COMP3052.SEC Computer Security

Session 15-1 Crypto I

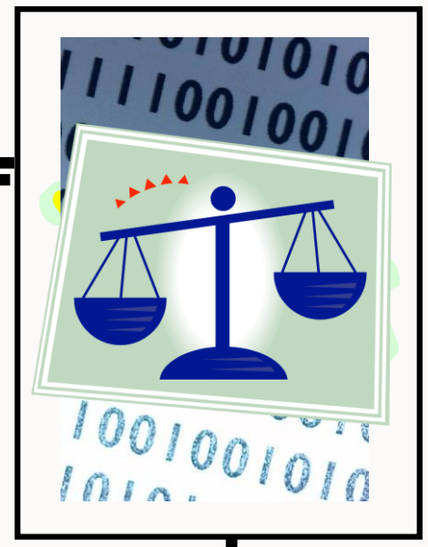
ACKNOWLEDGEMENTS

- Some of the materials we use this semester may come directly from previous teachers of this module, and other sources ...

- Thank you to (amongst others):
  - Michel Valstar, Milena Radenkovic, Mike Pound, Dave Towey …

TOPICS COVERED

- Concepts of Cryptology
- The Mathematics of Cryptology
- Algorithms and Mechanisms
  - Integrity Checking
  - Digital Signatures
  - Encryption
- Assessment of Algorithms and Mechanisms



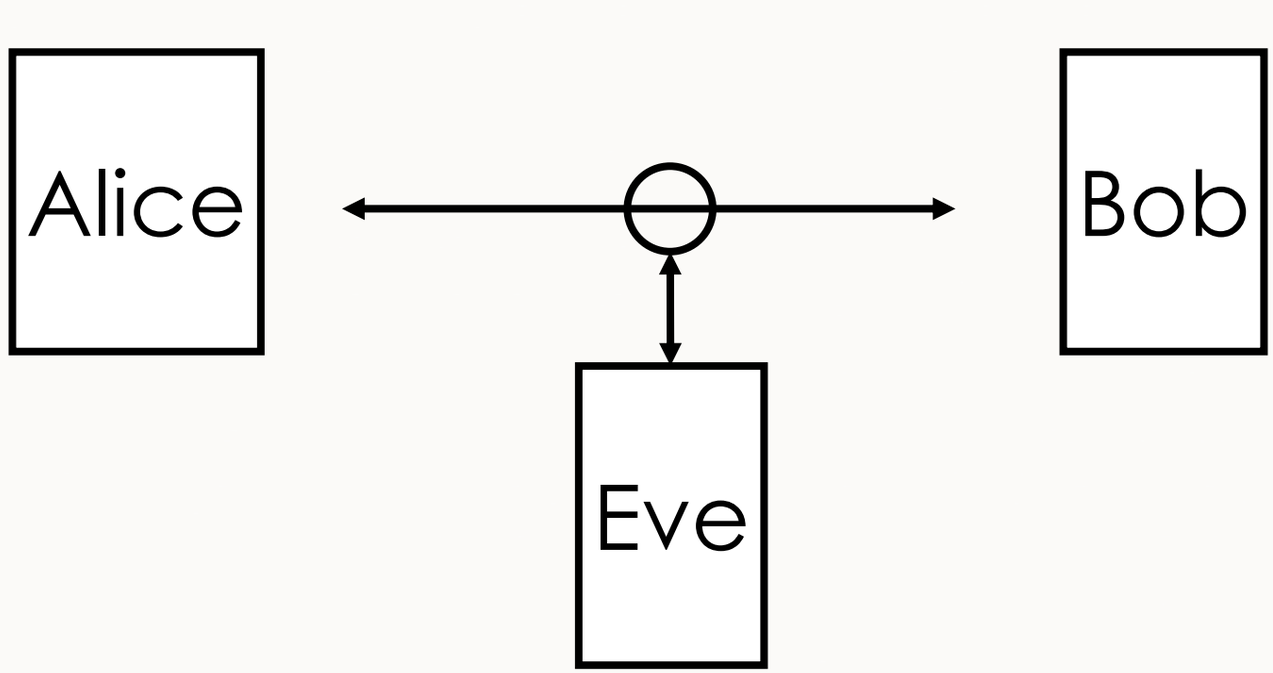
DEFINITIONS

- ***Cryptography***

- ***Cryptanalysis***

- ***Cryptology***

THE COMMUNICATION MODEL



ALL ABOUT EVE

Objectives

-Read messages

-Edit Messages

-Write messages

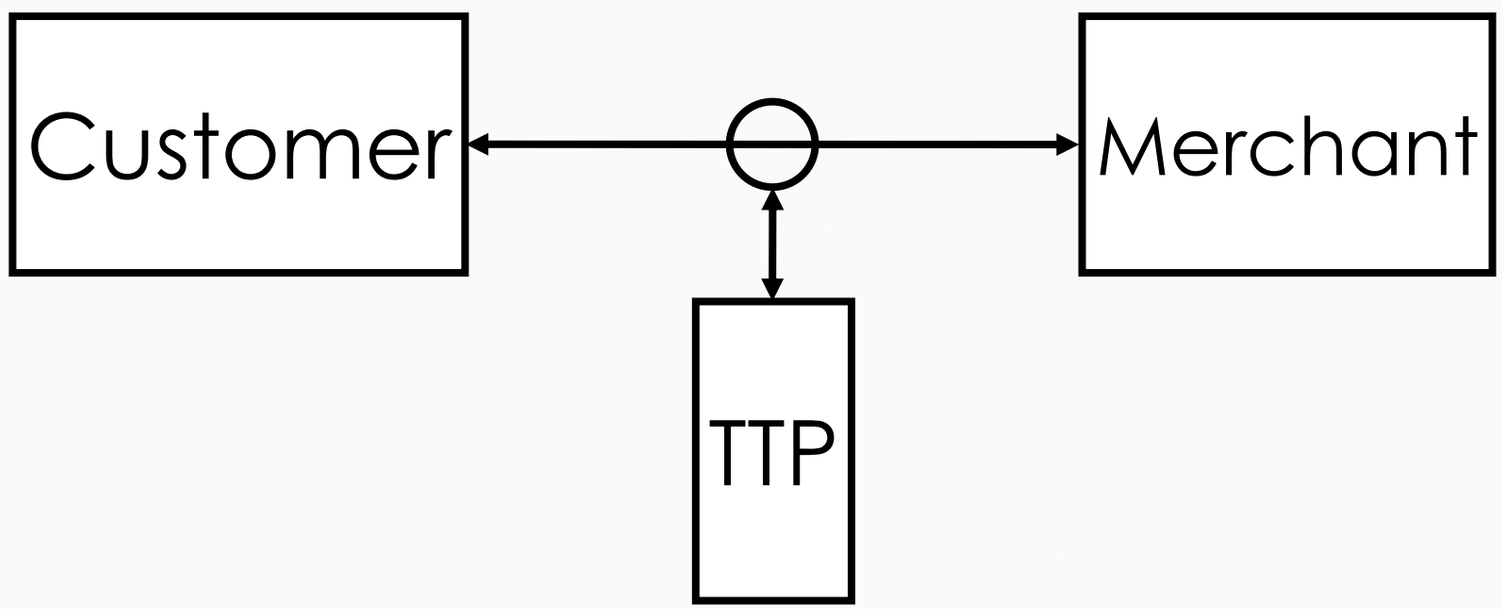
Countermeasures

-Encryption

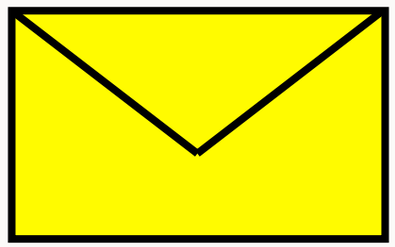
-Integrity Checking

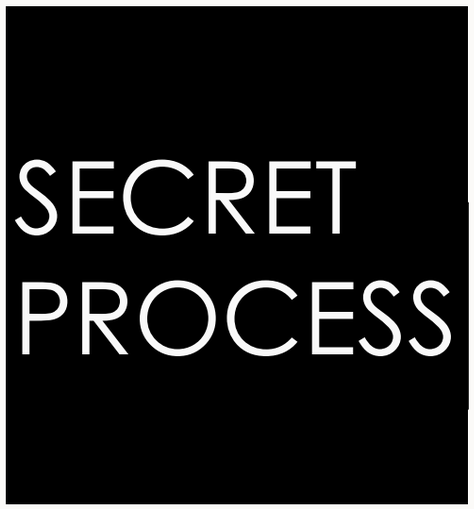
-Origin Checking

THE COMPUTER SECURITY MODEL



CRYPTOGRAPHY PARADIGMS

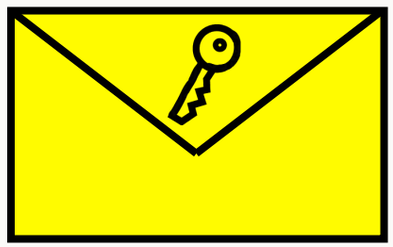




ACTIVITY …

- Please answer:
  - What happens if someone reversely engineers/sells your process?
  - Who verifies that your technique is strong enough?

THE NEW PARADIGM



Known process

KEYS AND STRENGTH

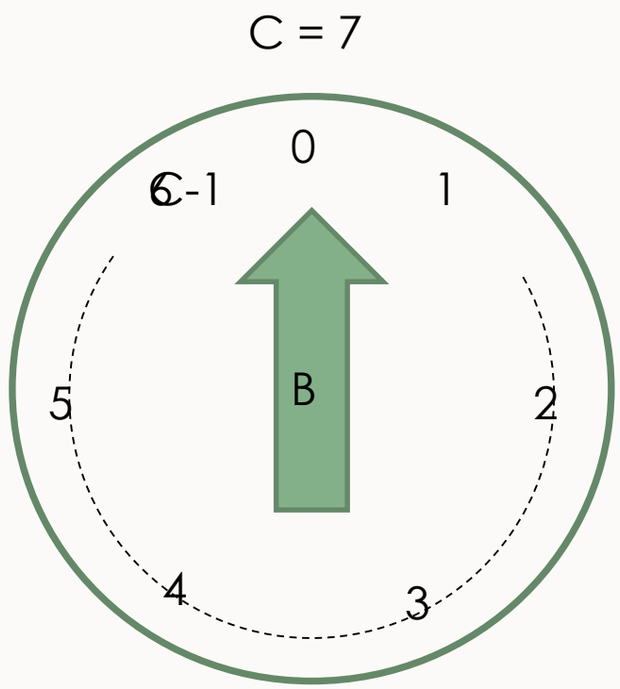
- "Security is as strong as its weakest link"
  – Bruce Schneier
- Keys vary in strength
- Algorithms vary in strength
- "Brute Force" attacks

KEY MANAGEMENT

- Where do we generate keys?
- How do we generate keys?
- Where are keys stored?
- How do we transport keys?
- Where are keys used?
- How are keys revoked and replaced?

MODULUS MATHEMATICS

A = 80



MODULUS MATHEMATICS

$$A \equiv B \bmod C$$

PROPERTIES OF THE MODULUS

Let

$a_1 \equiv b_1 \bmod C$

$a_2 \equiv b_2 \bmod C$

Then

$a_1 + a_2 \equiv (b_1 + b_2) \bmod C$

$a_1 a_2 \equiv (b_1 b_2) \bmod C$

PROPERTIES OF THE MODULUS

- Constrained to integers so division is complex
- Let's define division as the opposite of multiplication

- If f = (1/e) mod C
  - then (e*f) mod C = 1

- If multiple prime solutions exist for 1/e, the division is ***undefined*** like X/0 in regular mathematics

ACTIVITY ...

- Calculate the value of f:

```
f = (1/3) Mod 7
```

ACTIVITY …

- Calculate the value of f:

$$f = (1/3) \bmod 7$$

  - f = (1/3) Mod 7
  - So (3f)Mod 7 = 1
  - Possible values of f:
    - 0, 0 Mod 7 = 0
    - 1, 3 Mod 7 = 3
    - 2, 6 Mod 7 = 6
    - 3, 9 Mod 7 = 2
    - 4, 12 Mod 7 = 5
    - **5, 15 Mod 7 = 1**
    - 6, 18 Mod 7 = 4
  - So f = (1/3) Mod 7 = 5

UNDEFINED MODULAR DIVISION

- What about
  - f= (5/5) Mod 10
  - (5f) Mod 10 = 5

UNDEFINED MODULAR DIVISION

- What about
  - f= (5/5) Mod 10
  - (5f) Mod 10 = 5

  - 0, 0 Mod 10 = 0
  - 1, 5 Mod 10 = 5
  - 2, 10 Mod 10 = 0
  - 3, 15 Mod 10 = 5
  - 4, 20 Mod 10 = 0

Etc

Undefined – too many prime answers!
There are cases where there are no answers!

PRIMES AND MODULUS

- Let p be a prime number
- Let a be an integer

- iff a ≢ 0 mod p

There is always another integer d such that:

$$a * d = 1 \mod p$$

FERMAT'S LITTLE THEOREM

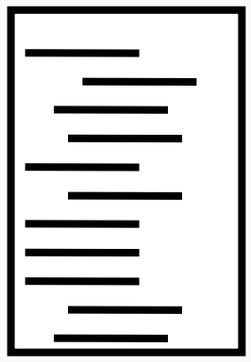
- iff $a \not\equiv 0 \bmod p$

$$a^{p-1} = 1 \bmod p$$

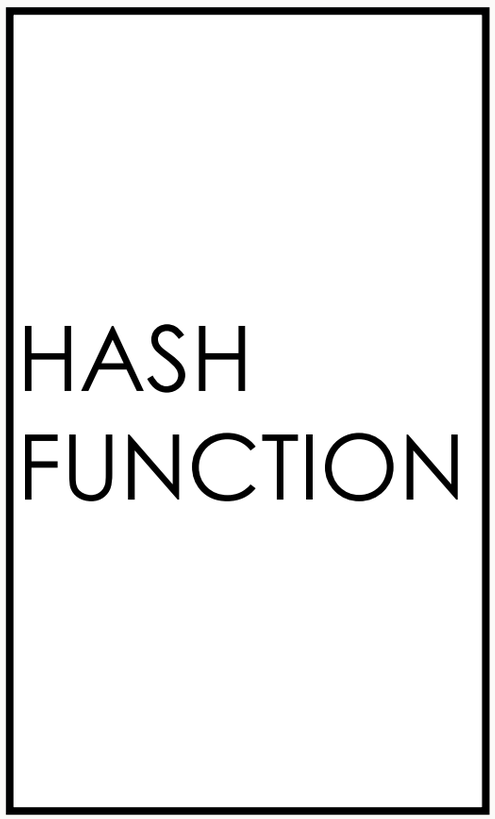
- This yields a good way of testing if a number is prime
  - If you calculate a^{p-1} (mod p) for a series of numbers if there are no 1's it probably isn't prime!

HASH FUNCTIONS

X



Any number of bits



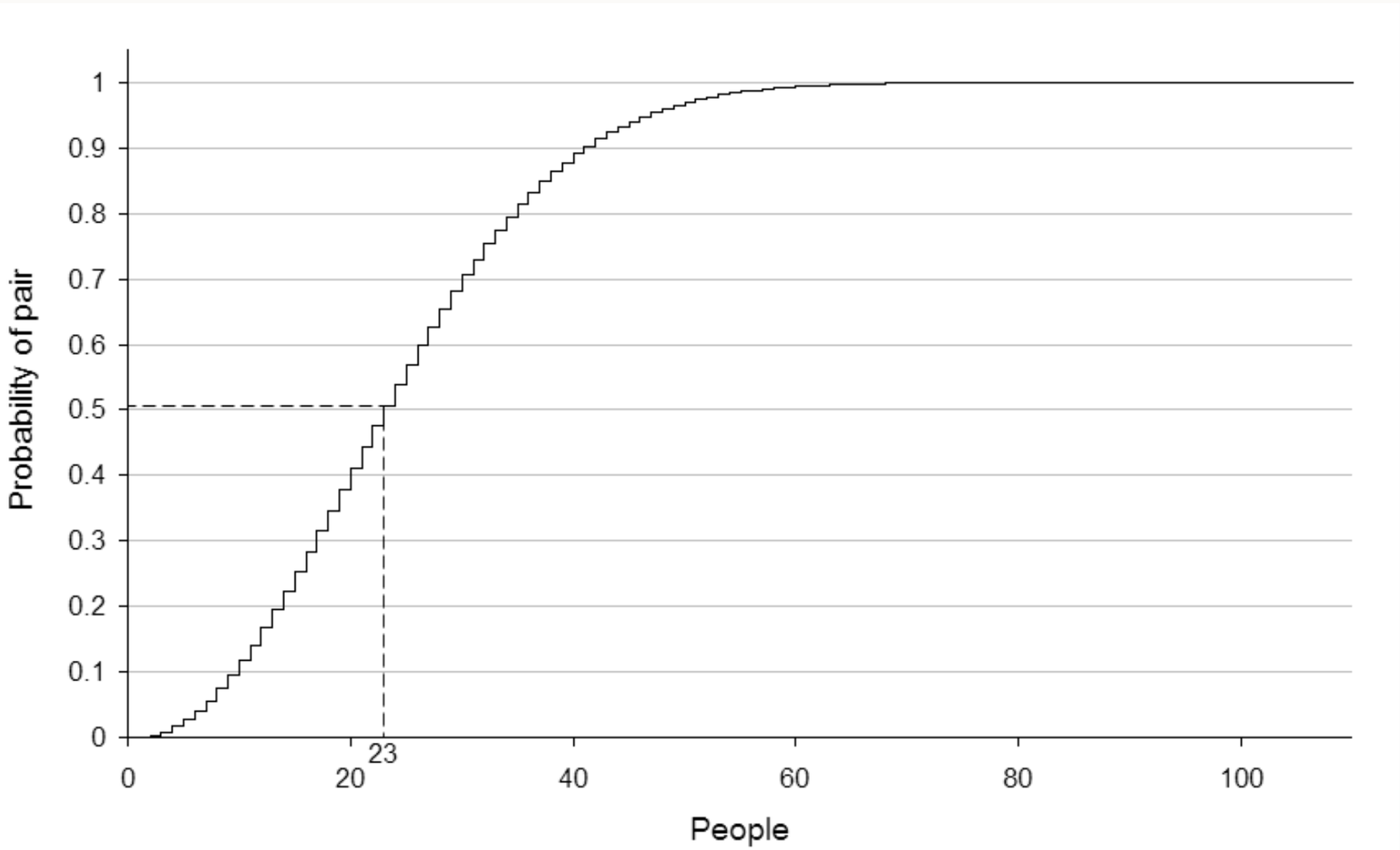
HASH FUNCTION PROPERTIES

- Compression
  - No matter how long the input is, the output has the same length
- Ease of computability
  - Given x, it should be easy to find h(x)
- Collision Avoidance
  - It should be "computationally infeasible" to find collisions

QUESTION/ACTIVITY …

THE BIRTHDAY PARADOX

- How many of you share a birthday?



