Ciphers

Luke Anderson

luke@lukeanderson.com.au

8th March 2019

University Of Sydney

Overview

Crypto-Bulletin

Crypto-Bulletin

Google says Chrome on Windows combo zero-day exploited in the wild

https://www.itnews.com.au/news/google-says-chrome-on-windows-combo-zero-day-exploited-in-the-wild-520275

Huawei sues US government over federal business ban

https://www.itnews.com.au/news/huawei-sues-us-government-over-federal-business-ban-520241

Hackers Sell Access to Bait-and-Switch Empire

https://krebsonsecurity.com/2019/03/hackers-sell-access-to-bait-and-switch-empire/

Ciphers

Cryptography

Cryptography is the study of mathematical techniques related to the design of ciphers.

It's one of the core examples of the many mechanisms making up security:

- Cryptography
- Signature / Pattern Matching
- Access Control
- Statistical Profiling
- Traffic Security
- Countermeasures
- Tamper Resistance

Cryptography

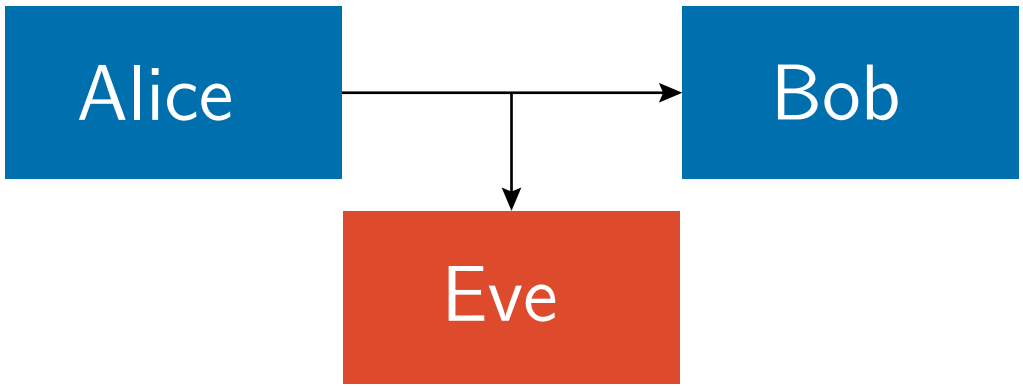
The fundamental application of cryptography is enabling **secure communications** over an **insecure channel**.

How can Alice send a secure message to Bob over an insecure channel when Eve is listening in?

Eve is an active attacker and may tap, insert or modify messages in transit.

How does one use cryptography to provide security such as:

