CPSC 467b: Cryptography and Computer Security

Michael J. Fischer

Lecture 3
January 15, 2010

Caesar cipher

Recall: Base Caesar cipher:

$$E_k(m) = (m + k) \bmod 26$$

$$D_k(c) = (c - k) \bmod 26.$$

Full Caesar cipher:

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

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

Simplified Caesar cipher

A probabilistic analysis of the Caesar cipher.

Simplify by restricting to a 3-letter alphabet.

$\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0, 1, 2\}$
$E_k(m) = (m + k) \bmod 3$
$D_k(m) = (m - k) \bmod 3.$

A priori message probabilities:

m	p_m
0	1/2
1	1/3
2	1/6

Joint message-key distribution

Each key has probability 1/3.

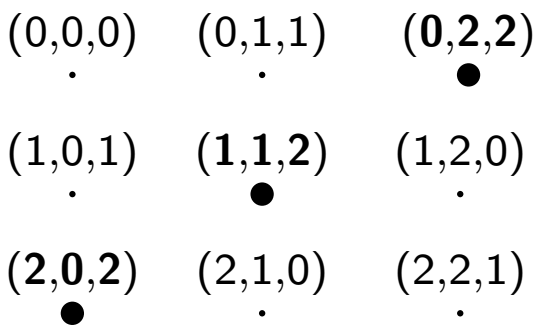
Joint probability distribution:

m \ k	0	1	2
0	1/6	1/6	1/6
1	1/9	1/9	1/9
2	1/18	1/18	1/18

Conditional probability distribution

Recall $P[m = 1] = 1/3$.
Eve sees $c = 2$.
She wishes to compute $P[m = 1 \mid c = 2]$.

First, find the sample space Ω.
Points in Ω are triples (m, k, c), where $c = E_k(m)$.:



Points for which $c = 2$ are shown in bold.

Proof of perfect secrecy

$P[c = 2]$ is the sum of the probabilities of the bold face points, i.e., $1/6 + 1/9 + 1/18 = 6/18 = 1/3$.

		k = 0	1	2
m	0	1/6	1/6	**1/6**
	1	1/9	**1/9**	1/9
	2	**1/18**	1/18	1/18

The only point for which $m = 1$ is $(1, 1, 2)$ (the center point).
It's probability is $1/9$, so $P[m = 1 \wedge c = 2] = 1/9$.
By definition of conditional probability,

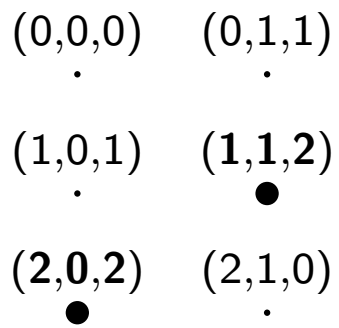
$$P[m = 1 \mid c = 2] = \frac{P[m = 1 \wedge c = 2]}{P[c = 2]} = \frac{1/9}{1/3} = \frac{1}{3} = P[m = 1].$$

Similarly, $P[m = m_0 \mid c = c_0] = P[m = m_0]$ for all m_0 and c_0.
Hence, simplified Caesar cipher is information-theoretically secure.

A minor change

Suppose we reduce the key space to $\mathcal{K} = \{0, 1\}$.
The a priori message distribution stays the same, but the joint probability distribution changes as does the sample space.

m \ k	0	1
0	1/4	1/4
1	1/6	1/6
2	1/12	1/12



Now, $P[c = 2] = 1/6 + 1/12 = 3/12 = 1/4$, and $P[m = 1 \wedge c = 2] = 1/6$. Hence,

$$P[m = 1 \mid c = 2] = \frac{1/6}{1/4} = \frac{2}{3} \neq \frac{1}{3} = P[m = 1].$$

Perfect secrecy lost

The probability that $m = 1$ given $c = 2$ is double what it was.

Once Eve sees $c = 2$ there are only two possibilities for m:

1. $m = 1$ (and $k = 1$)
2. $m = 2$ (and $k = 0$).

No longer possible that $m = 0$!

Eve narrows the possibilities for m to the set $M = \{1, 2\} \subseteq \mathcal{M}$. Her probabilistic knowledge of m changes from the initial distribution $(1/2, 1/3, 1/6)$ to the new distribution $(0, 2/3, 1/3)$. She has learned at lot about m, even without finding it exactly/

A seemingly minor change turns a cryptosystem with perfect secrecy into one that leaks a considerable amount of information!

Caveats with perfect secrecy

Perfect secrecy seems like the gold standard of security.

Nevertheless, even a scheme with perfect secrecy is not without defects and must still be used carefully.

Two problems:

- It succumbs immediately to a known plaintext attack.
- It is subject to a modification attack.

Known plaintext attack against simplified Caesar cipher

Suppose one knows even a single plaintext-ciphertext pair (m_1, c_1).

One easily solves 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, the system is completely broken.

Man-in-the-middle attacks

An *active attacker* is one who can both read and alter messages en route to their destinations.