HASH FUNCTION PROPERTIES

- Given a hash function that produces N bit hashes
  - If you generate around $2^{N/2}$ random inputs, you are likely to find a collision

HASH FUNCTION PROPERTIES

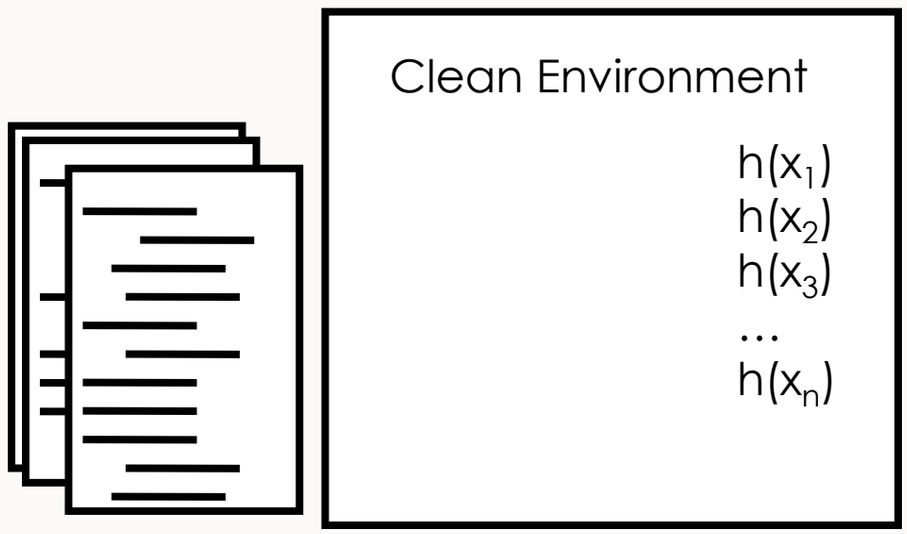
- Preimage Resistance
  - Given y, it should be "computationally infeasible" to find x to satisfy:
    - h(x) = y
    - Expected number of tries is 2^{n-1} for an n-bit hash
- Second Preimage Resistance
  - (weak collision resistance)
  - Given x and h(x), it should be "computationally infeasible" to find x' to satisfy
    - h(x) = h(x')

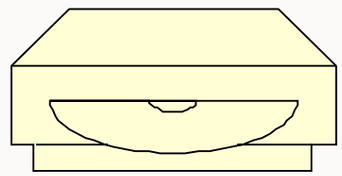
HASH FUNCTION PROPERTIES

- Collision Resistance
  - (strong collision resistance)
  - It should be "computationally infeasible" to find any x and x' that satisfy:
  - h(x) = h(x')

MDC-2

- Modification Detection Codes 2
- A cryptographic hash function





SUMMARY

- What is cryptology?
- Communication/encryption paradigms
- Modulus Mathematics
- Hash functions
- MDCs

Read:

1. Gollman: Chapter 14
2. Anderson: Chapter 5

COMP3052.SEC Computer Security

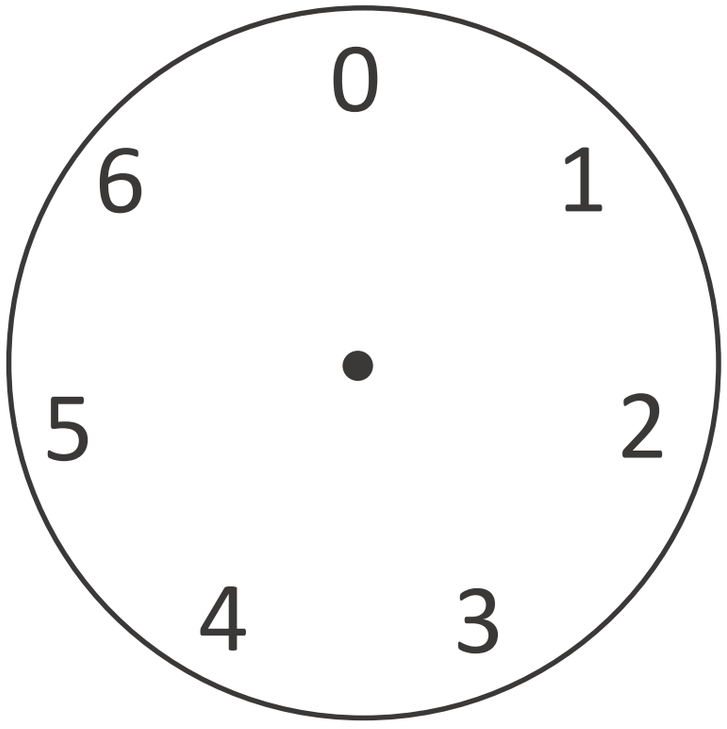
Session 15-2 Crypto II: EXTRA MATHS MATERIALS

Acknowledgements

- Some of the materials we use this semester may come directly from previous teachers of this module, and other sources ...
- Thank you to (amongst others):
  - Michel Valstar, Milena Radenkovic, Michael Pound, Dave Towey …

Modular Arithmetic

- A system of arithmetic based around cycles of numbers
  - Numbers modulo n are a finite field



The set of numbers modulo 7

The Congruence Relation

$$a \equiv b \pmod{n}$$

$$a \pmod{n} = b \pmod{n}$$

Equivalences

$$\big((\mathrm{a} \bmod \mathrm{n}) + (b \bmod n)\big) \bmod n = (a + b) \bmod n$$

$$\big((\mathrm{a} \bmod \mathrm{n}) \cdot (b \bmod n)\big) \bmod n = (a \cdot b) \bmod n$$

Multiplication Example

Rule:

$$\big((\text{a mod n}) \cdot (b \bmod n)\big) \bmod n = (a \cdot b) \bmod n$$

Example: $(29013 \cdot 1123) \bmod 7$

$32581599 \bmod 7$

Multiplication Example

Rule:

$$\big((\text{a mod n}) \cdot (b \bmod n)\big) \bmod n = (a \cdot b) \bmod n$$

Example: $(29013 \cdot 1123) \bmod 7$

$32581599 \bmod 7$

Or: $\big((29013 \bmod 7) \cdot (1123 \bmod 7)\big) \bmod 7$

$(5 \cdot 3) \bmod 7$

$15 \bmod 7 = \mathbf{1}$

Exponentiation

Example: $13^{11} \; mod \; 7 = ?$

Exponentiation

Example: $13^{11} \; mod \; 7 = ?$

$$\left((13^2 \; mod \; 7) \cdot (13^9 \; mod \; 7)\right) mod \; 7$$

$$169 \; mod \; 7 = 1$$

Exponentiation

Example: $13^{11} \; mod \; 7 = ?$

$$\left((13^2 \; mod \; 7) \cdot (13^9 \; mod \; 7)\right) mod \; 7$$

$$\left(1 \cdot (13^9 \; mod \; 7)\right) mod \; 7$$

$$\left(1 \cdot (13^2 \; mod \; 7) \cdot (13^7 \; mod \; 7)\right) mod \; 7$$

etc.

Exponentiation

Example: $13^{11} \; mod \; 7 = ?$

$$\left((13^2 \; mod \; 7) \cdot (13^9 \; mod \; 7)\right) mod \; 7$$

$$\left(1 \cdot (13^9 \; mod \; 7)\right) mod \; 7$$

$$\left(1 \cdot (13^2 \; mod \; 7) \cdot (13^7 \; mod \; 7)\right) mod \; 7$$

etc.

$$\left(1 \cdot (13^1 \; mod \; 7)\right) mod \; 7$$

$$13 \; mod \; 7 = \mathbf{6}$$

Logarithms

- A logarithm is the inverse function to exponentiation:

$$a^b = c$$

$$b = \log_a(c)$$

- This is easy to compute even for large numbers

Discrete Logarithms

- When operating mod n, we call the operation a discrete logarithm:

$$a^b = c \pmod{n}$$

$$b = \mathrm{dlog}_{a,n}(c)$$

Example:

$$7^2 = 4 \pmod{9}$$

$$2 = \mathrm{dlog}_{7,9}(4)$$

Discrete Logarithms

- Discrete logs are much harder to compute

$$3^{?} = 1 \pmod 7$$

$$? = \mathrm{dlog}_{3,7}(1)$$

Discrete Logarithms

- Discrete logs are much harder to compute

$$3^{?} = 1 \pmod 7$$

$$? = \mathrm{dlog}_{3,7}(1)$$

Brute force:

$$3^1 = 3 \pmod 7$$

$$3^2 = 2 \pmod 7$$

$$3^3 = 6 \pmod 7$$

$$3^4 = 4 \pmod 7$$

$$3^5 = 5 \pmod 7$$

$$3^6 = 1 \pmod 7$$

Summary

- This very brief session was really just to give you a taste (hopefully a reminder) of the kind of maths to be ready to do

COMP3052.SEC Computer Security

Session 15-3 Crypto III

ACKNOWLEDGEMENTS

- Some of the materials we use this semester may come directly from previous teachers of this module, and other sources ...

- Thank you to (amongst others):
  - Michel Valstar, Milena Radenkovic, Mike Pound, Dave Towey…

TOPICS COVERED

- What is encryption?
- Primitive types
- Historic ciphers
- The One Time Pad
- Stream ciphers

WHAT DOES THIS MEAN?

Security Overview

This page is secure (valid HTTPS).

Valid Certificate

The connection to this site is using a valid, trusted server certificate.

View certificate

Secure Connection

The connection to this site is encrypted and authenticated using a strong protocol (TLS 1.2), a strong key exchange (ECDHE_RSA with P-256), and a strong cipher (AES_128_GCM).

Secure Resources

All resources on this page are served securely.

ENCRYPTION

- Encryption: We encode a message such that only authorised users may read it
- Cipher: takes a string of plaintext, and converts it into a string of ciphertext
- Encryption can provide:
  - Confidentiality
  - Integrity
  - Authenticity

NOTATION

- A cipher converts a plaintext message *M* into a ciphertext *C* under the control of a key *K*
- C is not a secret, but without knowledge of the key, it should be impossible to reconstruct M
- Comes in two forms:
  - Symmetric – same key for encryption / decryption
  - Asymmetric – separate keys

PRIMITIVE TYPES

- Stream Ciphers
  - Operate on a stream of input data
- Block Ciphers
  - Operate on a fixed sized block
- Hash Functions
  - Take data of any size and output a block of fixed size

THE CAESAR CIPHER

- An early substitution cipher, we replace each letter of plaintext with a shifted letter further down the alphabet
- Vulnerable to frequency analysis

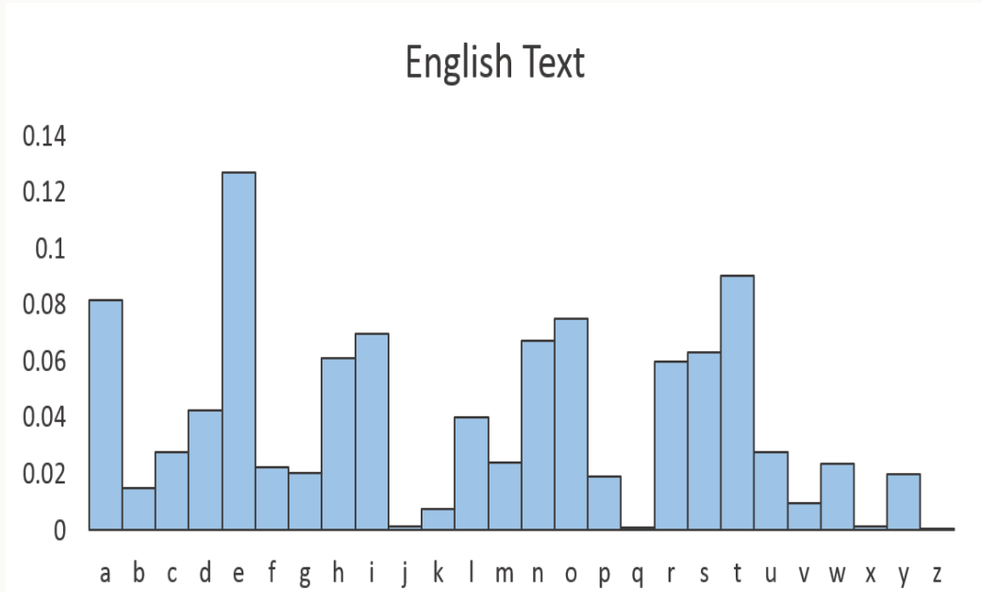
s	e	c	u	r	i	t	y

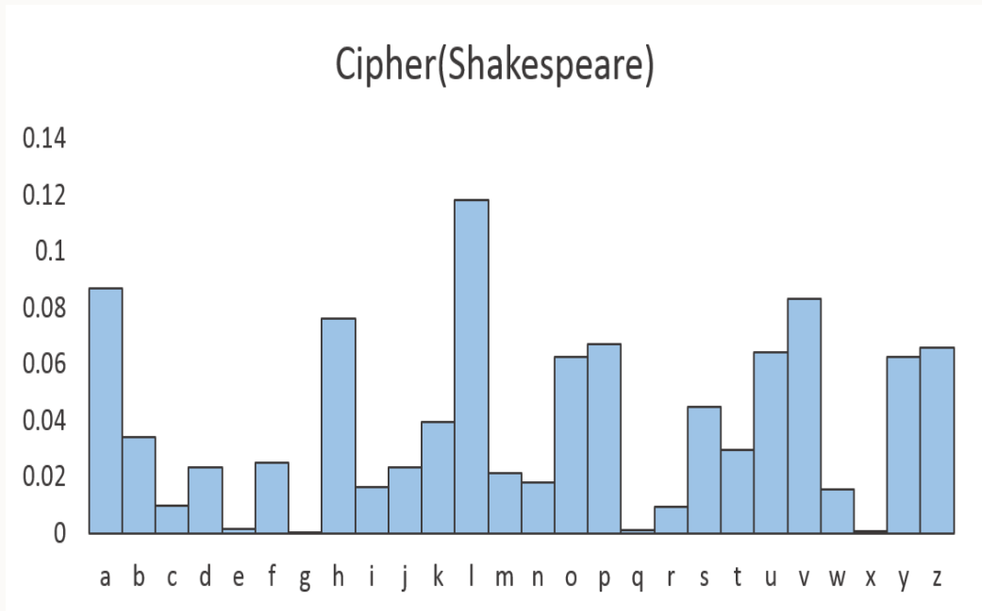
Key = 4 (‘E’)

W	I	G	Y	V	M	X	C

FREQUENCY ANALYSIS

- The frequency of occurrences of each character are very consistent across the same language
- The longer the ciphertext, the easier this becomes





MONOALPHABETIC SUBSTITUTION

- A *slightly* better approach, a key is used to alter the cipher alphabet:

```
Abcdefghijklmnopqrstuvwxyz
SECRTKYABDFGHIJLMNOPQUVWXZ
```

- Still extremely vulnerable to frequency analysis

THE VIGENÈRE CIPHER
(POLYALPHABETIC SUBSTITUTION)

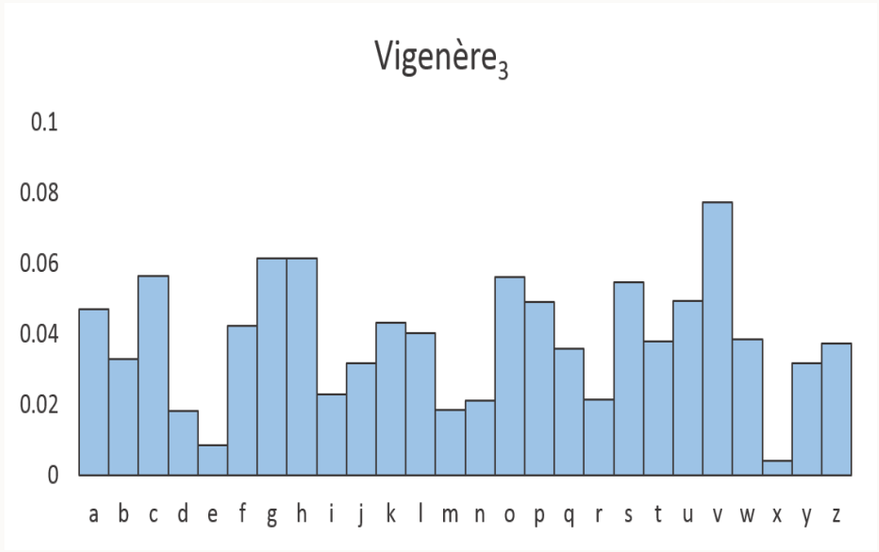
- An early stream cipher, the Vigenère cipher adds the plaintext to a key modulo 26:

- Unlike the Caesar cipher, the key is repeated for as long as is required

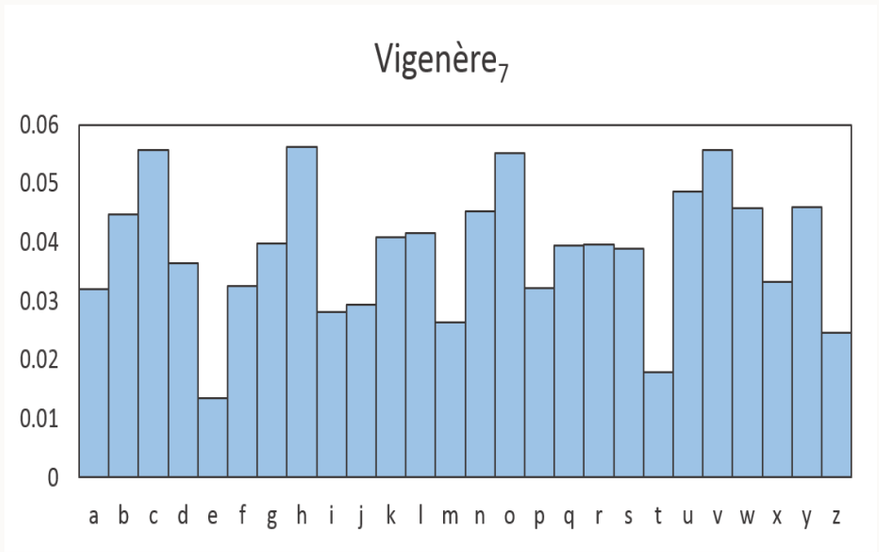
```
tobeornottobethatisthequestion
keykeykeykeykeykeykeykeykeykeykey
DSZOSPXSRDSZOXFKXGCXFOUSOWRSSL
```

THE VIGENÈRE CIPHER

- Equivalent to multiple interleaved Caesar ciphers



- Spreads out the occurrences of characters making frequency analysis hard



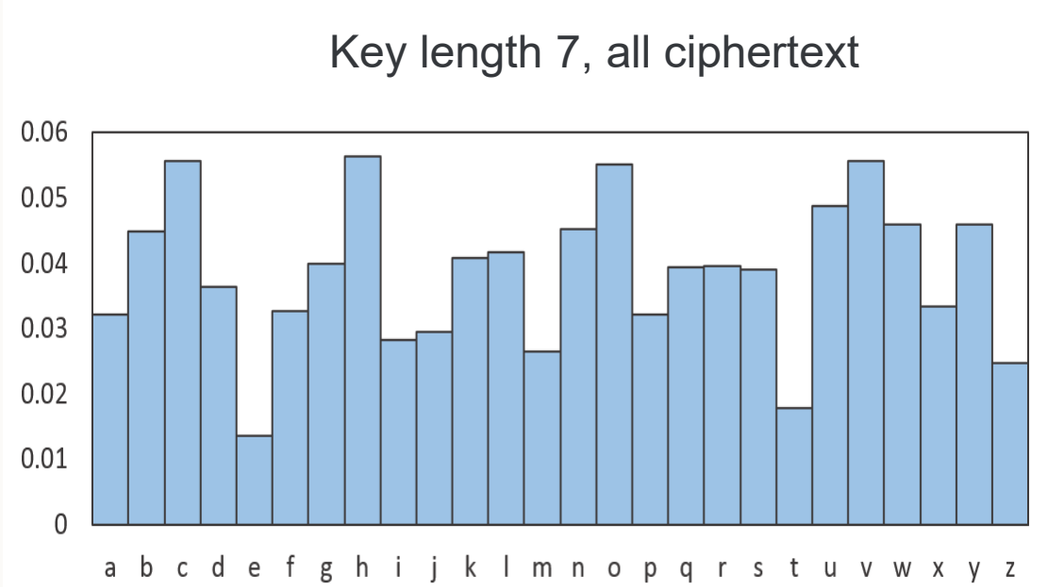
THE VIGENÈRE CIPHER

- This cipher is weak to Kasiski examination:
  - Repeated phrases in the ciphertext give away clues as to the length of the running key
  - Once the length of the key is known, simple frequency analysis can be performed

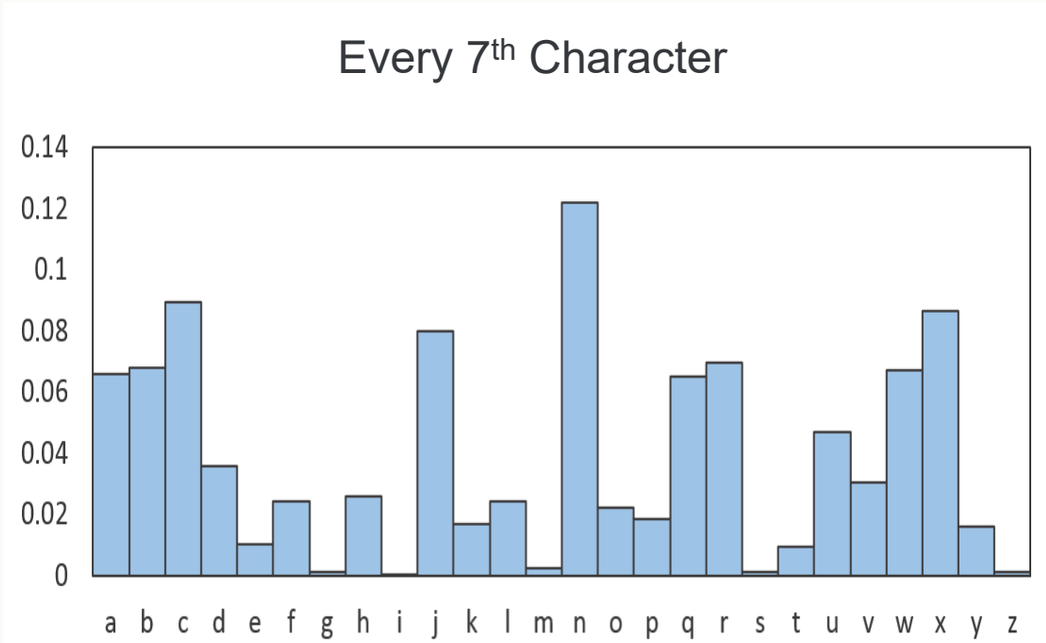
```
tobeornottobethatisthequestion
keykeykeykeykeykeykeykeykeykey
DSZOSPXSRDSZOXFKXGCXFOUSOWRSSL
```

THE VIGENÈRE CIPHER

- Frequency analysis of the Shakespeare text:



- After determining the key length, frequency analysis is straightforward



PLAYFAIR

- Playfair is an early block cipher
- By using blocks of two characters (digrams) we increase the number of mappings to 650 (=26x25)
- This makes frequency analysis much harder
- Nice example on Wikipedia:
  https://en.wikipedia.org/wiki/Playfair_cipher

PLAYFAIR

- Playfair is an early block cipher
- By using blocks of two characters (digrams), we increase the number of mappings to 650
- This makes frequency analysis much harder

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

attackatzerofourforty

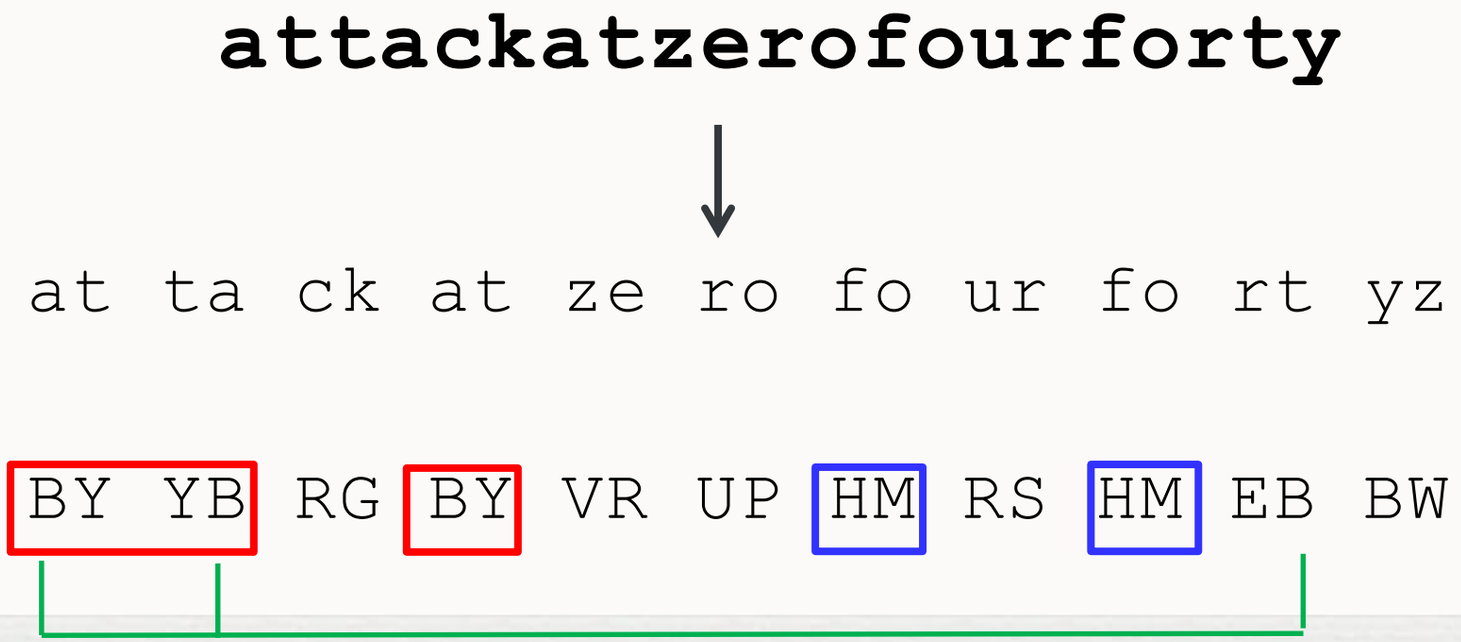
at ta ck at ze ro fo ur fo rt yz

BY YB RG BY VR UP HM RS HM EB BW

PLAYFAIR

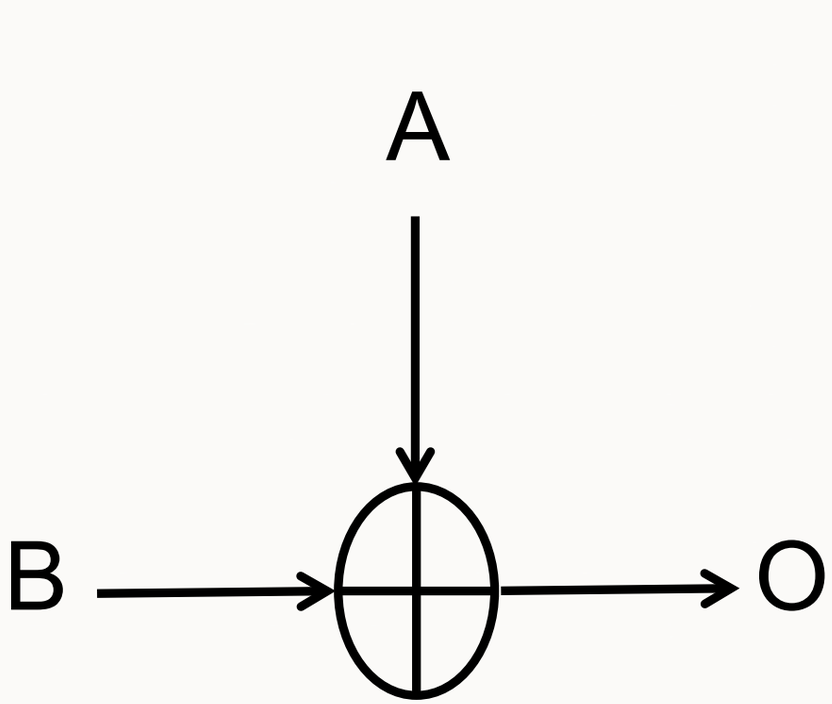
- Block size not sufficient – can piece together table
- Reverse plaintext leads to reversed ciphertext
- Probable plaintext can be guessed
- Changes often propagate to only one character

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



XOR

"XOR is a binary operator between two values that returns true if either one input or the other is true, but not both"



A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

Think of A determining if B flips when output

WHY XOR IS COOL!

- Applying XOR twice, reverses its effect:

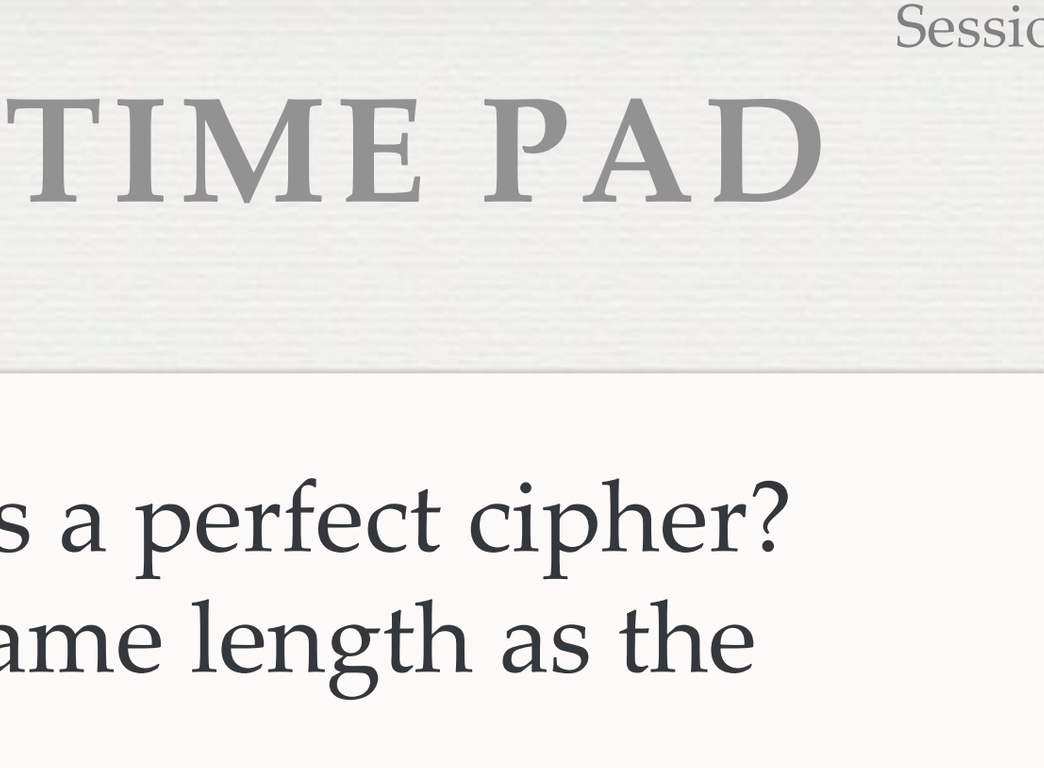
$$A \oplus B \oplus A = ((A \oplus A) \oplus B)$$
$$= 0 \oplus B$$
$$= B$$

- A “encrypts” B, and then “decrypts” it again

THE ONE-TIME PAD

- Is there such a thing as a perfect cipher?
- Use a key that is the same length as the message
  - The one-time pad is the only example of perfect secrecy

```
M  iloveyou
K  emrqytpn
C  MXFLCRDH
```



THE ONE-TIME PAD

Can we design a cipher that uses XOR to encrypt and decrypt a message?