- Authentication
- Confidentiality
- Integrity
- Non-Repudiation

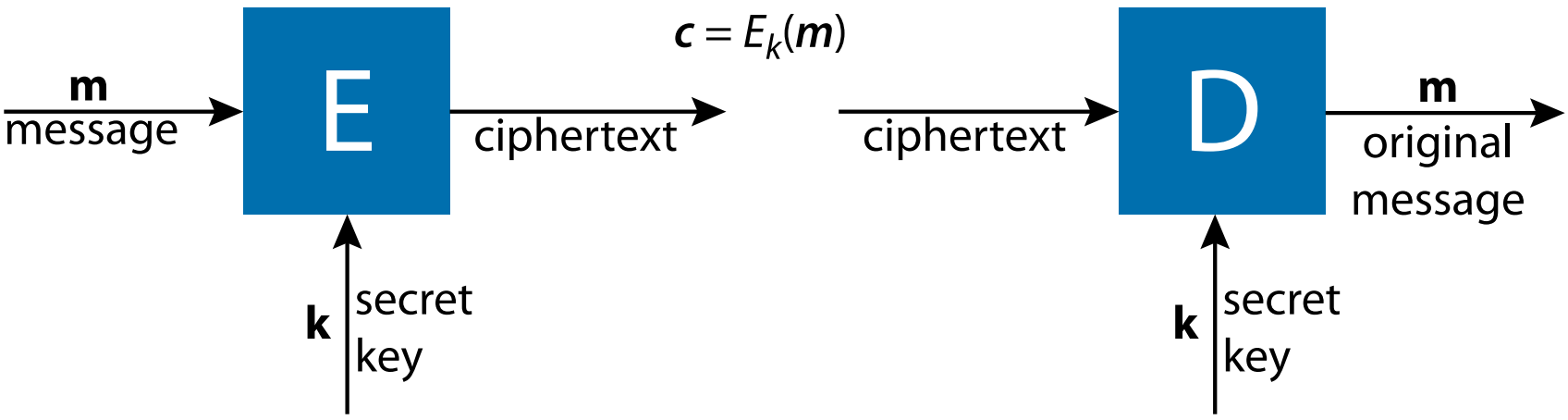


Symmetric Ciphers

The traditional way of achieving secrecy is through a **shared secret key**. This is also known as **symmetric encryption** since the key used to both encrypt and decrypt messages is the same.

A symmetric cipher is an encryption algorithm $\mathrm{E_k}$ and a decryption algorithm $\mathrm{D_k}$ which inverts $\mathrm{E_k}$. In other words, for all keys k and messages m,

$$\mathrm{D_k}(\mathrm{E_k}(\mathrm{m})) = \mathrm{m}$$



Communications with Symmetric Ciphers

Alice and Bob share:

- A secret key: k
- An encryption algorithm: E_k
- A decryption algorithm: D_k

Alice wants to send Bob the message m.

The unencrypted message is known as either the **plaintext** or **cleartext**.

Alice encrypts m by computing the **ciphertext**:

$$c = E_k(m)$$

Bob decrypts c by computing the original plaintext message m:

$$D_k(c) = m$$

Symmetric Cryptosystem

It is computationally hard to decrypt the ciphertext c without the secret key k.

The secret key k is usually a large number of bits ($\geqslant 64$)

The range of possible values of k is called the **key space** K
For 64 bit keys, $\mathrm{K} = \{0,1\}^{64}$, i.e the numbers $\{0, 1, \ldots, 2^{64} - 1\}$

The range of possible messages is called the **message space**: M

A **cryptosystem** is a system consisting of an algorithm, plus all possible plaintexts, ciphertexts, and keys.

Types of Symmetric Ciphers

Stream Ciphers
Operate on a single bit or byte at a time.

Block Ciphers
Operate on blocks (numbers of bits) of plaintext at a time.

Cryptanalysis

Definition

Cryptanalysis is used to breach cryptographic security systems and gain access to the contents of encrypted messages, even if the cryptographic key is unknown.

We *always* assume that attackers have:

- Complete access to the communications channel
- Complete knowledge about the cryptosystem

Secrecy must *only* depend on the key.

Ciphertext only attack (COA)

Attacker only has access to the ciphertext.

Given:

$$c_1 = E_k(m_1), c_2 = E_k(m_2), \cdots, c_n = E_k(m_n)$$

Find any of:

- $m_1, m_2, \cdots, m_n$
- k
- An algorithm to infer m_{n+1} from c_{n+1}

Known Plaintext Attack (KPA)

Attacker intercepts a random plaintext / ciphertext pair: (m, c).

Given:

$$[\mathrm{m}_1, \mathrm{c}_1 = \mathrm{E}_\mathrm{k}(\mathrm{m}_1)], \cdots, [\mathrm{m}_\mathrm{n}, \mathrm{c}_\mathrm{n} = \mathrm{E}_\mathrm{k}(\mathrm{m}_\mathrm{n})]$$

Find any of:

- k
- An algorithm to infer $\mathrm{m}_{\mathrm{n}+1}$ from $\mathrm{c}_{\mathrm{n}+1}$

Chosen Plaintext Attack (CPA)

Attacker selects a message m and receives the ciphertext c.
Stronger than KPA – Some ciphers resistant to KPA are not resistant to CPA.

Given:

$$[\mathrm{m}_1, \mathrm{c}_1 = \mathrm{E}_\mathrm{k}(\mathrm{m}_1)], \cdots, [\mathrm{m}_\mathrm{n}, \mathrm{c}_\mathrm{n} = \mathrm{E}_\mathrm{k}(\mathrm{m}_\mathrm{n})]$$

with chosen m.