We refer to such an attacker as "Mallory", and we call such an attack a *man-in-the-middle* attack.

In a *modification attack*, mallory can modify the contents of a message in specific semantically-meaningful ways even though he has no idea what the message actually is.

Modification attack against base Caesar cipher

Suppose Alice sends c to Bob. Mallory intercepts it and changes c to $(c + 5)$ mod 26.

Even though he doesn't know the key and cannot read m, he knows that he has changed m to $(m + 5)$ mod 26.

Why? Let's do the calculations. (All arithmetic is 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.

A modification attack on English vowels

In our encoding scheme, vowels are represented by even numbers: $A = 0$, $E = 4$, $I = 8$, $O = 14$, and $U = 20$. If m is a vowel, then $m' = (m + 5) \bmod 26$ is guaranteed not to 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 when it is to her advantage.*

The fact that she was using a cryptosystem for which perfect secrecy is known did not protect her.

Moral

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 and a ciphertext-only attack fails miserably against a known plaintext attack or against an active attacker.

Exclusive-or

The *one-time pad* is an information-theoretically secure cryptosystem that works for messages of arbitrary length.

It is important because

- it is sometimes used in practice;
- it is the basis for many stream ciphers, where the truly random key is replaced by a pseudo-random bit string.

It is based on *exclusive-or* (XOR), which we write as $\oplus$.

$x \oplus y$ is true when exactly one of x and y is true.
$x \oplus y$ is false when x and y are both true or both false.

Exclusive-or is just sum modulo two if 1 represents true and 0 represents false.

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

The one-time pad cryptosystem

$\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^r$ for some length r.

$E_k(m) = D_k(m) = k \oplus m$, where $\oplus$ is applied componentwise to corresponding bits of k and m.

XOR is associative and is its own inverse. Thus,

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

Like the base Caesar cipher, for given m and c, there is exactly one key k such that $E_k(m) = c$ (namely, $k = m \oplus c$).

Fixing c, as k varies over all possible keys, m varies over all possible messages, so every message is equally likely given c.

It follows that the one-time pad is information-theoretically secure.

One-time pad in practice

The one-time pad would seem to be the perfect cryptosystem.

- It works for messages of any length (by choosing a key of the same length).
- It is easy to encrypt and decrypt.
- It is information-theoretically secure.

In fact, it is sometimes used for highly sensitive data.

It has two major drawbacks:

1. The key k must be as long as the message to be encrypted.
2. The same key must never be used more than once. (Hence the term "one-time".)

Together, these make the problem of key distribution and key management very difficult.

One-time pad vulnerable to a known plaintext attack

If Eve knows just one plaintext-ciphertext pair (m_1, c_1), then she can recover the key $k = m_1 \oplus c_1$.
This allows 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 messages m_1 and m_2.

In particular, she can compute $m_1 \oplus m_2$ without knowing either m_1 or 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.

Block ciphers

A *block cipher* is an encryption system where the base message space $\mathcal{M}_0$ is finite. Elements of $\mathcal{M}_0$ are called *blocks*.
Blocks are typically bit strings of some convenient length such as 64 or 128.

A block cipher can be used in *electronic codebook (ECB) mode* to encrypt arbitrarily long messages:

1. Represent message m as a sequence of blocks $b_1, b_2, \ldots, b_r$.
2. Encrypt each block using the base cipher, so $c_i = E_k(b_i)$, $i \in [1 \ldots r]$. The same key k is used for each.
3. Output the sequence $c_1, c_2, \ldots, c_r$ of encrypted blocks.

Analysis of the Caesar cipher

The Caesar cipher is an example of a *block cipher*, where the blocks are single letters.

Although the Caesar cipher is not practical because of its small key space, many of its properties are representative of any block cipher used in ECB mode.

We now explore its security properties.

Breaking the Caesar cipher: An example

Suppose you intercept the ciphertext JXQ.
You quickly discover that $E_3(\text{GUN}) = \text{JXQ}$.
But is $k = 3$ and GUN is the correct decryption?

You then discover that $E_{23}(\text{MAT}) = \text{JXQ}$.
Now you are in a quandary. Which decryption is correct?

Have you broken the system or haven't you?

You haven't found the plaintext for sure, but you've reduced the possibilities down to a small set.

Breaking the Caesar cipher: Extending these ideas

The longer the correct message, the more likely that only one key results in a sensible decryption.

For example, suppose the ciphertext were "EXB JXQ".
We saw two possible keys for "JXQ" — 3 and 23.
Trying them both we get:

$$k = 3: \quad D_3(\text{EXB JXQ}) = \text{BUY GUN}.$$

$$k = 23: \quad D_{23}(\text{EXB JXQ}) = \text{HAE MAT}.$$

Latter is nonsense, so we know $k = 3$ and the message is "BUY GUN".

Breaking the Caesar cipher: Conclusion

Information-theoretic security of the Caesar cipher.

- $r = 1$: It is perfectly secure.
- $r > 1$: It is only partially secure or completely breakable, depending on message length and redundancy present in the message.