- Use a key that has the same length as the message
- XOR each message bit with each key bit

M 01011010 00110101

⊕

K 01001011 10111001

=

C 00010001 10001100

Perfect Secrecy

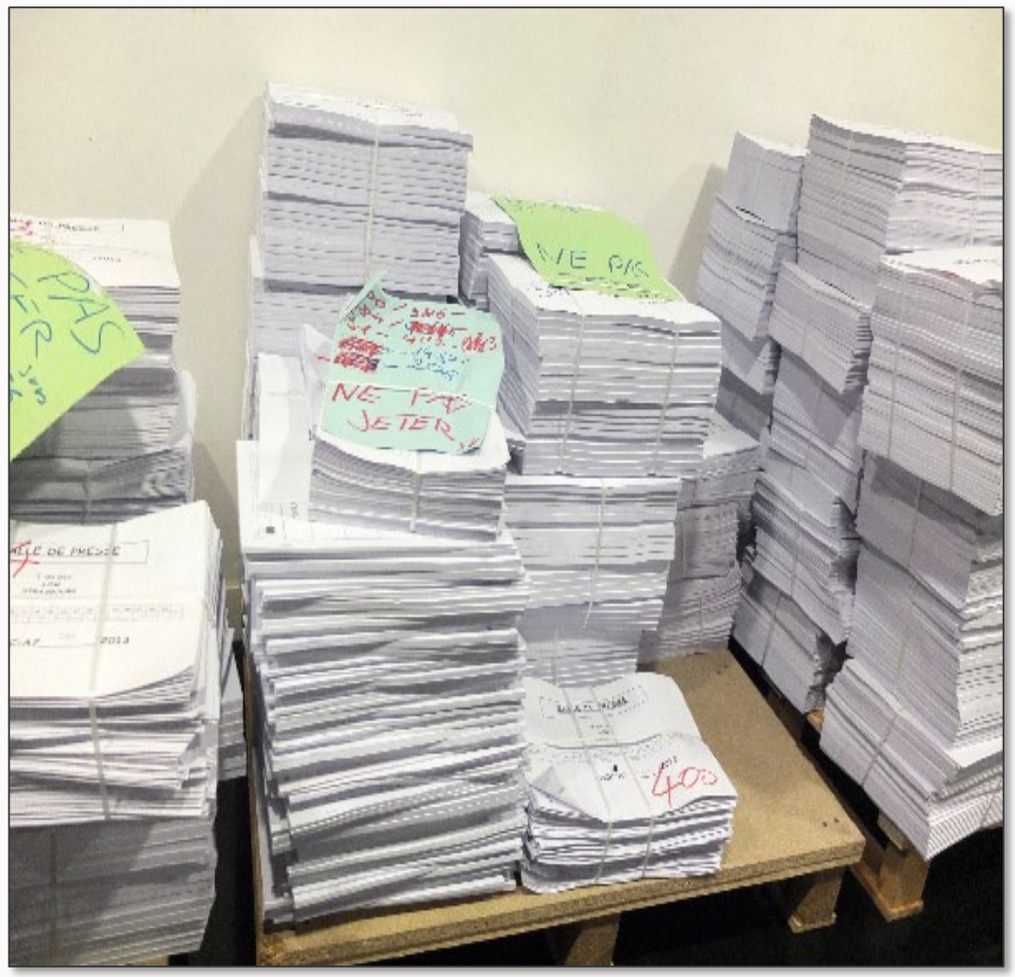
THE ONE-TIME PAD

- Any possible plaintext can be recovered depending on the key
- Brute force is pointless
- Example with M = (C – K) mod 26

C	**MXFLCRDH**	**MXFLCRDH**
K	**emrqytpn**	**eqfsytpn**
M	**iloveyou**	**ihateyou**

BUT…

- The one time pad is not practical:
  - A 1GB file would need a 1GB key!
  - How are we transporting these keys? Or storing them?
  - If you ever reuse a key, the entire cipher is broken

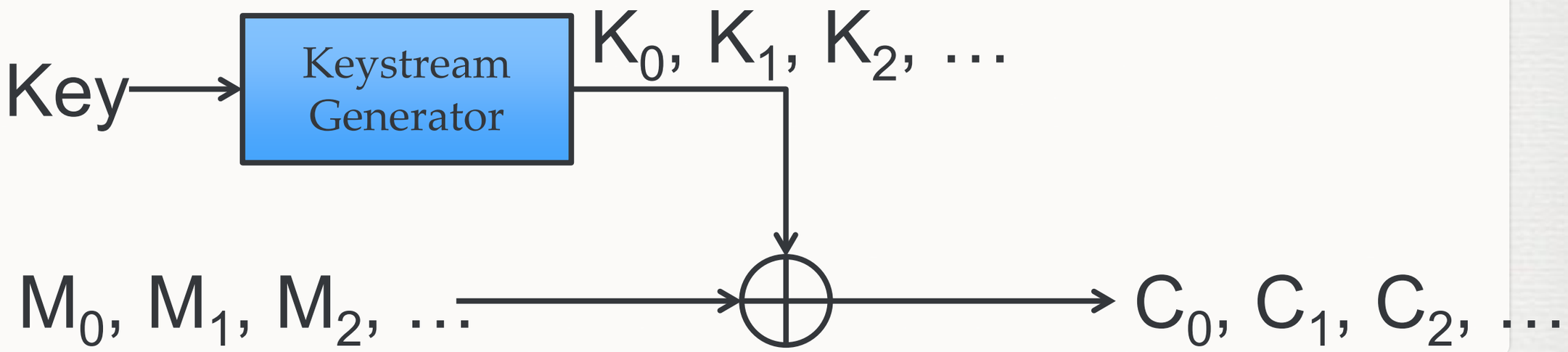


THE ONE-TIME PAD

- What the one-time pad gives us in secrecy, it lacks in:
  - Portability – the key has the same size as the message
  - Convenience – you must *never* reuse a key
- Modern stream ciphers attempt to approximate the one-time pad with a pseudorandom key stream

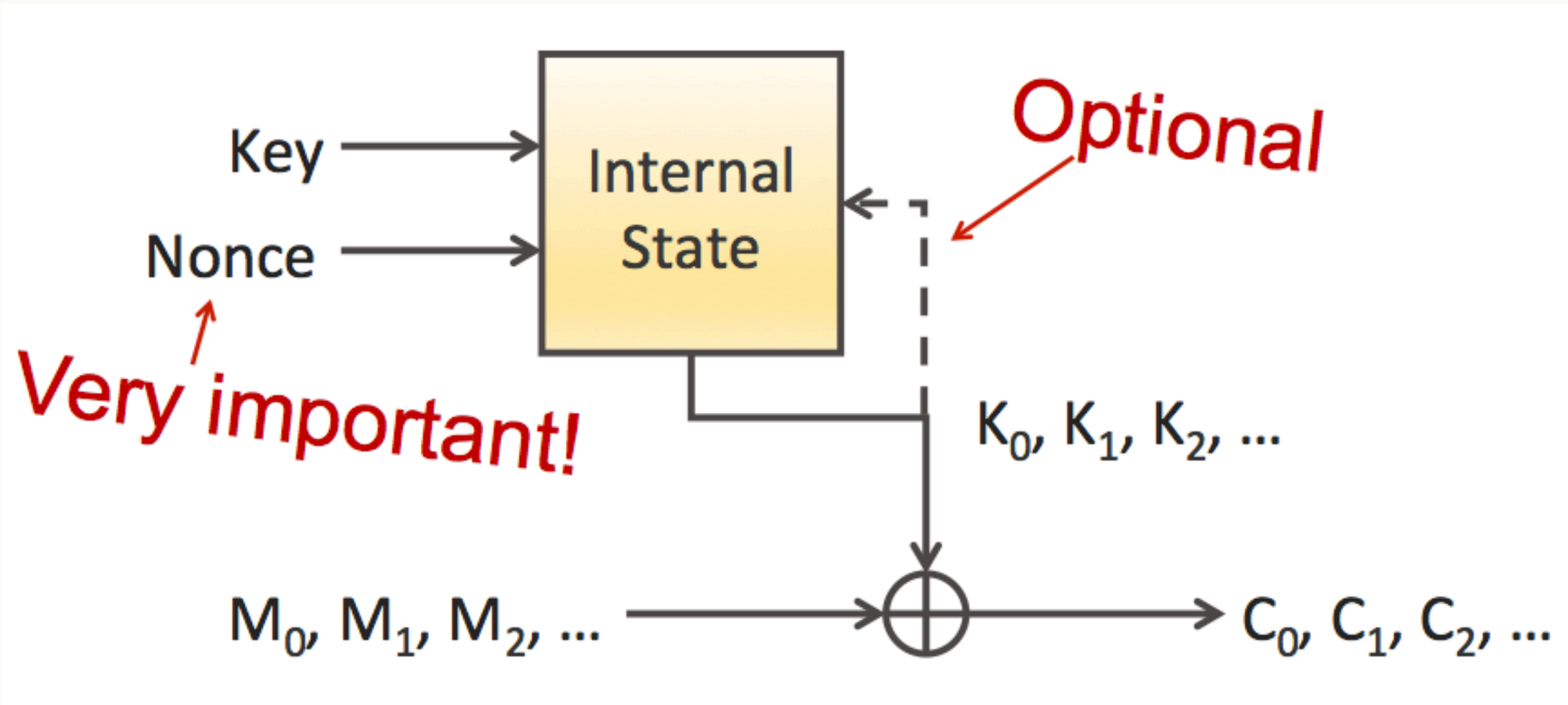
MODERN STREAM CIPHERS

- Modern stream ciphers use an initial seed key to generate an infinite pseudorandom keystream
- The message and keystream are usually combined using XOR - $\oplus$ - which is reversible if applied twice



MODERN STREAM CIPHERS

- Common to seed an initial state using a key, then update this state for as long as needed:

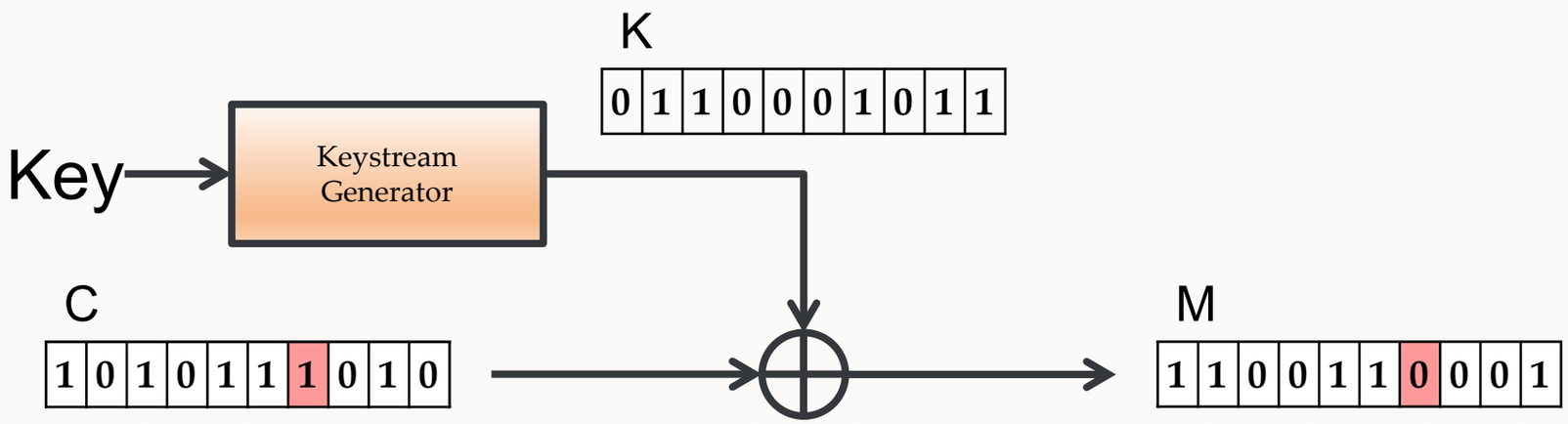


MODERN STREAM CIPHERS

- Stream ciphers are usually reserved for situations where:
  - Hardware resources might be limited
  - The message stream is of unknown length, but long and continuous
    - For example, GSM mobile communications, Bluetooth, …
- Extremely fast with a low memory footprint, ideal for low-power devices
- If designed well, can seek to any location in the stream
  - E.g. Streaming video with DRM

MODERN STREAM CIPHERS

- Stream ciphers give us confidentiality, but not integrity
- We must include another mechanism



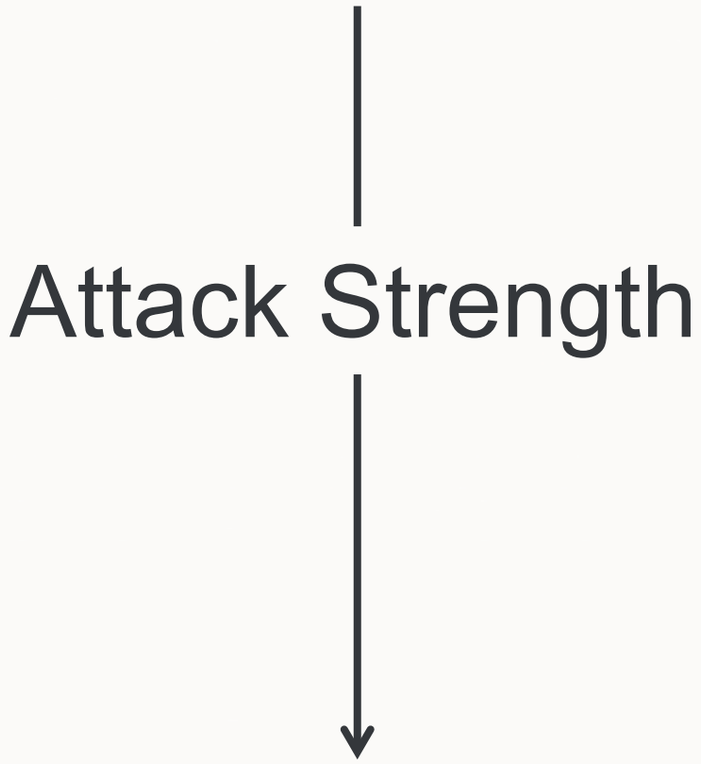
KERCKHOFFS' PRINCIPLE

"A cryptographic system must be secure even if everything is known about the system with the exception of the secret key"

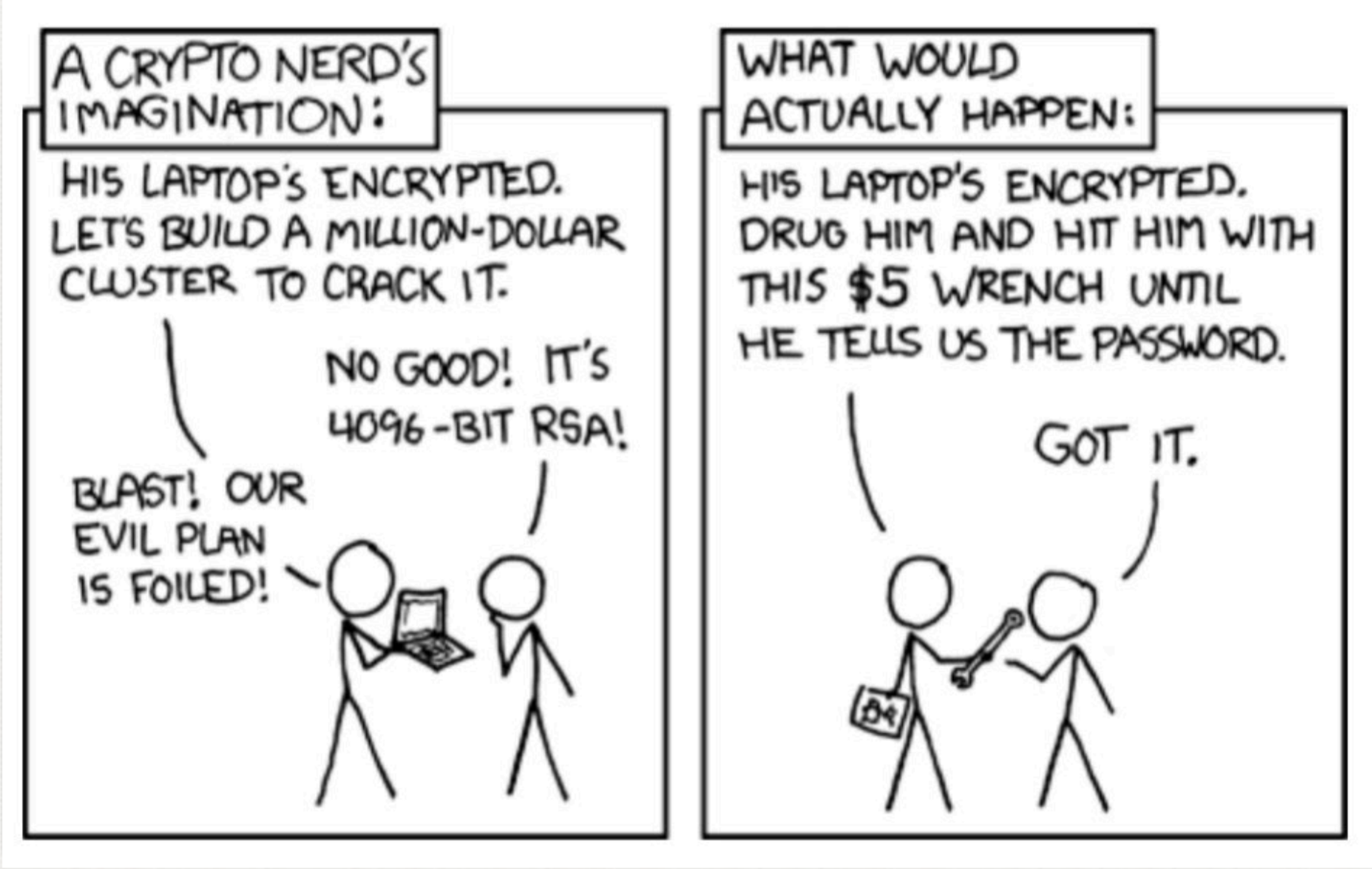
- Algorithms can be published, but will work providing that the key remains secret - rather than security through obscurity

CRYPTOGRAPHIC ATTACK MODELS

- If we know an algorithm inside-out, what can we do to attack it?
  - Brute-force
  - Ciphertext-only
  - Known-plaintext
  - Chosen-plaintext
  - Chosen-ciphertext
  - Related-key attack



BUT ...


