY" would be broken into the pairs "ME" "ET" "ME" "AT" "TH" "ES" "UB" "WA" "YX". We have padded the message with a trailing "X" in order to make its length even, and spaces have been suppressed. In case a pair of identical letters is about to be produced, an "X" is inserted to prevent that. Thus, the message "A GOOD BOOK" becomes "AG", "OX", "OD" "BO", "OK".

Next, each pair of plaintext letters ab is encrypted. Consider the rectangle with a and b at its corners.

1. If the rectangle is non-trivial (i.e., has at least two rows and columns), replace each of a and b by the letter at the opposite corner of the same row. For example, the encryption of "AT" is "EY" since the rectangle determined by "AT"

$$\begin{array}{ccc} \mathrm{Y} & \mathrm{P} & \mathbf{T} \\ \mathbf{A} & \mathrm{H} & \mathrm{E} \end{array}$$

   has "E" opposite "A" and "Y" opposite "T".

2. If a and b appear in the same row, then replace a by the next letter circularly to its right in the row, and similarly for b. For example, the encryption of "LK" is "KB" since "L" and "K" occur in the same row, "K" is immediately to the right of "L", and "B" is immediately to the right of "K".

3. If a and b appear in the same column, then replace a by the next letter circularly down in the column, and similarly for b.

Applying these rules to our example message, "MEET ME AT THE SUBWAY", we obtain the ciphertext, "ZONEZOEYPEHNBVYIPW".

Decryption is by a similar procedure. In decrypting, one must manually remove the suprious occurrences of "X" and resolve the "I/J" ambiguities.

See Trappe and Washington [TW06] for a discussion of how the system was successfully attacked by French cryptanalyst Georges Painvin and the Bureau du Chiffre.

References

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. ISBN: 0-8493-8523-7. URL `http://www.cacr.math.uwaterloo.ca/hac/`.

[Sti06] Douglas R. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, third edition, 2006. ISBN-10: 1-58488-508-4; ISBN-13: 978-58488-508-5. URL `http://www.crcpress.com/shopping_cart/products/product_detail.asp?sku=C5084&isbn=9781584885085&parent_id=&pc=`.

[TW06] Wade Trappe and Lawrence C. Washington. *Introduction to Cryptography with Coding Theory*. Prentice Hall, second edition, 2006. ISBN 0-13-186239-1. URL `http://vig.prenhall.com/catalog/academic/product/0,1144,0131862391,00.html`.

[Wik] Wikipedia. Playfair cipher. URL `http://en.wikipedia.org/wiki/Playfair_cipher`.