Find any of:

- k
- An algorithm to infer $\mathrm{m}_{\mathrm{n}+1}$ from $\mathrm{c}_{\mathrm{n}+1}$

Chosen Ciphertext Attack (CCA)

Attacker specifies a ciphertext c and receives the plaintext m.

Given:

$$[\mathrm{c}_1, \mathrm{m}_1 = \mathrm{D}_\mathrm{k}(\mathrm{c}_1)], \cdots, [\mathrm{c}_\mathrm{n}, \mathrm{m}_\mathrm{n} = \mathrm{D}_\mathrm{k}(\mathrm{c}_\mathrm{n})]$$

with chosen c.

Find any of:

- k

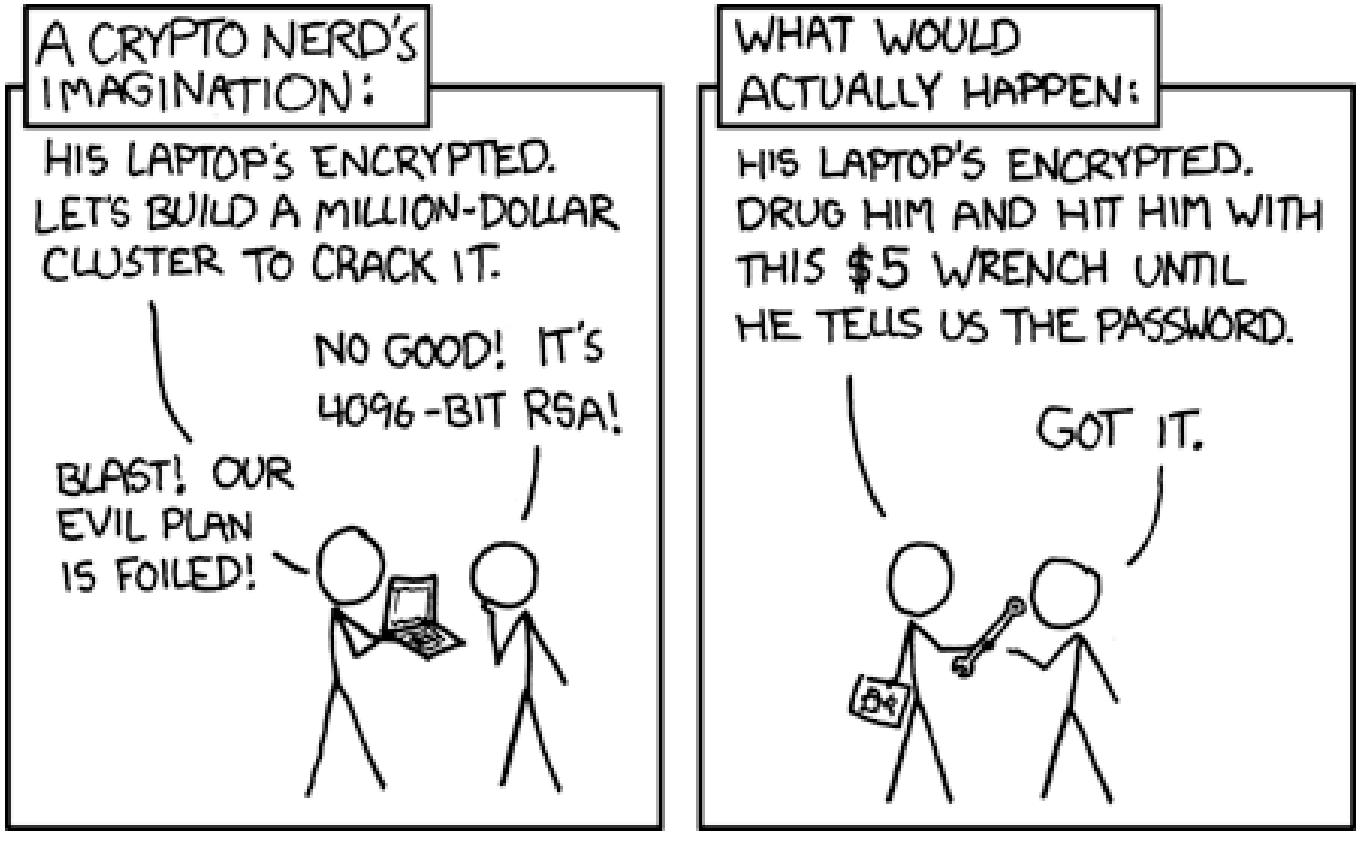
Rubber hose attack (RHA)

Also known as a **purchase-key attack** or **rubber-hose cryptanalysis**.

