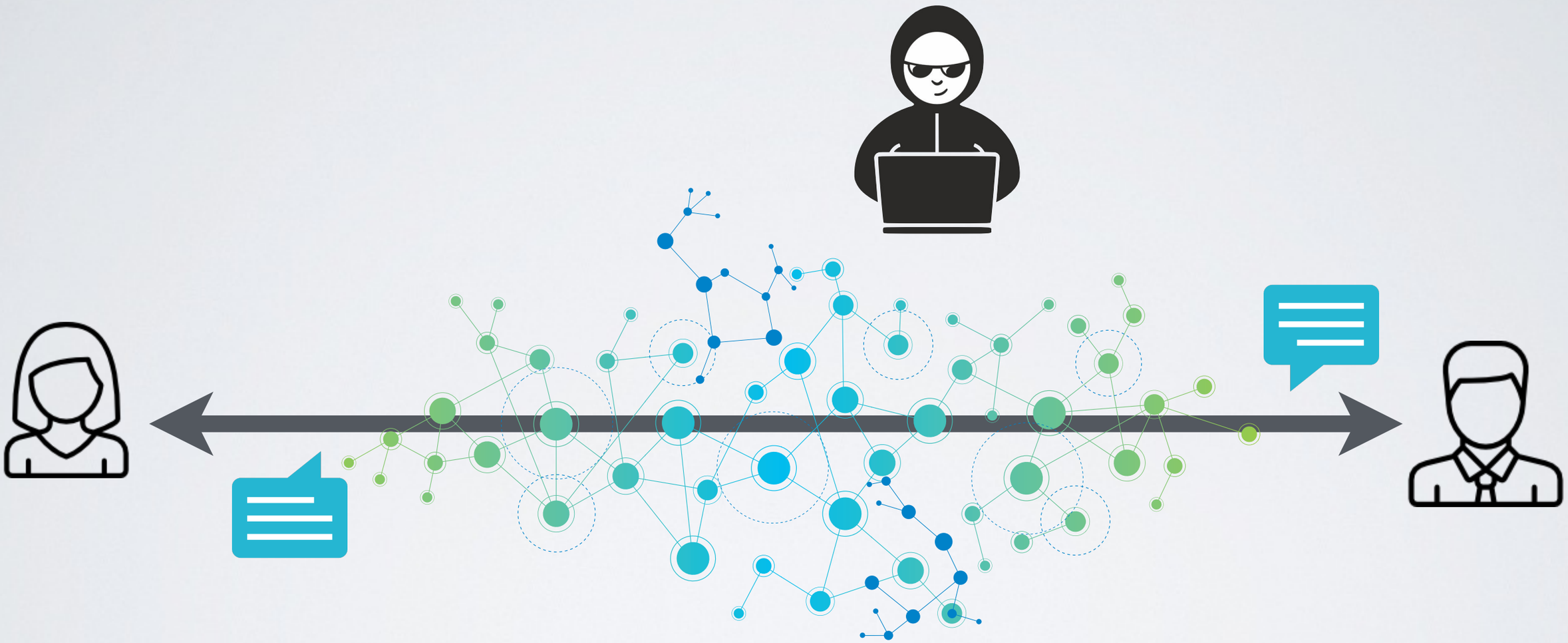
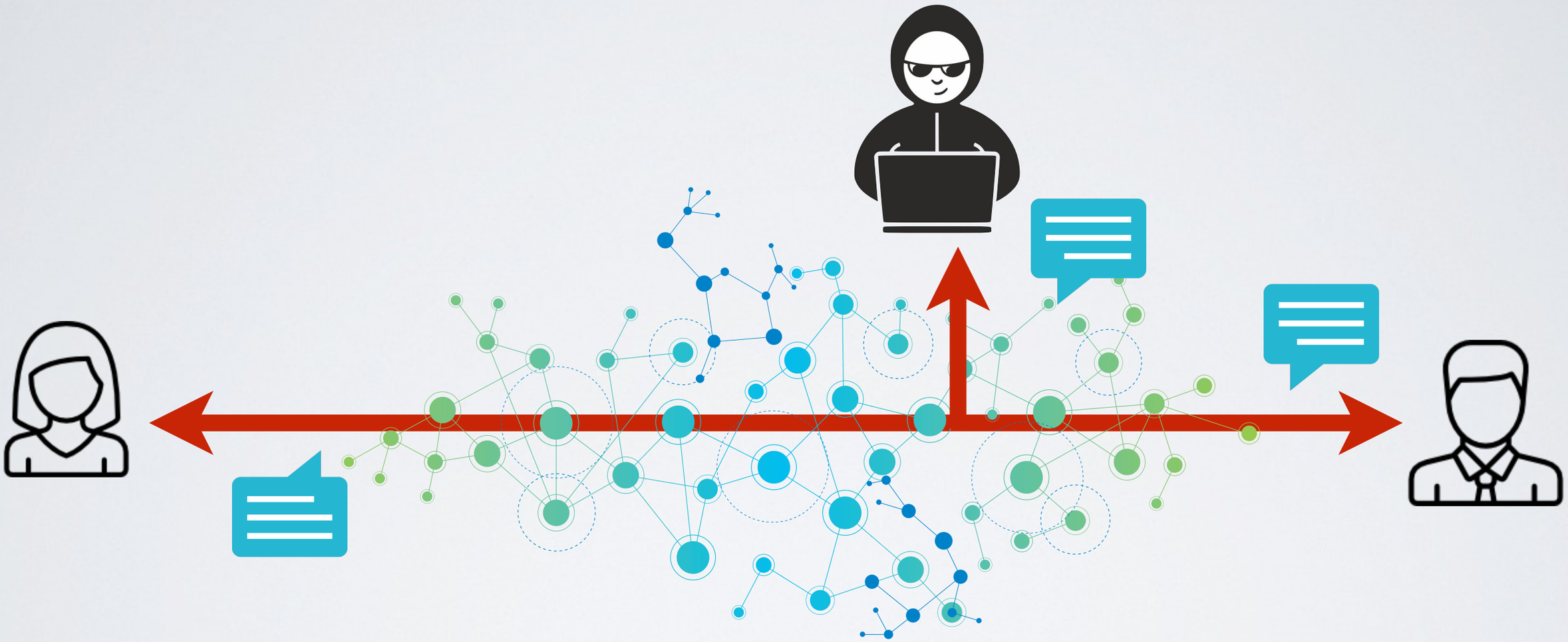
Introductory Cryptography