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.

Trying all keys

A *brute force attack* tries all possible keys k.

For each k, Eve computes $m_k = D_k(c)$ and tests if m_k is meaningful. If exactly one meaningful m_k is found, she knows that $m = m_k$.

Given long enough messages, the Caesar cipher is easily broken by brute force—one simply tries all 26 possible keys to see which leads to a sensible plaintext.

The Caesar cipher succumbs because the key space is so small.

Automating brute force attacks

With modern computers, it is quite feasible for an attacker to try millions ($\sim 2^{20}$) or billions ($\sim 2^{30}$) of keys.

The attacker also needs an automated test to determine when she has a likely candidate for the real key.

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.

Random English-like messages

Consider random messages whose letter frequencies are similar to that of valid English sentences.

For each letter b, let p_b be the probability (relative frequency) of that letter in normal English text.

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.

Determining likely keys

Assume Eve knows that $c = E_k(m)$, where m was chosen randomly as described above and k is uniformly distributed.

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 (including English text) and sufficiently long messages, it works correctly with very high probability.

How long should the keys be?

The DES (Data Encryption Standard) cryptosystem (which we will talk about shortly) has 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. (Click here 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.

Now, 12 years later, the same task could likely be done on a commercial cluster computer such as Amazon's Elastic Compute Cloud (EC2) at modest cost.

What is safe today and into the future?

DES with its 56-bit keys offers little security today.

80-bit keys were considered acceptable in the past decade, but in 2005, NIST proposed that they be used only until 2010.

Triple DES (with 112-bit keys) and AES (with 128-bit keys) will probably always be safe from brute-force attacks (but not necessarily from other kinds of attacks).

Quantum computers, if they become a reality, would cut the effective key length in half (see Wikipedia "key size"), so some people recommend 256-bit keys (which AES supports).

Cryptography before computers

Large-scale brute force attacks were not feasible before computers.

While Caesar is easily broken by hand, clever systems have been devised that can be used by hand but are surprisingly secure.

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 $|\mathcal{K}| = 26! > 2^{74}$, large enough to be moderately resistant to a brute force attack.

Nevertheless, monoalphabetic ciphers can be readily broken using letter frequency analysis, given a long enough message.

This is because *monoalphabetic ciphers preserve letter frequencies*.

How to break monoalphabetic ciphers

Each occurrence of a in m is replaced by $k(a)$ to get c.
Hence, if a is the most frequent letter in m, $k(a)$ will be the most frequent letter in c.

Eve now guesses that a is one of the most frequently-occurring letters in English, i.e., 'e' or 't'.
She then repeats on successively less frequent ciphertext letters.

Of course, not all of these guesses will be correct, but in this way the search space is vastly reduced.
Moreover, many wrong guesses can be quickly discarded even without constructing the entire trial key because they lead to unlikely letter combinations.

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. Pairs of plaintext letters are then located in the matrix and used to produce a corresponding pair of ciphertext letters.

How Playfair works

Construct the matrix from the passphrase.

- Construct the matrix by writing the passphrase into the matrix cells from left to right and top to bottom.
- Omit any letters that have previously been used.
- Fill remaining cells 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.

Example Playfair matrix

Let the passphrase be

"CRYPTOGRAPHY REQUIRES STRONG KEYS".

The resulting matrix is

C	R	Y	P	T
O	G	A	H	E
Q	U	I/J	S	N
K	B	D	F	L
M	V	W	X	Z

First occurrence of each letter in the passphrase shown in orange:

"CRYPTOGRAPHY REQUIRES STRONG KEYS".

Letters not occurring in the passphrase: BDFLMVWXZ.

Encrypting in Playfair: preparing the message

To encrypt a message using Playfair:

- Construct the matrix.
- Remove spaces and pad the message with a trailing 'X', if necessary, to make the length even.
- Break up the message into pairs of letters.
- In case a pair of identical letters is about to be produced, insert an "X" to prevent that.

Examples:

- "MEET ME AT THE SUBWAY" becomes "ME" "ET" "ME" "AT" "TH" "ES" "UB" "WA" "YX".
- "A GOOD BOOK" becomes "AG", "OX", "OD" "BO", "OK".

Encrypting in Playfair: substituting the pairs

To encrypt pair *ab*, look at rectangle with *a* and *b* at its corners.

1. If *a* and *b* appear in different rows and different columns, replace each by the letter at the opposite end of the corresponding row. Example: replace "AT" by "EY":

   Y P **T**
   A H E

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".
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*.

Example: "MEET ME AT THE SUBWAY" encrypts as "ZONEZOEYPEHNBVYIPW".

Decrypting in Playfair

Decryption is by a similar procedure.

In decrypting, one must manually remove the spurious 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


Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone.
Handbook of Applied Cryptography.
CRC Press, 1996.


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.


Wade Trappe and Lawrence C. Washington.
Introduction to Cryptography with Coding Theory.
Prentice Hall, second edition, 2006.
ISBN 0-13-186239-1.


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