The cryptanalyst blackmails, threatens or tortures someone to retrieve k.

Extremely powerful, usually the easiest way to break a cryptosystem.

Rubber hose attack (RHA)



XKCD: Security

Attack Examples

Known Plaintext Attack

An attacker knowing that source code is being encrypted
(the first bytes are likely `#include`, copyright notices, etc)

Famous break of the Japanese PURPLE cipher in WW2

- A complex cipher used to protect high level communications
- The allies had already broken several of the Japanese diplomatic ciphers
- PURPLE was used to protect communications but the Japanese could only afford to build and deploy 12 cipher machines
- They sent these to the twelve most important embassies
- Some messages needed to be broadcast to all embassies
- So some messages had to be sent using old ciphers the US had already broken!

Attack Examples

Chosen Plaintext Attack

Feed intelligence to an ambassador with the goal that it ends up encrypted and sent back home.

Ciphertext Only Attack

Stealing cookies through weaknesses in RC4

- The start of RC4 cipher streams have biases that can be calculated when the same message is encrypted multiple times.
- RC4 was previously used in the majority of all SSL traffic; start of the stream is almost always cookies (HTTP).
- Simply trick client into making many requests, inspect the traffic, then steal the cookies.

Attack Metrics

An algorithm is **unconditionally secure** if no matter how much ciphertext an attacker has, there is not enough information to deduce the plaintext.

Information security is a resource game with attacks measured in terms of:

Data Requirements
: How much data is necessary to succeed?

Processing requirements (work factor)
: How much time is needed to perform the attack?

Memory requirements
: How much storage space is required?

Computational cost
: How many instances running on EC2?

Substitution Ciphers

Substitution ciphers are the oldest form of cipher.

The secret key consists of a table which maps letter substitutions between plaintext and ciphertext.

The most famous is the *Caesar cipher* where each letter is shifted by 3 (modulo 26):

```
abcdefghijklmnopqrstuvwxyz
DEFGHIJKLMNOPQRSTUVWXYZABC
```

Similar to ROT13 which shifts plaintext 13 places – largest advantage is that encrypting twice results in the plaintext:

$$\mathrm{ROT13}(\mathrm{ROT13}(\mathrm{m})) = \mathrm{m}$$

Substitution Ciphers

There are 26! (factorial) different possible keys ($\approx 2^{88}$ or 88-bits).

Monoalphabetic (single character) substitution cipher:

$$\begin{aligned} \text{Src} &= \texttt{abcdefghijklmnopqrstuvwxyz} \\ \text{Key} &= \texttt{XNYAHPOGZQWBTSFLRCVMUEKJDI} \\ \\ \text{m} &= \texttt{thiscourserockstheblock} \\ \text{c} &= \texttt{MGZVYFUCVHCFYWVMGHNBFYW} \end{aligned}$$

Substitution ciphers are easy to break using **frequency analysis** of the letters:

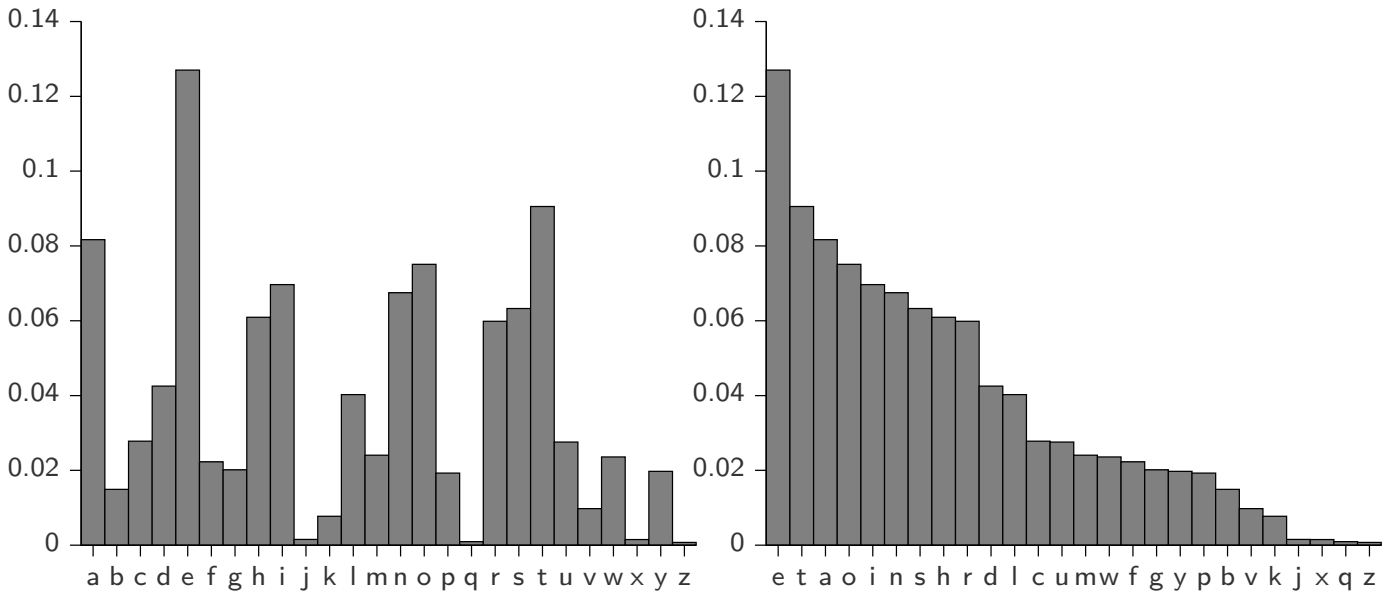
- Single letters
- Digraphs (pairs of letters)
- Trigraphs (three letters)

This is a **ciphertext only attack**.

Frequency Analysis on Substitution Ciphers

Substitution ciphers are easy to break by using **frequency analysis** of the letters:

- Order the histogram and the spikes should follow a similar pattern.
- Try mapping replacements, using digraph and trigraphs to check & assist you.



The frequency of English letters.

Improved Substitution Ciphers: Homophonic Ciphers

Homophonic ciphers are substitution ciphers that replace a common letter with multiple symbols (i.e. E can go to [C, ε, O])

Peaks or troughs in the letter frequency are hidden as they're broken down into multiple smaller spikes.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
================================================== 
D X S F Z E H C V I T P G A Q L K J R U O W M Y B N
9       7       3         5 0       4 6
        2
        1
```

- "D" would become "F"
- "E" would be randomly chosen as one of: [Z, 7, 2, 1].

As a result, the high frequency of the letter "E" (the most common letter in English) is spread amongst several characters, making frequency analysis much more difficult.

Improved Substitution Ciphers: Homophonic Ciphers

Still difficult to decipher even using modern computing:
success rate is measured in terms of alphabet size and ciphertext length.

Deciphered using nested hill climbing (heuristic algorithm / educated guessing):

- Outer layer determines the number of symbols each letter maps to
- Inner layer determines the exact mapping

Used by the Zodiac killer:

- "Zodiac 408" was solved within days of publication
- "Zodiac 340" still remains unsolved

http://www.cs.sjsu.edu/faculty/stamp/RUA/homophonic.pdf

Permutation Ciphers

Permutation ciphers are also known as **transposition ciphers**. The secret key, π, is a random permutation.

Given a message $m = [m_1, m_2 m_3, ..., m_n]$
we can compute the encryption via:
$E_\pi(m) = [m_{\pi(1)} m_{\pi(2)} m_{\pi(3)}, ..., m_{\pi(n)}]$

Suppose π is:

1	2	3	4	5	6
4	3	1	5	2	6

Then E_π("crypto") ="PYCTRO".

Vigenere Cipher

Originated in Rome in the sixteenth century, a Vigenere cipher is a polyalphabetic substitution cipher (made of multiple monoalphabetic substitution ciphers). The secret key is a repeated word with encryption performed by adding the key modulo 26.

Vigenere Encryption