Kc Udonsi

Communication over an **insecure** medium

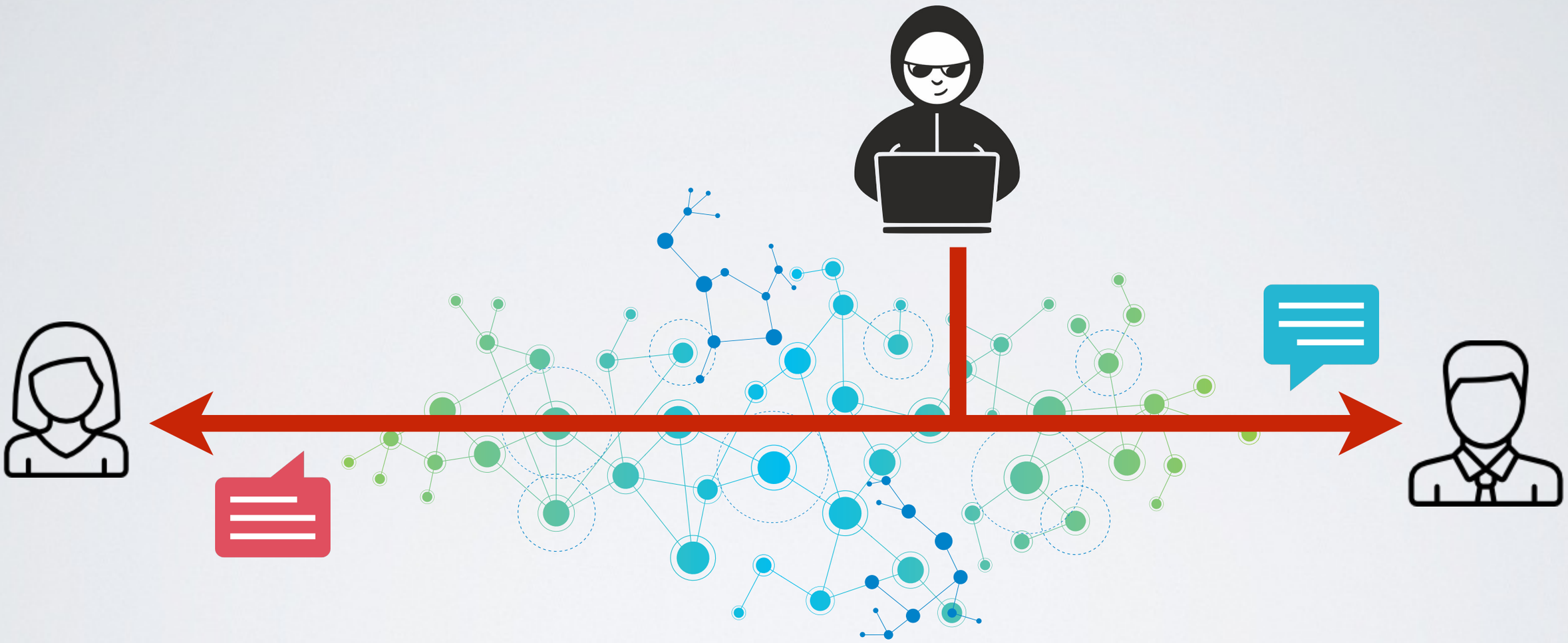


Threat 1 - **Interception**