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

SUMMARY

- What is encryption?
- Primitive types
- Historic ciphers
- The One Time Pad
- Stream ciphers

Read:

. Gollman: Chapter 14
. Anderson: Chapter 5

COMP3052.SEC Computer Security

Session 15-4: Cryptology IV

ACKNOWLEDGEMENTS

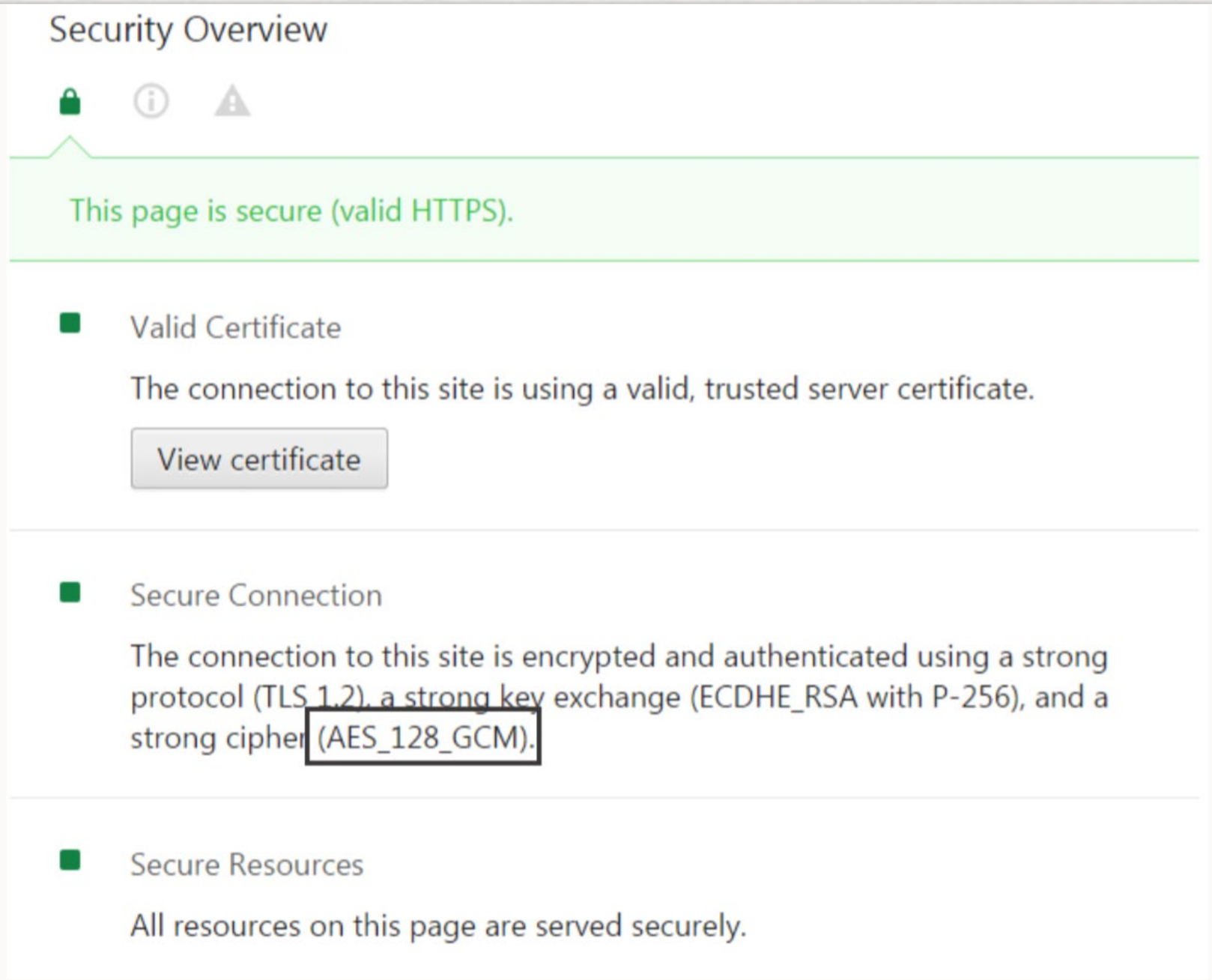
- Some of the materials we use this semester may come directly from previous teachers of this module, and other sources ...

- Thank you to (amongst others):
  - Michel Valstar, Milena Radenkovic, Mike Pound, Dave Towey…

TOPICS COVERED

- Block Ciphers
  - DES -> AES
  - Modes of Operation
- Public Key Crypto
- Maths …but don't panic!

WHAT DOES THIS MEAN?



BLOCK CIPHERS

- Block ciphers use a key to encrypt a fixed-size block of plaintext into a fixed-size block of ciphertext
- They are usually more computationally expensive than stream ciphers, but have numerous benefits
- If you're careful, you can convert between block and stream ciphers using modes of operation

BLOCK CIPHERS

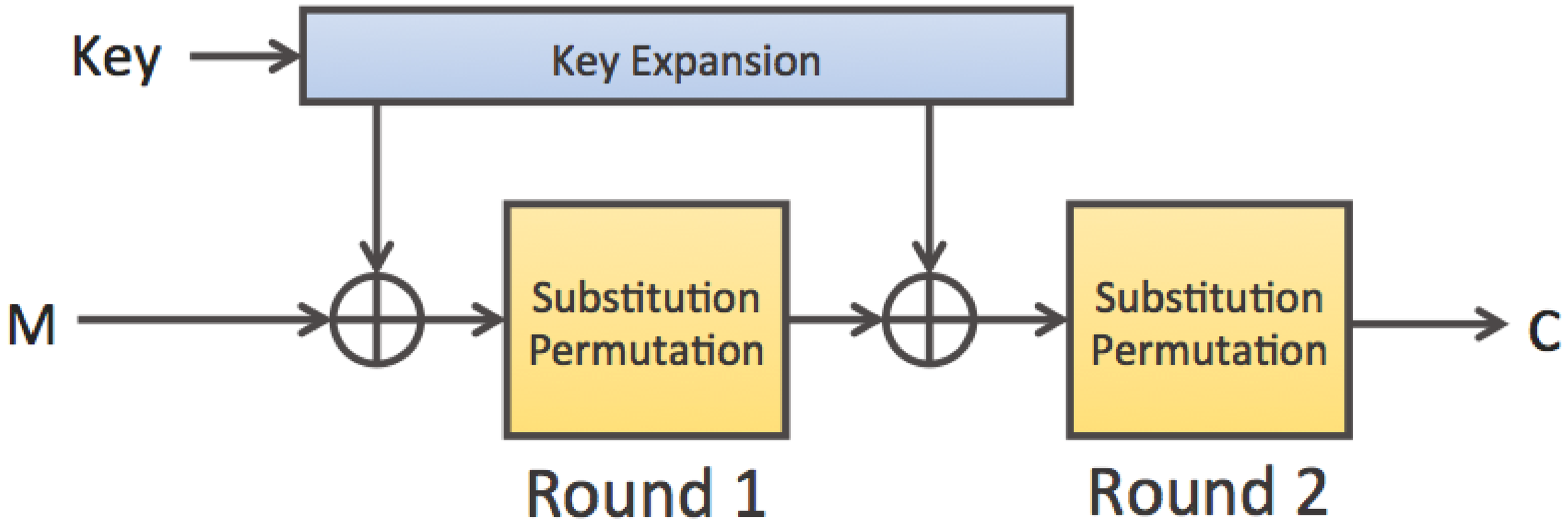
- If we change one bit of plaintext in a traditional cipher, one bit of output will change
  - Vulnerable to known and chosen plaintext attacks
- Modern block ciphers are designed to diffuse changes throughout each block

SP-NETWORKS

- Claude Shannon suggested that all that was required for a strong cipher was repeated substitution and permutation
- SP-Networks combine a substitution process with a permutation into a single round
- Rounds are then repeated enough times to ensure the algorithm is secure

SP-NETWORKS

- Can add a key using XOR:



FEISTEL CIPHERS

- Developed by Horst Feistel in the 1970s at IBM
- Allows us to chain multiple rounds together, using any round function
- The basis of many block ciphers
  - Blowfish, **DES**, CAST-128, GOST 28147-89

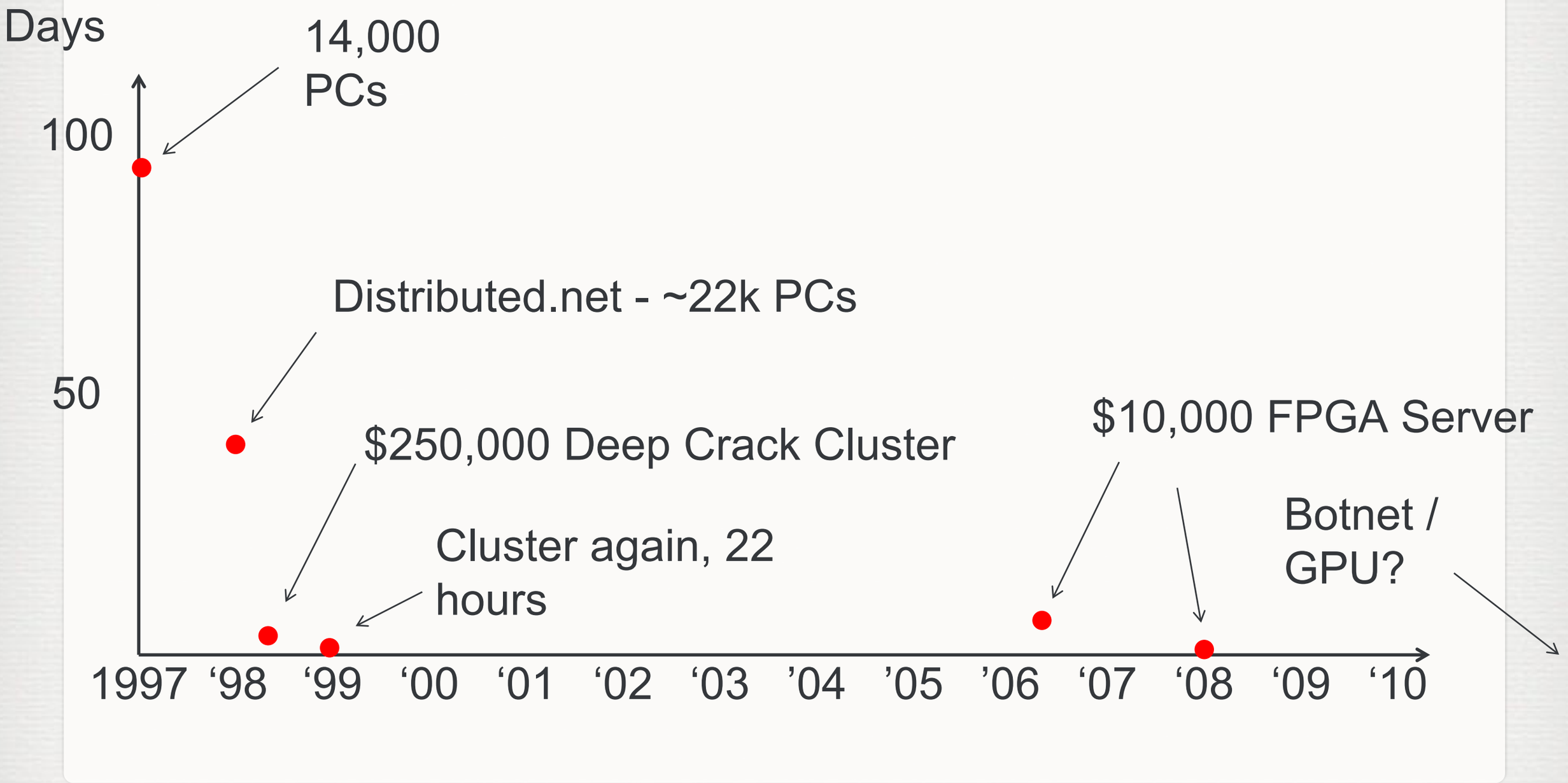
LUBY-RACKOFF

- Mike Luby and Charlie Rackoff showed that if you have well-designed rounds (appear statistically random) in your Feistel cipher:
  - 3 rounds is sufficient to appear so random that a chosen plaintext attack won't work
  - 4 rounds is sufficient to beat a chosen plaintext *and* ciphertext attack

DATA ENCRYPTION STANDARD

- DES was, until quite recently, one of the most used symmetric ciphers
- It is a 64-bit blocklength, 16 round Feistel cipher, with a 56-bit key
- Developed by IBM in 1970s, with a bit of interference from the NSA
- Once released, researchers were suspicious of some secrecy regarding the design, and the short key

CRACKING DES



ADVANCED ENCRYPTION STANDARD

- Superseded DES as a standard in 2002
- A standard built around the Rijndael algorithm
- Rijndael is an SP-Network with a 128-bit block size, and a key length of 128, 192 or 256-bits
- Round count depends on key length
  - 10, 12 or 14 cycles

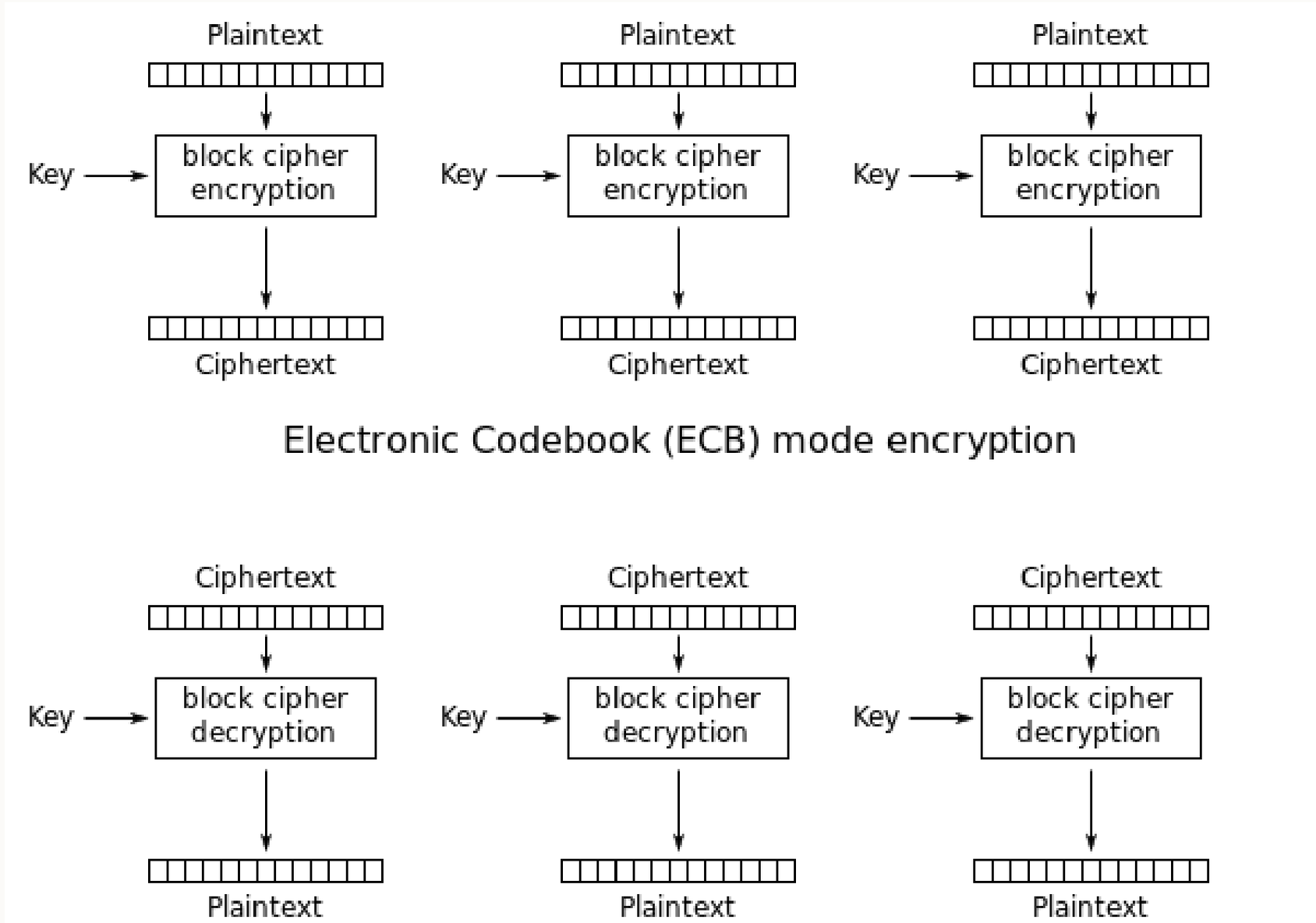
AES SECURITY

- Is currently the standard algorithm for symmetric encryption, and is likely to remain that way
- 2^{128} keys, let's say we get lucky and crack it at 2^{127}
- Let's say 1000 flops per key test, we'd break an AES key in about 30 trillion years

BLOCK CIPHER MODES

- Most messages aren't in convenient 128-blocks
  - Run a block cipher repeatedly on consecutive blocks
    - Electronic Code Book (ECB)
      - encrypt each block one after another
    - Cipher Block Chaining (CBC)
      - XOR the output of each cipher block with the next input
    - Counter Mode (CTR)
      - Encrypting a counter to produce a stream cipher
    - Galois Counter Mode (GCM)
      - Extension of CTR