```
Plaintext:   launchmissilesatlosangeles
Keystream:   cryptocryptocryptocryptocr
=====================================
Ciphertext:  nrscvvozqhbzgjyiecurlvwzgj
```

$$\mathrm{L} + \mathrm{C} = 11 + 2 \quad = 13 \bmod 26 = 13^{\mathrm{th}}\text{char} \quad \Rightarrow \mathrm{N}$$

$$\mathrm{N} + \mathrm{Y} = 13 + 24 \quad = 37 \bmod 26 = 11^{\mathrm{th}}\text{char} \quad \Rightarrow \mathrm{L}$$

Note: The zero index – 0^{th} character – is A.

Also punctuation and white space are removed to increase cryptanalysis difficulty.

Breaking Vigenere Ciphers

The index of coincidence is a statistical measure of text that is used for distinguishing simple substitution ciphers from Vigenere ciphers.

Intuitively it's the measure of the probability of a collision if a string is compared to a randomly shifted version of itself.
(i.e. How many characters occur in the same spot if you shuffle the string?)

The index of coincidence κ is calculated by the following formula:

$$\kappa = \frac{1}{N(N-1)} \sum_{i=A}^{i=Z} F_i(F_i - 1)$$

Where F_i is the count of letter i (where $i = A, B, \ldots, Z$) in the ciphertext, and N is the length of the ciphertext.

Breaking Vigenere Ciphers

The index of coincidence can be used to detect the length of the key in a Vigenere cipher, by use of the following observation:

Important property

A substitution cipher does not change the index of coincidence. Since a substitution cipher only rearranges the terms in the sum, it does not change the index of coincidence.

Using standard frequencies with which individual letters appear in English, the probability that a coincidence will occur is approximately $\kappa_p = 0.0669$.

If the text is random and letters chosen with equal probability then the probability of a coincidence is much smaller: $\kappa_p = 0.0385$ $(= 1/26)$.

Index of Coincidence by Language

Malay	0.085286
Dutch	0.079805
Japanese	0.077236
German	0.076667
Spanish	0.076613
Arabic	0.075889
French	0.074604
Portuguese	0.074528
Finnish	0.073796
Italian	0.073294
Danish	0.070731
Norwegian	0.069428
Greek	0.069165
English	**0.066895**
Swedish	0.064489
Serbo-Croatian	0.064363
Russian	0.056074
Random	0.038461

Example: Breaking Vigenere