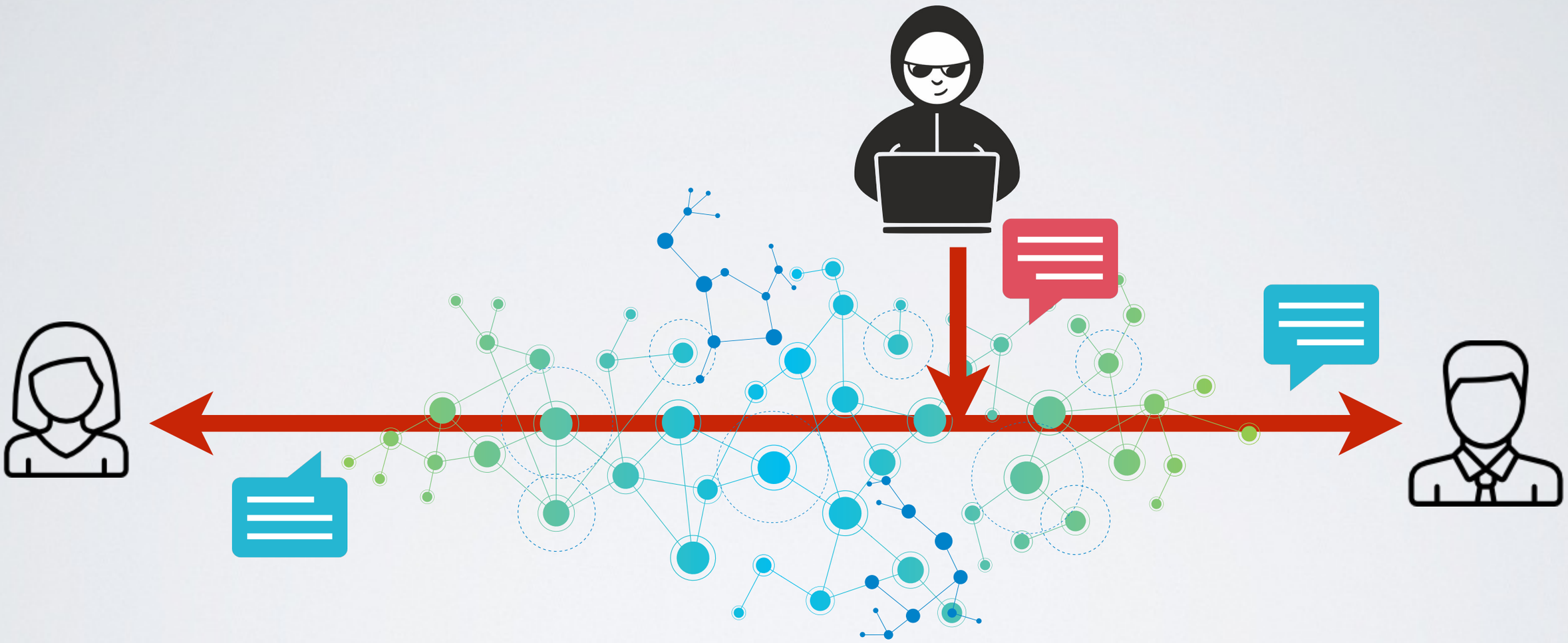
- **Interception** : an attacker can <u>read</u> messages