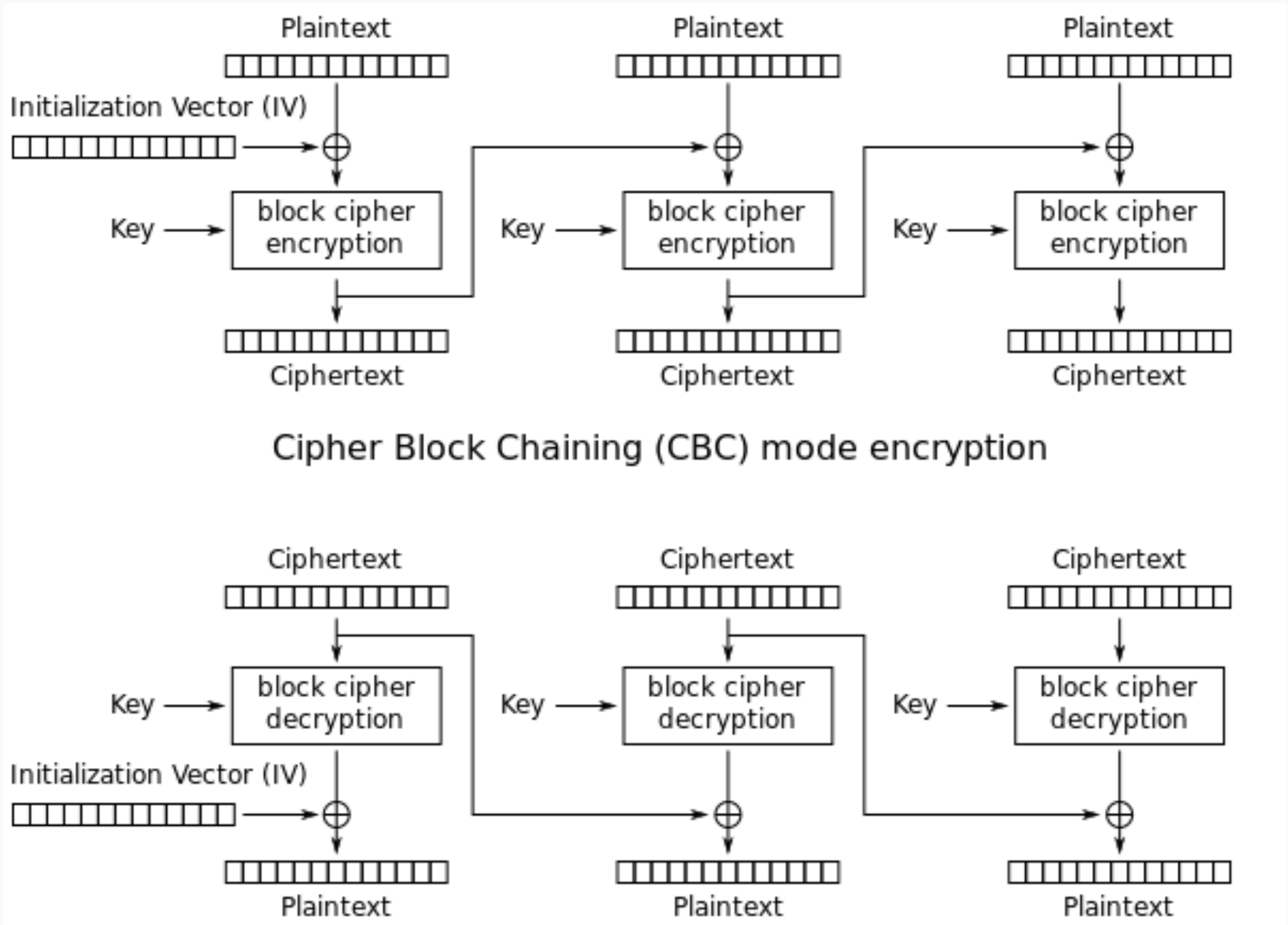
ELECTRONIC CODE BOOK



Electronic Codebook (ECB) mode encryption

Electronic Codebook (ECB) mode decryption

CIPHER BLOCK CHAINING

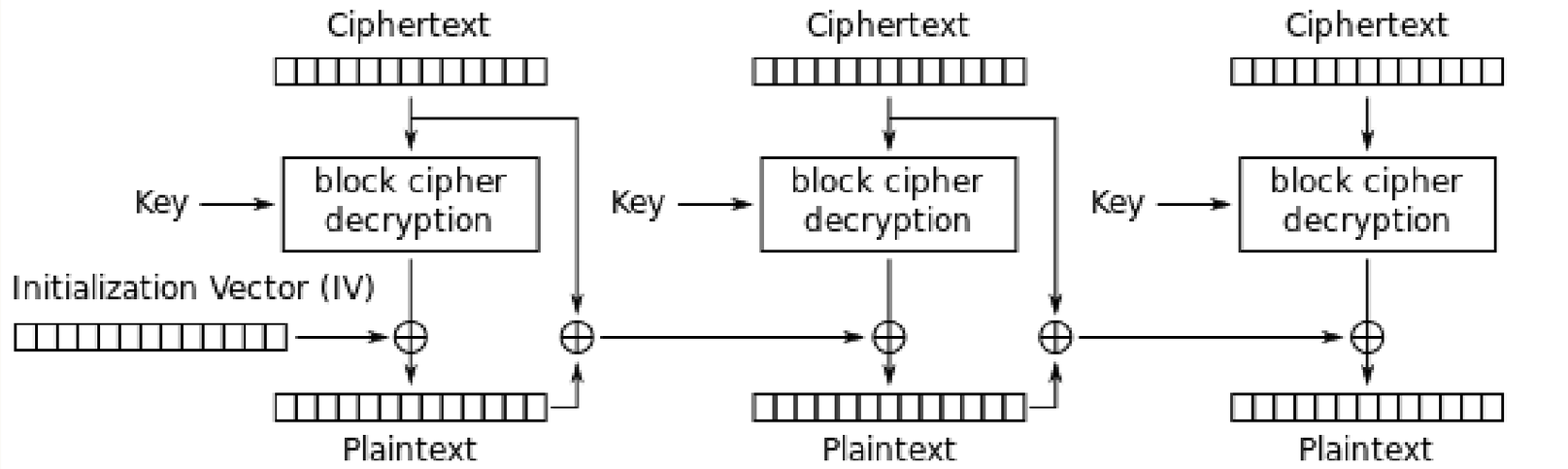


Cipher Block Chaining (CBC) mode encryption

Cipher Block Chaining (CBC) mode decryption

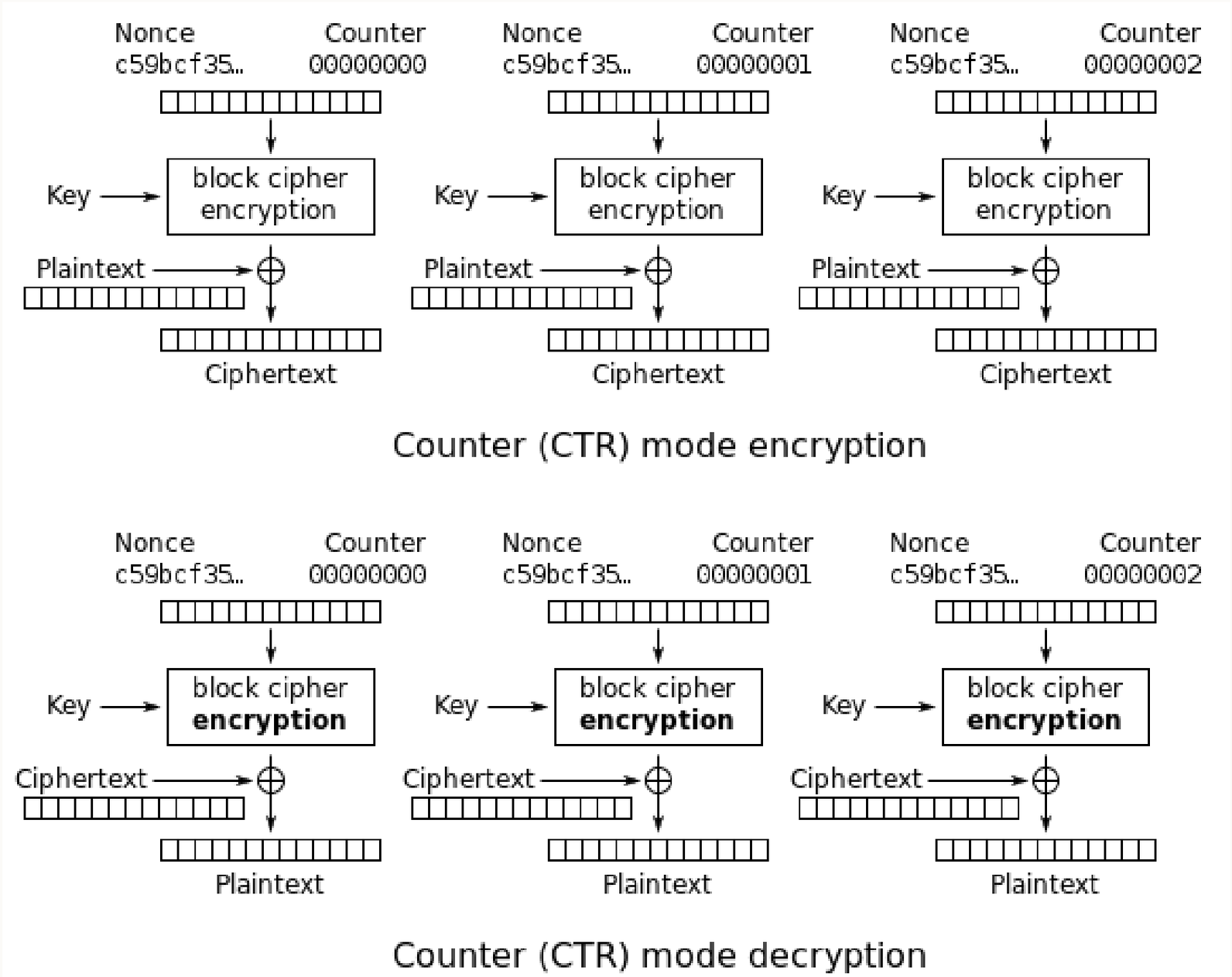
PROPAGATING CBC

Propagating Cipher Block Chaining (PCBC) mode encryption



Propagating Cipher Block Chaining (PCBC) mode decryption

COUNTER MODE



WHAT DOES THIS MEAN?

Security Overview

This page is secure (valid HTTPS).

Valid Certificate

The connection to this site is using a valid, trusted server certificate.

View certificate

Secure Connection

The connection to this site is encrypted and authenticated using a strong protocol (TLS 1.2), a strong key exchange (ECDHE_RSA with P-256), and a strong cipher (AES_128_GCM).

Secure Resources

All resources on this page are served securely.

Advanced Encryption Standard
(Rjindael)
128-bit Key Size
Galois Counter Mode

WHAT DOES THIS MEAN?

Security Overview

This page is secure (valid HTTPS).

Valid Certificate

The connection to this site is using a valid, trusted server certificate.

View certificate

Secure Connection

The connection to this site is encrypted and authenticated using a strong protocol (TLS 1.2), a strong key exchange (ECDHE_RSA with P-256), and a strong cipher (AES_128_GCM).

Secure Resources

All resources on this page are served securely.

Advanced Encryption Standard
(Rjindael)
128-bit Key Size
Galois Counter Mode

PUBLIC-KEY CRYPTOGRAPHY

- Two keys, a public key and a private key
- Public-key (asymmetric) cryptography hinges upon the premise that:

It is computationally infeasible to calculate a private from a public key

- In practice, this is achieved through intractable mathematical problems

WHY?

- Public-key cryptography gains us a few important abilities:
  - We can exchange a private symmetric key "in the open"
  - We can verify the sender of a message
  - Non-repudiation – you can't deny you did something

SYMMETRIC VS. ASYMMETRIC

Symmetric	Asymmetric
One Key	Two Keys
Keys are usually 128 or 256-bits	Keys are much longer, 2048 or 4096-bits
Usually extremely fast	Computationally slower
Longer term communication	Key exchange, verification and authentication only
Based on circuits of permutation and substitution	Based entirely on mathematical principles

SOME MATHS ...

- Modular Arithmetic
- Exponentiation
- Logarithms
- Discrete Logarithms
- Primitive Roots
- Elliptic Curve Cryptography
- ...

Primitive Roots

- The number that is raised to a certain power, is called the generator *g*

g = 9

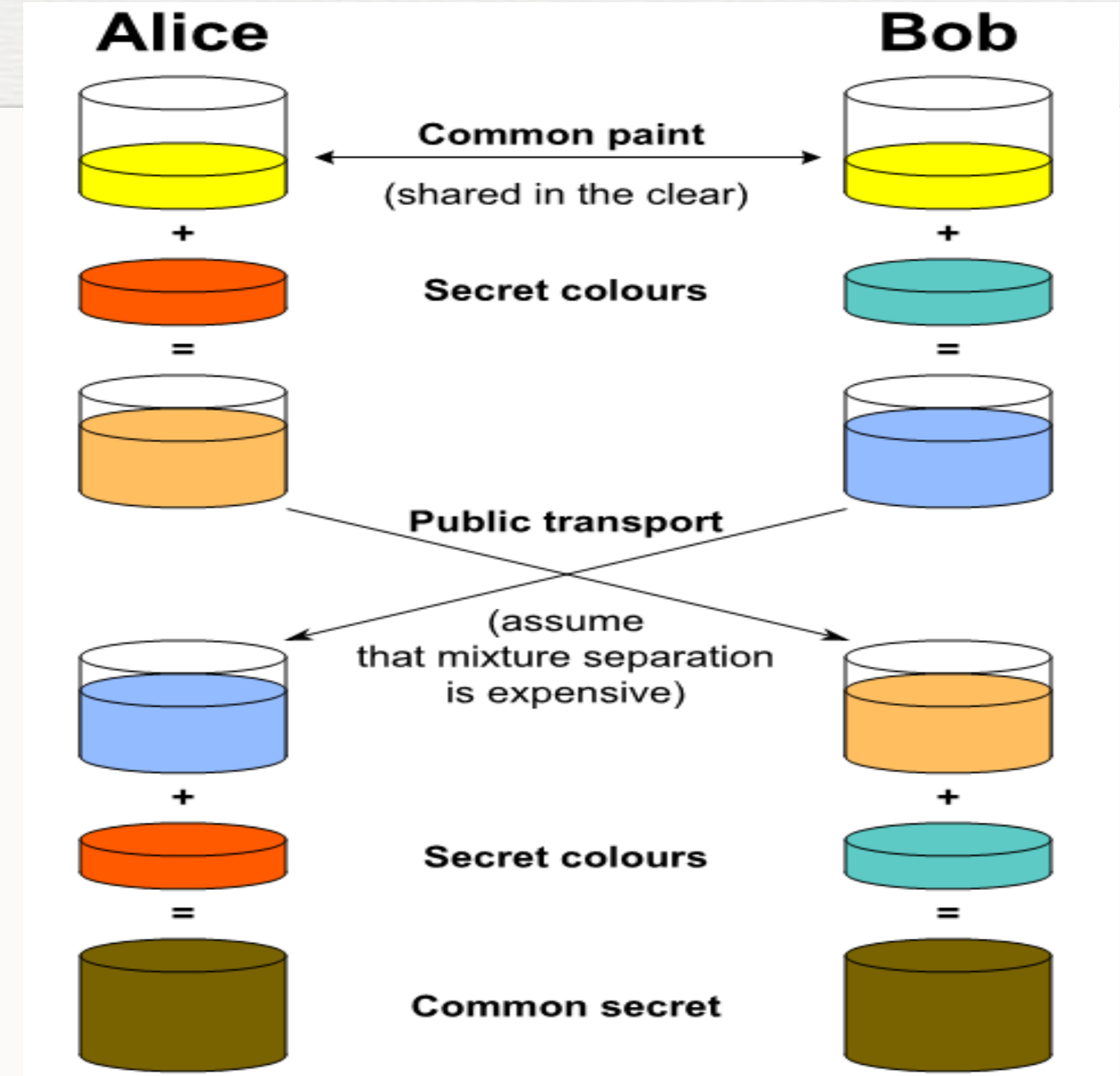
$$9^1 = 2 \pmod 7$$
$$9^2 = 4 \pmod 7$$
$$9^3 = 1 \pmod 7$$
$$9^4 = 2 \pmod 7$$
$$9^5 = 4 \pmod 7$$
$$9^6 = 1 \pmod 7$$

g = 3

$$3^1 = 3 \pmod 7$$
$$3^2 = 2 \pmod 7$$
$$3^3 = 6 \pmod 7$$
$$3^4 = 4 \pmod 7$$
$$3^5 = 5 \pmod 7$$
$$3^6 = 1 \pmod 7$$

primitive root

DIFFIE-HELLMAN KEY EXCHANGE



DIFFIE-HELLMAN KEY EXCHANGE

- Diffie-Hellman uses a public-key protocol to exchange a symmetric key in private
- Relies on the difficulty of finding discrete logs

DIFFIE-HELLMAN

1. Alice and Bob agree on a large prime p, and a generator g that is a primitive root of p
2. Alice chooses a private value a at random, then sends Bob a public $g^a \bmod p$
3. Bob chooses a private value b at random, then sends Alice a public $g^b \bmod p$
4. Alice computes $(g^b)^a \bmod p$, which is actually g^{ab}
5. Bob computes $(g^a)^b \bmod p$, which is also g^{ab}

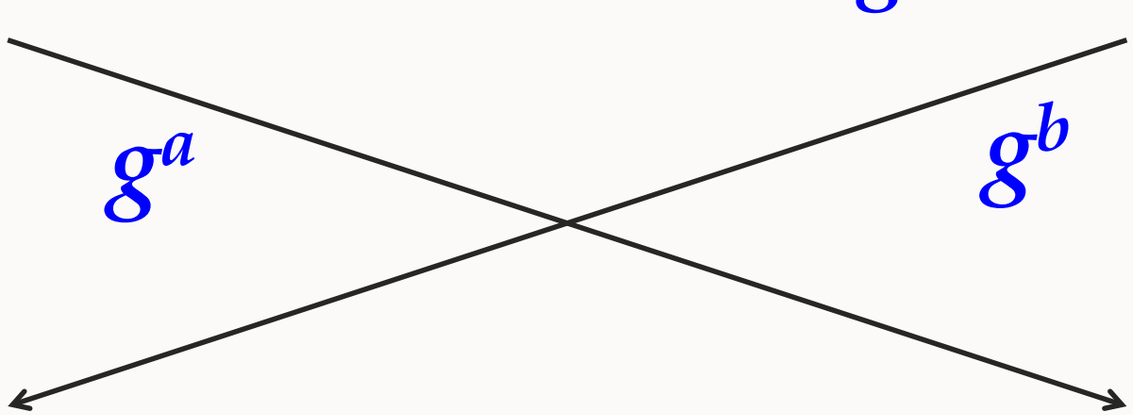
Public, Private

EXAMPLE

Alice and Bob agree on g = 3 and p = 29

Alice chooses a = 23, then $g^a = 3^{23} \bmod 29 = 8$

Bob chooses b = 12, then $g^b = 3^{12} \bmod 29 = 16$



Alice calculates:
$(g^b)^a \bmod 29 =$
$16^{23} \bmod 29 =$ 24

Bob calculates:
$(g^a)^b \bmod 29 =$
$8^{12} \bmod 29 =$ 24

Public, Private

WHY IS DH KEX SECURE?