```
TPCTY LVEOO GBVRC BTWXS IHDKD QIRVQ QUKWL TMNQO EKMLP AURKL VHIUX YJRNV QWJEK UEQVD IXPLU RKLVT QSLKI LWAZI JWXPL
QRKIO PWFME XLLCP KDIKV EUXYX EAAQV MEKVN AVZRQ JGEMX LQUPM PCRLO IZPZZ FPONI AYPVQ RMVHC QZFLV IKGUK LRXER MDWVG
RVMPQ PWLWT TIYEQ JAYMK XBPUK PZJBF WIRKS QJMSV FYKFP QLRXE OIPID IQQLI ICPFP LMVBR BUAMW KLLUM FLRXE CDQFV IKNWZ
KUIXF BTIII CQZQM JQVUX UVZXL XMDAY IIOMP AZXEK VYIDC EGIDX NMQJQ ZQVMP FMESC EQGQD IDIJD MDXYI ACGES WSIFQ YIUMQ
CBQSE EINBT CNSOM AUQLW BQVFL VALTS AJKLV JIZHJ MPVZQ XTLCQ ZFLDC ECVPW LRQQB TIVQV UWGPK LFTAF IKLXH BQVKL BGIEE
KLFTA FCCEK FAQPR LEGID QVWMG MPMCC LNWDH DCPRQ DMKJX KTQXY LFFMZ SKXEA NMGVJ OQUYI CIPVQ NICMH GCZXF XEGUF LRXDQ
LAAEM KVWFL VTFVK MYJIJ GBALV EOVPK PFZFP OWMEH KGAEM EXEGU AVEMK INAVZ RQJMQ HFMQT CEXTE RUMYI KSHPW IXYIT CGILV
VBKVU WYSRN LIECO CQZUP ZJQWX YCJSR NCZXF XEGMP ICMSG ZYIFP LTLRV FQJKV QIEIJ KMEMW PBGCZ XFXEG MFSYM AGUQX VEZJU
QXFHL VPKAZ PIHWD XYSRC ZFQPK LFBTC JTFTQ FMJKL QLXIR HJGQZ XFXEG TMRUS CWXDM XLQPM EWHYF ESQRD ILNWD HWSOV PKRRQ
BUAMO VJLTB TCIMD JBQSL WKGAE WROBD ZURXQ VUWGP FYQQN FVFYY NMMRU SCVPK QVVZA KGXFJ COQZI VRBOQ QWRRA FMEXI SVCTX
XYIJV PMXRJ CNQOX DCPQC XJFVF CUFLP WBTDM RK
```

Shift Index

1: .028	5: .042
2: .045	6: .035
3: .034	**7: .070**
4: .037	8: 0.32

Key length is most likey 7, since it's closest to 6.6%.

Breaking Vigenere Ciphers

Once the key length N is known, we attack the N subtexts of the message.

Taking every Nth symbol in ciphertext C gives N monoalphabetic substitution ciphers:

$$(C_0, C_N, C_{2N}, \ldots),$$
$$(C_1, C_{N+1}, C_{2N+1}, \ldots),$$
$$\ldots$$
$$(C_{N-1}, C_{N+(N-1)}, C_{2N+(N-1)}, \ldots)$$

Note the index of coincidence varies by language and can be domain specific (e.g. may be noticeably different for a physics journal paper).

XOR and OTP

What is XOR?

XOR is the "exclusive or" operation: one or the other, but not both.

It is addition modulo 2 and is represented by $\oplus$.

$$a \oplus b = (a + b) \bmod 2$$

a	b	a&b	a||b	$a \oplus b$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Table: Truth Table with XOR

XORing Bits

Typically we XOR bits together:

XOR Encryption