Threat 2 - **Modification**



- **Modification** : an attacker can <u>modify</u> messages

Threat 3 - **Fabrication**



- **Fabrication** : an attacker can inject messages

Threat 4 - **Interruption**

- **Interruption** : an attacker can block messages

Confidentiality and Integrity of communications

➡ Implement a **virtual trusted channel** over an insecure medium

Storage on an **insecure** medium

- Threat 1: **Loss**
  An attacker can corrupt or destroy data at rest
- Threat 2: **Disclosure**
  An attacker can disclose data at rest to other unauthorized parties
- Threat 3: **Theft**
  An attacker can obtain and store data at an arbitrary location
- Threat 4: **Modification**
  An attacker can compromise the integrity of data at rest

Storage on an **insecure** medium

- Threat 1: **Loss**
  Cryptography cannot be used to prevent loss. It may be used to yield loss. E.g ransomware
- Threat 2 & 3: **Disclosure & Theft**
  Encrypted data at rest cannot be meaningfully disclosed or utilized without decryption. E.g PGP Whole Disk Encryption
- Threat 4: **Modification**
  Cryptography can be used to verify the integrity of data at rest

Definitions of a cryptosystem

Definitions

Plaintext
The message in its clear form (the original message).

Ciphertext
The message in its ciphered form (the encrypted message).

Encryption
Transform a plaintext into ciphertext.

Decryption
Transform a ciphertext into a plaintext

Definitions

Cryptographic algorithm
The method to do encryption and decryption.

Cryptographic key
An input variable used by the algorithm for the transformation

N-bit security entropy (a.k.a. the key space)
The number of bits necessary to encode the number of possible keys (could be different than the key length)

Monoalphabetic cipher
A specific letter in the plaintext is consistently substituted with another letter in the cipher text