- The secret shared key is g^{ab}
- Yet, only g, p, g^a and g^b have been transmitted and are public
- The only way to calculate g^{ab} is either $(g^a)^b$ or $(g^b)^a$
- The only way to find a or b is solve:

$$a = \mathrm{dlog}_{g,p}(g^a)$$

$$b = \mathrm{dlog}_{g,p}(g^b)$$

VULNERABILITIES

- Man-in-the-middle
  - A third party could intercept the initial communication from Alice, then create two separate key exchanges with both Alice and Bob
  - Simply re-encrypting each message would allow them to sit in the middle of the conversation
  - We can avoid this by combining DH with another cryptographic protocol

PERFECT FORWARD SECRECY

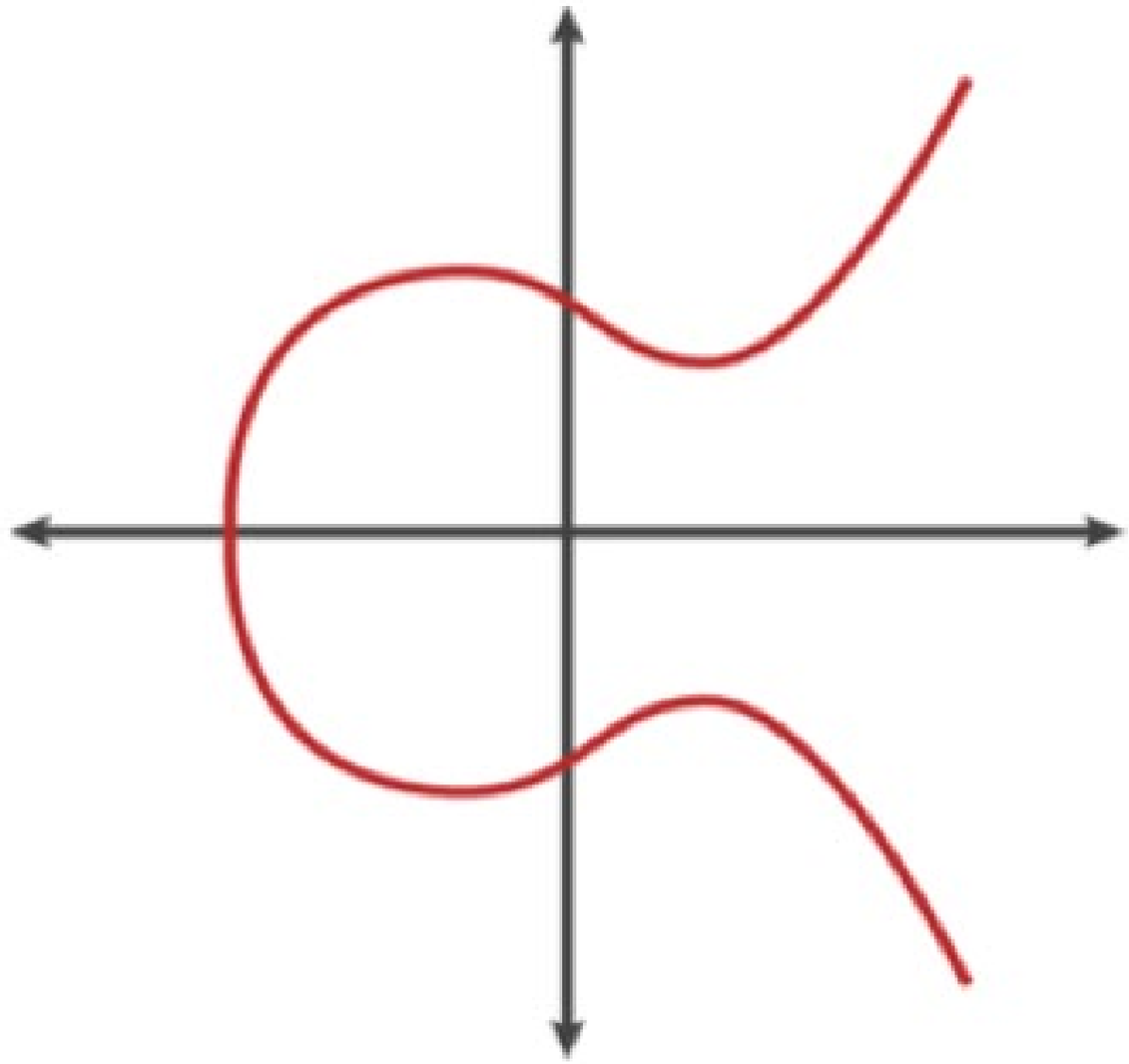
- There's always a chance a DHKEX key might be broken
- If we establish a symmetric key, how long should we use it for?
- Perfect forward secrecy means we generate new keys for each session, rather than persistent keys

EPHEMERAL MODE

- In protocols like TLS, running Diffie-Hellman in ephemeral mode forces a new key exchange every time
- The recommended settings for TLS are now 2048-bit DH keys, in ephemeral mode

ELLIPTIC CURVE CRYPTOGRAPHY

- Elliptic curves, of the form $y^2 = x^3 + ax + b$ can be used in place of mod arithmetic in DHKEX



ELLIPTIC CURVE CRYPTOGRAPHY

- Elliptic curve Diffie-Hellman is very similar to the regular version, but instead of g^a mod p we calculate aG mod p, so G + G + G … a times
- Given another point on the curve P, and the generator G, it is extremely hard to calculate *a* where

$$P = aG$$

- This is the elliptic curve discrete logarithm problem

WHAT DOES THIS MEAN?

Security Overview

This page is secure (valid HTTPS).

Valid Certificate

The connection to this site is using a valid, trusted server certificate.

View certificate

Secure Connection

The connection to this site is encrypted and authenticated using a strong protocol (TLS 1.2), a strong key exchange (ECDHE_RSA with P-256), and a strong cipher (AES_128_GCM).

Secure Resources

All resources on this page are served securely.

Elliptic Curve
Diffie-Hellman
Ephemeral

Advanced Encryption Standard
(Rjindael)
128-bit Key Size
Galois Counter Mode

SUMMARY

- Block Ciphers
  - DES -> AES
  - Modes of Operation
- Public Key Crypto
- (some) maths

Read:
- Gollman: Chapter 14
- Anderson: Chapter 5

COMP3052.SEC Computer Security

Session 15-5: Cryptology V

ACKNOWLEDGEMENTS

- Some of the materials we use this semester may come directly from previous teachers of this module, and other sources ...

- Thank you to (amongst others):
  - Michel Valstar, Milena Radenkovic, Mike Pound, Dave Towey…

TOPICS COVERED

- More maths … (still no need to panic)
- RSA
- Message Authentication Codes
- Digital Certificates

WHAT DOES THIS MEAN?

Security Overview

This page is secure (valid HTTPS).

Valid Certificate

The connection to this site is using a valid, trusted server certificate.

View certificate

Secure Connection

The connection to this site is encrypted and authenticated using a strong protocol (TLS 1.2), a strong key exchange (ECDHE_RSA with P-256), and a strong cipher (AES_128_GCM).

Secure Resources

All resources on this page are served securely.

Elliptic Curve
Diffie-Hellman
Ephemeral

Advanced Encryption Standard
(Rjindael)
128-bit Key Size
Galois Counter Mode

WHAT DOES THIS MEAN?

Security Overview

This page is secure (valid HTTPS).

Valid Certificate

The connection to this site is using a valid, trusted server certificate.

View certificate

Secure Connection

The connection to this site is encrypted and authenticated using a strong protocol (TLS 1.2), a strong key exchange (ECDHE_RSA with P-256), and a strong cipher (AES_128_GCM).

Secure Resources

All resources on this page are served securely.

Elliptic Curve
Diffie-Hellman
Ephemeral

Advanced Encryption Standard
(Rjindael)
128-bit Key Size
Galois Counter Mode

INTEGER FACTORISATION

- Any integer can be expressed as the multiplication of a list of prime numbers:

Example: 103284720

= 2 x 2 x 2 x 2 x 3 x 3 x 5
x 7 x 9 x 9 x 11 x 23

- The longer the value, the harder (and slower) this gets

INTEGER FACTORISATION

- Any integer can be expressed as the multiplication of a list of prime numbers:

Example: 103284720

= 2 x 2 x 2 x 2 x
3 x 3 x 3 x 3 x 3 x 3 x
5 x 7 x 11 x 23

- The longer the value, the harder (and slower) this gets

INTEGER FACTORISATION

- Semi-primes are the hardest numbers to factor:
  - Product of two primes,

Example: n = 1522605027922533360535618378132637429718068114961380688657908494580122963258952897654000350692006139

What are p and q?

INTEGER FACTORISATION

- Semi-primes are the hardest numbers to factor:
  - Product of two primes,

Example: n = 15226050279225333605356183781326374297180681149613806886579084945801229632589528976540003506920006139

What are p and q?

p = 379752279369436739228088727554456278545655366381 99
q = 400946909509208810306837352927614683892148997240 61

EULER TOTIENT FUNCTION

- Integers a and b are relatively prime if they do not share a divisor (except 1)
- The Euler totient Φ is the integers from 1 to n-1 that are relatively prime with n

number of

1. What is $\Phi(9)$?
2. What about $\Phi(11)$?

EULER TOTIENT FUNCTION

- The totient value of a prime p is simply p-1 (Euler)
- For two primes multiplied together it's (p-1)(q-1) (relatively)
- Euler's Theorem: *a* and *n* are relatively prime

$$a^{\Phi(n)} \equiv 1 \ (mod\ n)$$
$$a^{\Phi(n)+1} \equiv a \ (mod\ n)$$

RSA

- Developed by Rivest, Shamir and Adleman
  - Based on integer factorisation
  - Can provide both encryption and authentication
  - The most common PK algorithm in the world

RSA

1. Choose two large primes, *p and q*, then calculate $n = pq$
2. Select a value *e* that is relatively prime with the totient of *n*.
   - (Remember, we know $\Phi(n)$ as $(p\text{-}1)(q\text{-}1)$)

relatively

Example: $p = 17, q = 11$

$n = p \cdot q = 187$

$\Phi(n) = 160$

$e = one\ of\ 3, 6, 7, 11 \ldots = 7$

Public, Private

RSA

3. Calculate a multiplicative inverse to *e*: *d*, where:

$$e \equiv d^{-1} \left(mod\ \Phi(n)\right)$$

Or: $(e \cdot d)\ mod\ \Phi(n) = 1$

4. This is easily achieved if we know $\Phi(n)$, but not otherwise, we won't show the formula here

Example: $e = 7, d =$?

$(7 \cdot$? $)\ mod\ 160 = 1$

Public, Private

RSA

3. Calculate a multiplicative inverse to *e*: *d*, where:

$$e \equiv d^{-1} \left(mod\ \Phi(n)\right)$$

Or: $(e \cdot d)\ mod\ \Phi(n) = 1$

4. This is easily achieved if we know $\Phi(n)$, but not otherwise, we won't show the formula here

Example: $e = 7, d = 23$

$$(7 \cdot 23)\ mod\ 160 = 1$$

Public, Private

ENCRYPTION WITH RSA

- Now we have a public key *e,n* and a private key *d*
- Recall encryption is performed by:

$$M^e = C \ (mod\ n)$$

$$C^d = M \ (mod\ n)$$

Example: $M = 74$

$$C = 74^7 \ mod\ 187 = 167$$

$$M' = 167^{23} \ mod\ 187 = ?$$

Public, Private

ENCRYPTION WITH RSA

- Now we have a public key *e,n* and a private key *d*
- Recall encryption is performed by:

$$M^e = C \ (mod\ n)$$

$$C^d = M \ (mod\ n)$$

Example: $M = 74$

$$C = 74^7 \ mod\ 187 = 167$$

$$M' = 167^{23} \ mod\ 187 = \mathbf{74}$$

Public, Private

WHY IS RSA SECURE

- We'd like the message M based on some ciphertext C, given the public key e:

$$C = ?^e \ (mod\ n)$$

Equivalent to: $M = C^? \ (mod\ n)$

Remember: $(e \cdot d)\ mod\ \Phi(n) = 1$

- Calculating d can only be achieved by knowing the totient Φ of *n*. Finding this is extremely hard, for example we could factor *n* into *p* and *q*

USING RSA

- The keys (e, n) and (d) are reversible – either can be used for encryption, and the other used for decryption
- Everyone knows the public key, only the owner knows the private key
- This leads us to two very useful use cases for RSA:
  - Encryption only the owner can read
  - Signing that must have been performed by the owner

MESSAGE AUTHENTICATION CODES

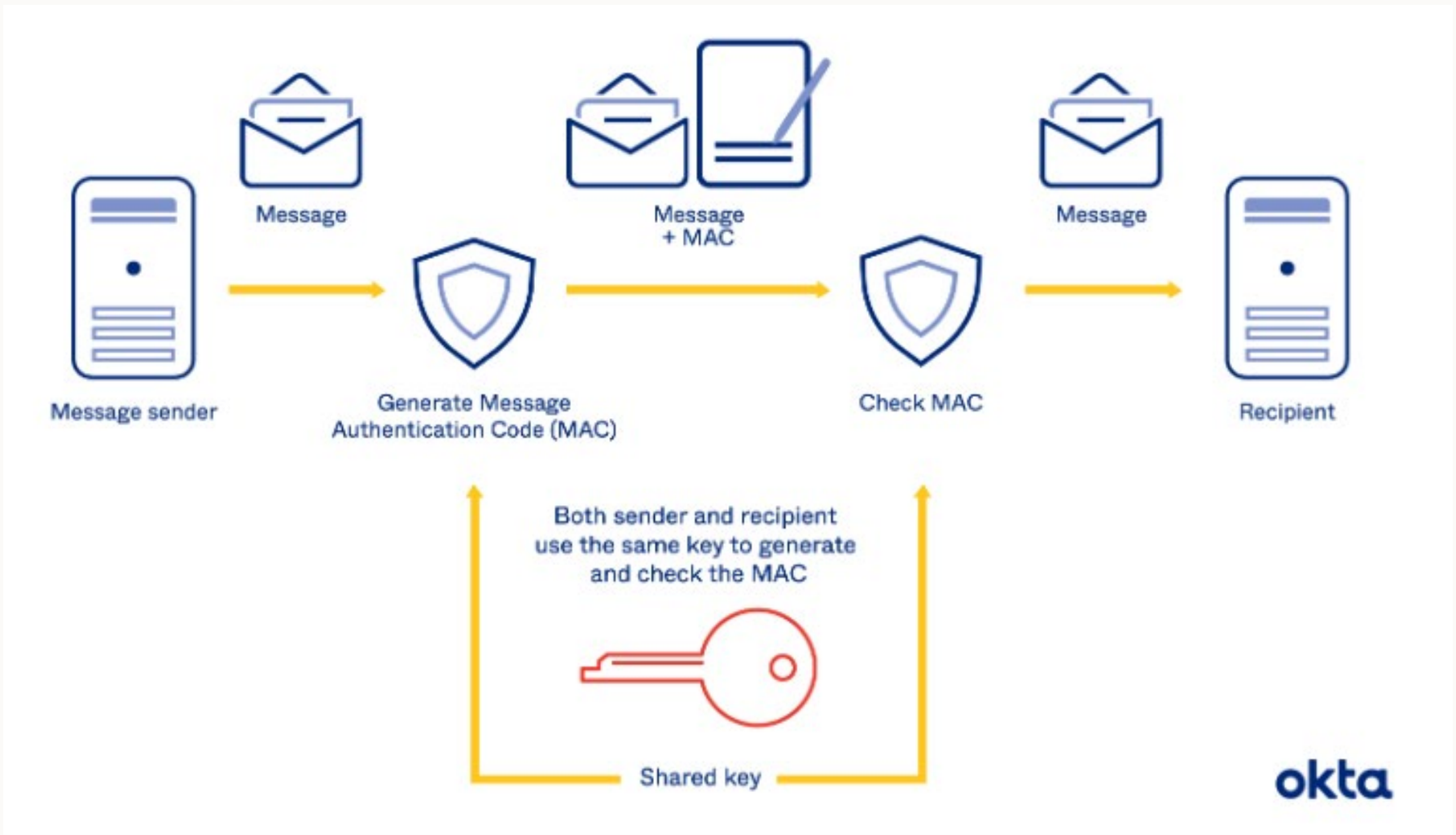
- Provide integrity and authenticity, not confidentiality
  - Protecting system files
  - Ensuring messages haven't been altered
- Calculate a hash of the message, then append this to the end of the message

KEYED HASHING

- Use a shared key to preserve message integrity
- Only those who know the key K can alter the message or hash

HMAC (HASH-BASED MAC)

- https://www.okta.com/identity-101/hmac/



HMAC ALGORITHMS

$$\mathrm{HMAC}(K, m) = \mathrm{H}\left(\left(K' \oplus opad\right) \parallel \mathrm{H}\left(\left(K' \oplus ipad\right) \parallel m\right)\right)$$

$$K' = \begin{cases} \mathrm{H}(K) & \text{if } K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

where

H is a cryptographic hash function.

m is the message to be authenticated.

K is the secret key.

K' is a block-sized key derived from the secret key, K; either by padding to the right with 0s up to the block size, or by hashing down to less than or equal to the block size first and then padding to the right with zeros.