```
Plaintext:   011011000110111101101100
Keystream:   011000110110000101110100
=====================================
Ciphertext:  000011110000111000011000
```

Often the plaintext will be XOR'd with a key stream to produce ciphertext.

This is effectively the same as a Vigenere cipher.
Where a XOR is addition modulo 2, a Vigenere cipher is addition modulo 26 since XOR works with bits and not letters.

Interesting XOR Properties

Something **XOR'd with with itself is zero**.

$$A \oplus A \equiv 0$$

XOR is also **associative**:

$$A \oplus (B \oplus C) \equiv (A \oplus B) \oplus C$$

and **commutative**:

$$A \oplus B \equiv B \oplus A$$

XOR

XOR (addition modulo 2) is commonly used to provide security in programs. It is very weak by itself, but forms the building block of most crypto primitives.

The message m is XOR'd bitwise with a secret key:

$$\mathrm{c} = \mathrm{m} \oplus \mathrm{k}$$
$$\mathrm{m} = \mathrm{c} \oplus \mathrm{k}$$

XOR is effectively a Vigenere cypher and easy to break:

- Determine the key length N from index of coincidence
- Shift cyphertext by N and XOR with itself
- This removes the key ($\mathrm{c} \oplus \mathrm{c}' = \mathrm{m} \oplus \mathrm{k} \oplus \mathrm{m}' \oplus \mathrm{k} = \mathrm{m} \oplus \mathrm{m}'$)
- Results in message XOR'd with a shifted version of itself
- Language is extremely redundant (English 1.3 bits / byte)
- Easy to then decrypt

Wizardry

XOR also useful for an old-school assembly / C programming trick

"How do you swap two vairables, x & y, without using a third?"

$$x = x \oplus y$$
$$y = x \oplus y$$
$$x = x \oplus y$$

One Time Pad (OTP)

A **one time pad** is using a different substitution cipher for each letter of the plaintext.

Provided that:

- The secret key, k, is truly random
- The plaintext does not repeat
- The keystream does not repeat

a one time pad is **perfectly secure**.

Failure to meet any one of these requirements results in **zero security**.

A-Z MOD 26

To encrypt for bits, it is simply the XOR operation, or modulo 2.

When dealing with A-Z, it is equivalent to addition modulo 26.

One Time Pad (OTP)

The strength comes from the fact that a truly random key added to plaintext, produces a truly random ciphertext.

No amount of computing power can break a one time pad.

brute force would yield each and every possible message that length.

Core Problems: key distribution, key destruction, synchronisation.

- $\mathtt{k}$ must be same length as $\mathtt{m}$:
  to encrypt 1GB you need a 1GB shared key.
- Used for ultra-secure, low bandwidth communications
  e.g. military satellites, Moscow-Washington phone line
- Future: Quantum Key Distribution
  secure distribution at a distance.

Perfect Secrecy

Goal of cryptography:

ciphertext reveals nothing about the plaintext.

A cipher has perfect secrecy if, for all $m \in M, c \in C$, the plaintext and ciphertext are statistically independent:

$$\Pr[m_1 = m_2 | c_1 = c_2] = \Pr[m_1 = m_2]$$

Assuming each transmitted message is equally likely, the probability that the transmitted message is m is:

$$\Pr[m_1 = m_2] = |M|^{-1}$$

Now the probability that the transmitted message is m given that the observed ciphertext is c is:

$$\Pr[m_1 = m] = \frac{|\{k : E_k(m) = c, k \in K\}|}{|K|}$$

Perfect Secrecy

The key space K must be at least as large as the set of plaintexts:

$$|K| \geqslant |M|$$

For $M = C = \{0,1\}^n$:

any cipher with perfect secrecy satisfies $|K| \geqslant 2^n$

The one time pad has perfect secrecy as: $M = C = \{0,1\}^n$
Thus:

$$Pr[m_1 = m_2] = \frac{1}{2^n}$$

$$Pr[m_1 = m_2 | c_1 = c_2] = \frac{1}{2^n}$$

Note: we require $k \in K$ to be as long as the message, which means we need to securely communicating a key as long as the message in advance

Breaking OTP: Two Time Pad

A **two-time pad** is **perfectly insecure**. Suppose two messages m_1, m_2 are encoded using the same key k:

$$c_1 = m_1 \oplus k$$
$$c_2 = m_2 \oplus k$$

Then the key k may be cancelled by XORing the ciphertexts:

$$\begin{aligned} c_1 \oplus c_2 &= (m_1 \oplus k) \oplus (m_2 \oplus k) \\ &= m_1 \oplus m_2 \oplus k \oplus k \\ &= m_1 \oplus m_2 \end{aligned}$$

$m_1 \oplus m_2$ is easy to separate due to the redundancy in English and in ASCII (for example, bit 6 is set in letters but not most punctuation).

Breaking OTP: Malleability

The OTP and all stream ciphers are **highly malleable**. Suppose plaintext is a one bit vote $v \in 0, 1$

- $v = 0$ is a vote for Labor
- $v = 1$ is a vote for Liberal

Alice encrypts her vote using OTP and sends to Bob:

$$c = v \oplus k \text{ where } k \in 0, 1 \text{ is randomly chosen}$$

Mallory intercepts the ciphertext and sends with bits flipped:

$$c' = c \oplus 1 = \neg c$$

Bob receives c' and decrypts vote:

$$\begin{aligned} c' \oplus k &= c \oplus 1 \oplus k \\ &= v \oplus k \oplus 1 \oplus k \\ &= v \oplus 1 \end{aligned}$$

Breaking OTP: Malleability

Malleability Attack Example

A competitor is selling shares of their company by using a "secure" share trading program that encrypts a four byte integer using a four byte key k:

$$c = [b_1, b_2, b_3, b_4] \oplus [k_1, k_2, k_3, k_4]$$

If I wanted to steal a controlling share, I could make him sell a massive number of shares by flipping a high order bit that I was certain he wouldn't use

References

Handbook of Applied Cryptography:

- 1.4 – 1.5
- 7.1 – 7.3

Stallings (3rd Ed):

- 2