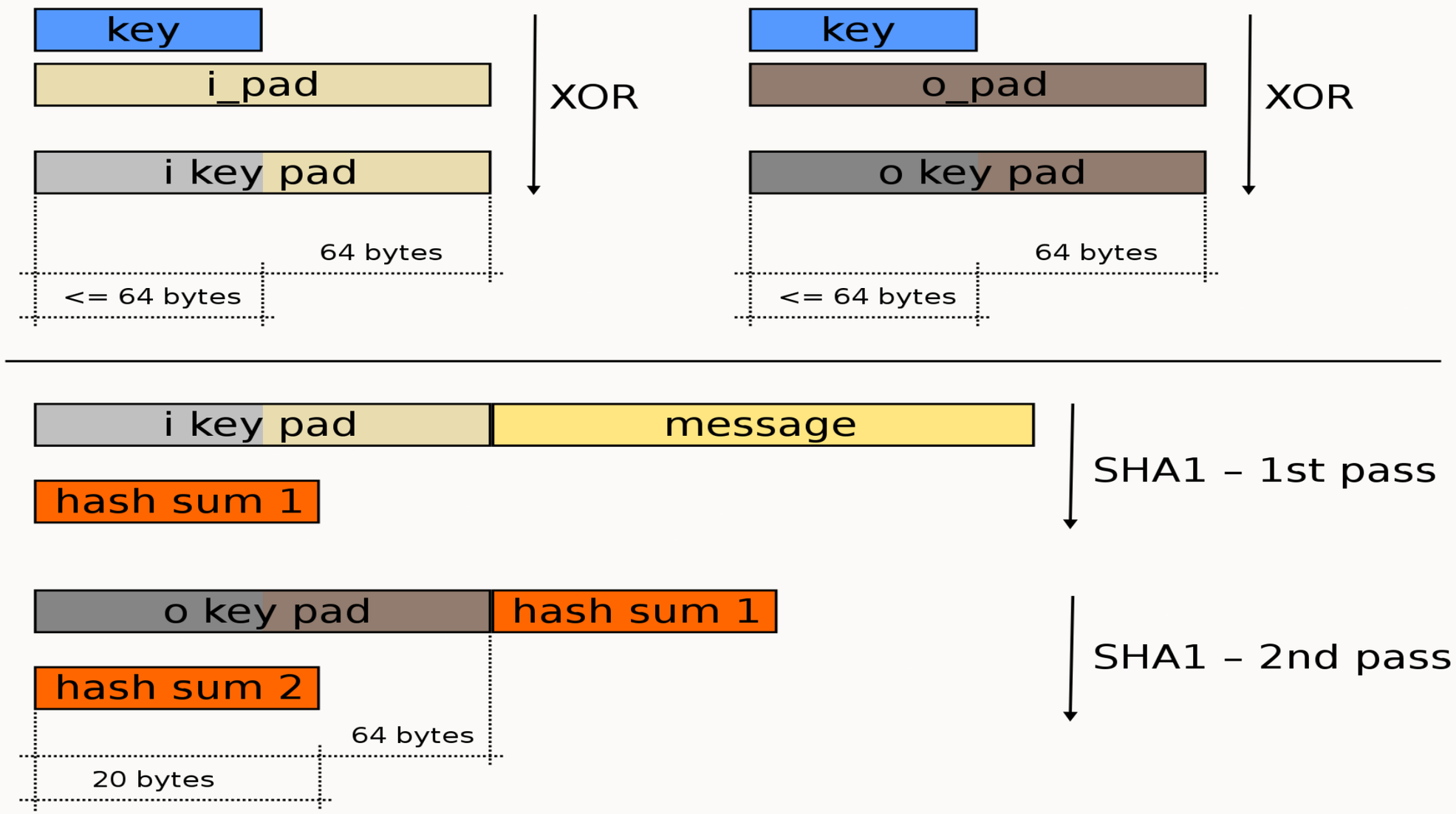
$\parallel$ denotes concatenation.

$\oplus$ denotes bitwise exclusive or (XOR).

$opad$ is the block-sized outer padding, consisting of repeated bytes valued 0x5c.

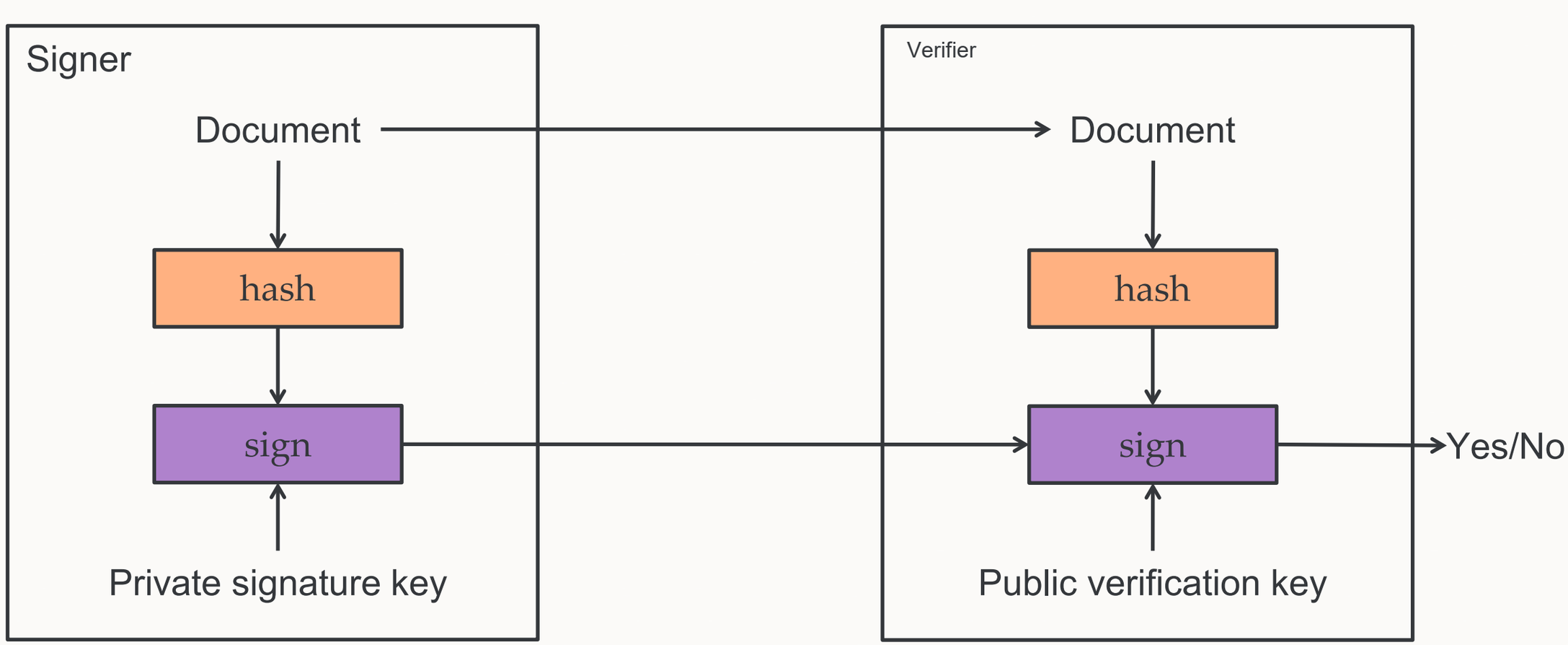
$ipad$ is the block-sized inner padding, consisting of repeated bytes valued 0x36.[3]

HMAC-SHA1


key
i_pad
XOR
i key pad
64 bytes
<= 64 bytes
key
o_pad
XOR
o key pad
64 bytes
<= 64 bytes
i key pad
message
hash sum 1
SHA1 – 1st pass
o key pad
hash sum 1
hash sum 2
SHA1 – 2nd pass
64 bytes
20 bytes

DIGITAL SIGNATURES

- Authentication codes provide integrity, but don't guarantee the sender
- Public-key encryption allows us to sign documents



DIGITAL SIGNATURES

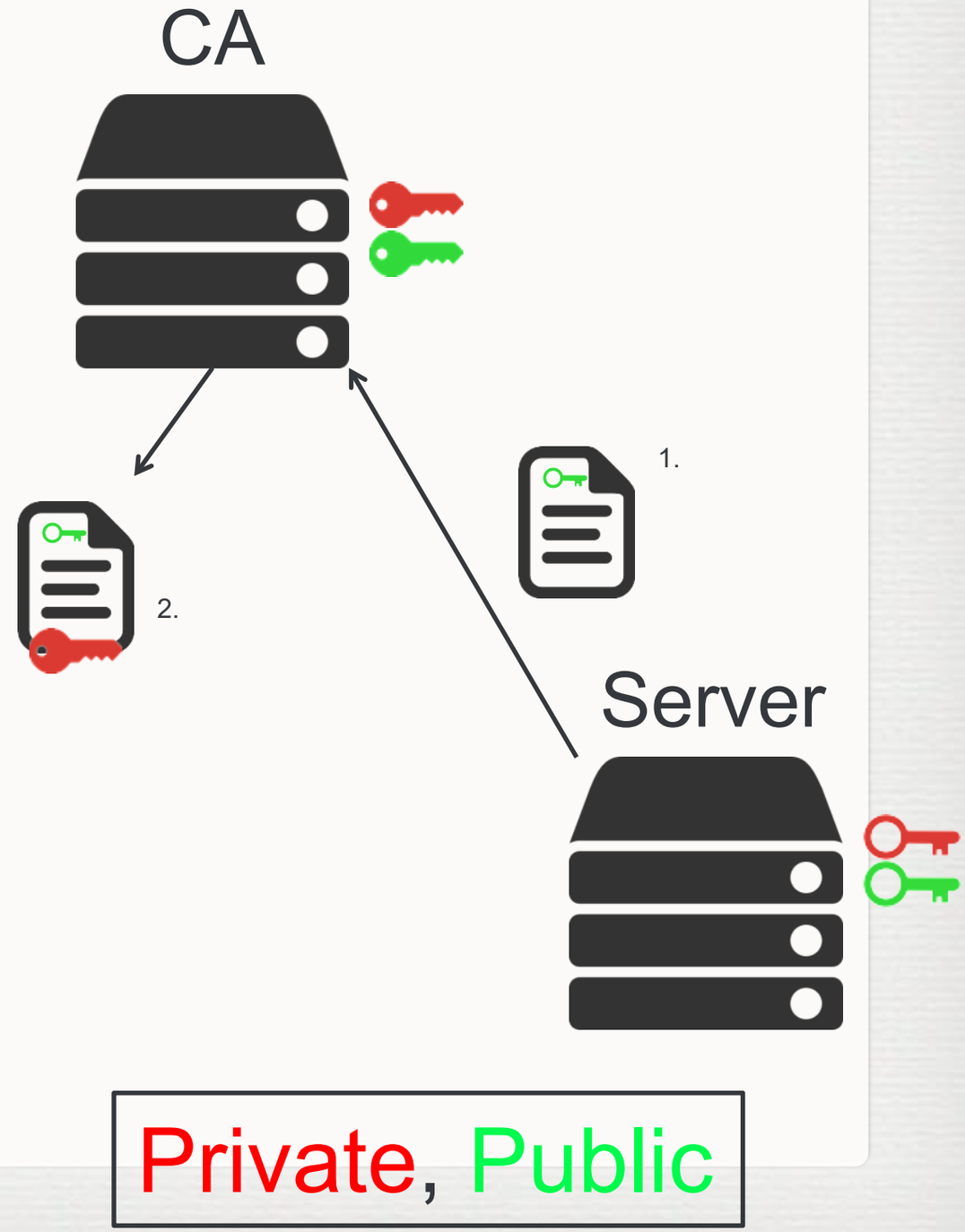
- Several digital signature algorithms in use, but majority of modern protocols use:
  - The Digital Signature Algorithm (**DSA**)
    - Based on large exponents in modulus arithmetic, much like Diffie-Hellman
  - **RSA** Signing
    - A variant of RSA, based on the problem of factoring large composite primes

DIGITAL CERTIFICATES

- We can use a trusted third party (TTP) in order to verify the ownership of a public key
- Bob then knows he has Alice's genuine key, not an imposter's
- Can also be 'self signed'
- An important part of Transport Layer Security (TLS)

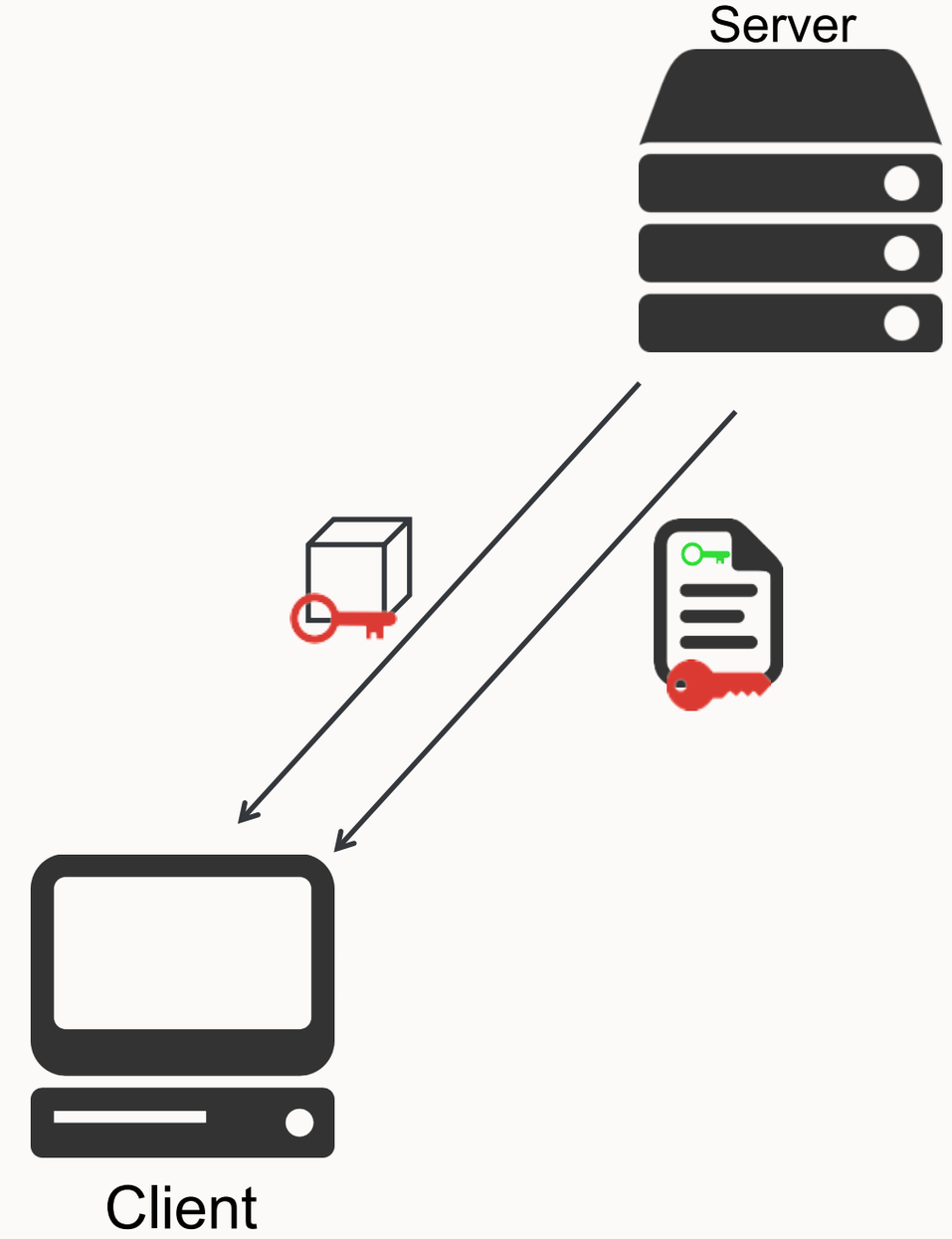
USING DIGITAL CERTIFICATES

- Server produces a certificate containing its public key, which wants to be trusted
- The certificate goes to a certificate authority (CA), who, after doing ID checks, signs the certificate with CA's private key



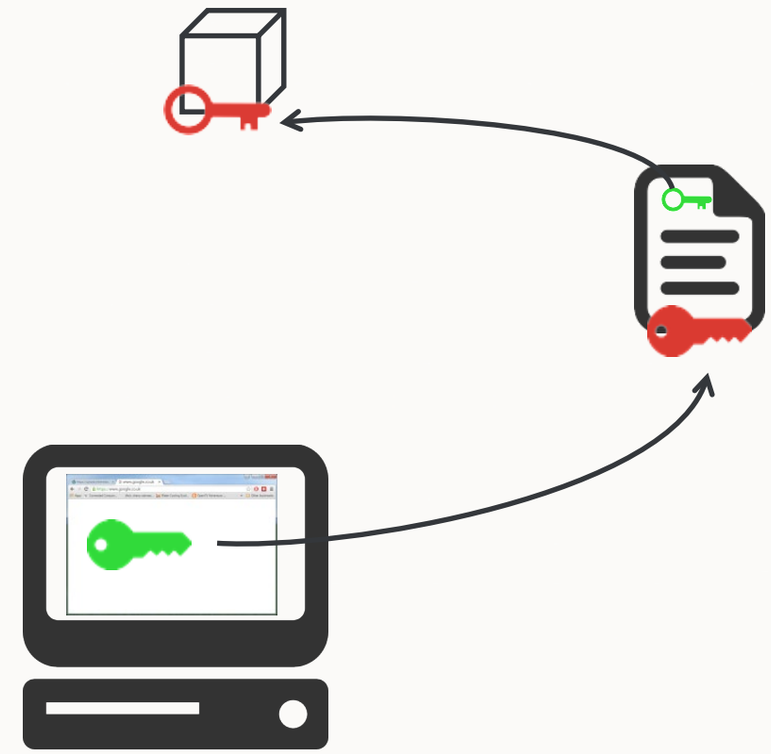
USING DIGITAL CERTIFICATES

- Client wants to use the server, who sends client something (e.g., DHKEX parameters) signed with its private key
- Server sends client a certificate, which has been signed by a CA for verification



USING DIGITAL CERTIFICATES

- Client has the CA public key stored in the browser. Client trusts the browser, so client trusts the CA. Client uses the CA public key to verify the certificate.
- When the certificate is verified, it is trusted, and the public key is used to verify the object



WHAT DOES THIS MEAN?

Security Overview

This page is secure (valid HTTPS).

Valid Certificate

The connection to this site is using a valid, trusted server certificate.

View certificate

Secure Connection

The connection to this site is encrypted and authenticated using a strong protocol (TLS 1.2), a strong key exchange (ECDHE_RSA with P-256), and a strong cipher (AES_128_GCM).

Secure Resources

All resources on this page are served securely.

Elliptic Curve
Diffie-Hellman
Ephemeral

RSA for message
authentication

Advanced Encryption Standard
(Rjindael)
128-bit Key Size
Galois Counter Mode

SUMMARY

- More maths
- RSA
- Message Authentication Codes
- Digital Certificates

Read:

- Gollman: Chapter 14
- Anderson: Chapter 5

CONCLUSIONS

- You should now have a working knowledge of:
  - The basic principles and terminology of cryptology
  - The mathematical foundations that cryptology is based on
  - The key factors to consider when choosing a security solution for a network

Read:
- Gollman: Chapter 14
- Anderson: Chapter 5