Definitions

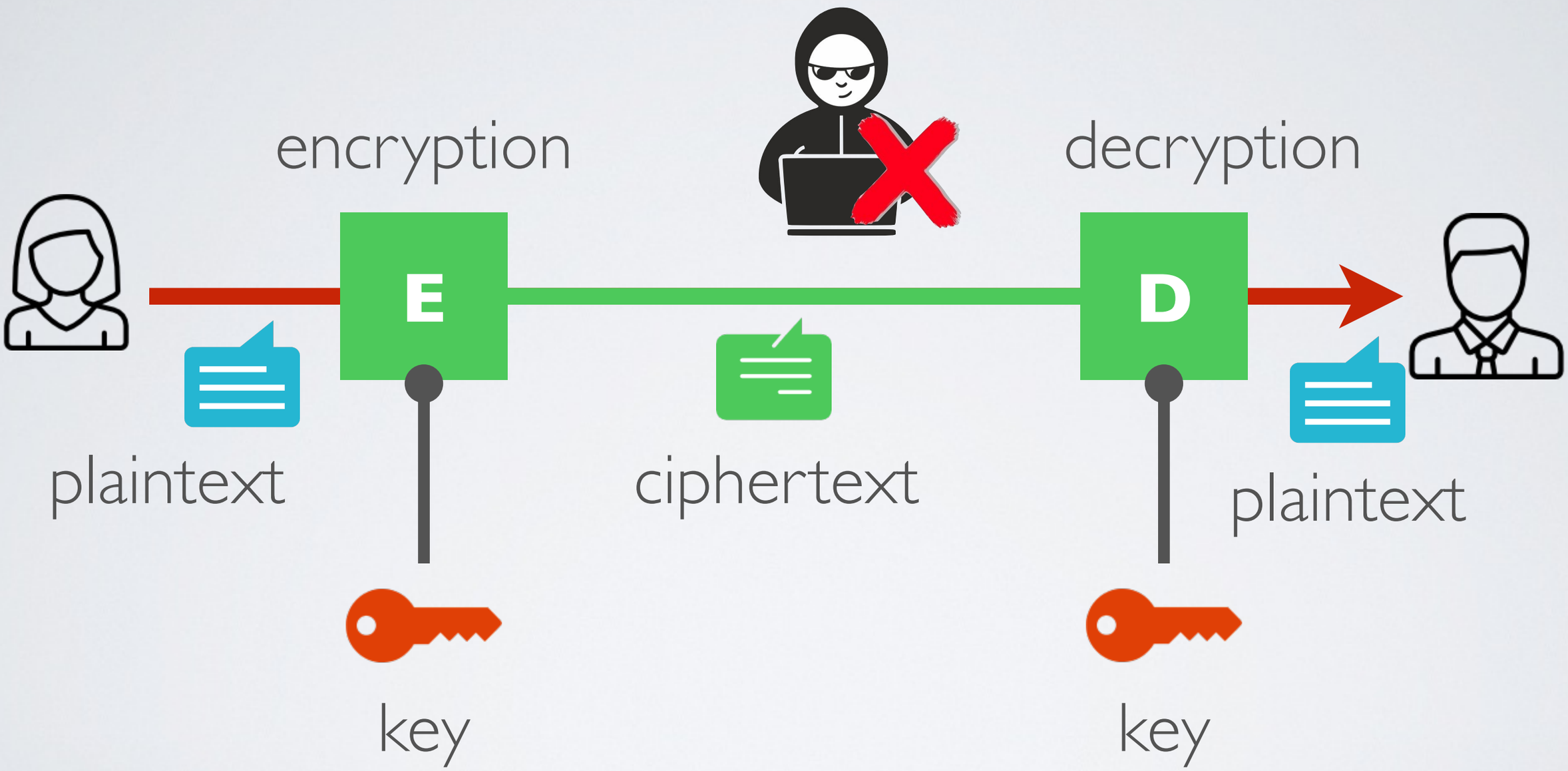
Polyalphabetic Cipher
A specific letter in the plaintext may be substituted with different letters in the cipher text

Cryptography
The art and science of securing messages

Cryptanalysis
The art and science of breaking secured messages to reveal the hidden message

Cryptography
The art and science of secure messaging which encompasses cryptography and cryptanalysis

The big picture



An early example…

Caesar Cipher - the oldest cryptosystem

A *shift* cipher – attributed to Julius Caesar (100-44 BC)

```
MEET ME AFTER THE TOGA PARTY
PHHW PH DIWHU WKH WRJD SDUWB
```

Shift the alphabet 23 places to the right and substitute letters

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

Representing data as numbers

Cryptographic algorithms are mathematical operations

- ➡ messages and keys must be represented as numbers
  for instance : ASCII encoding

Back to Caesar Cipher

Algorithm : shift the alphabet of a certain number of positions

Key : the number of positions to shift

Key space : 25 possible rotations (~ 5 bits security)

Encoding :

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Encrypting and decrypting one character is obtained as follows:

$$c = E(k,p) = (p + k) \bmod 26$$

$$p = D(k,c) = (c - k) \bmod 26$$

Cryptanalysis

Breaking the cipher

The Kerckhoffs' principle (1883)

"The enemy knows the system" - the security of a communication should not rely on the fact that the algorithms are secrets

➡ A cryptosystem should be secure even if everything about the system, except the key, is public knowledge

No security by obscurity

Breaking the cipher - the attacker's model

- **Exhaustive Search** (a.k.a brute force)
  Try all possible n keys (in average it takes n/2 tries)

- **Ciphertext only**
  You know one or several random ciphertexts

- **Known plaintext**
  You know one or several pairs of random plaintext and their corresponding ciphertexts

- **Chosen plaintext**
  You know one or several pairs of chosen plaintext and their corresponding ciphertexts

- **Chosen ciphertext**
  You know one or several pairs of plaintext and their corresponding chosen ciphertexts

➡ **A good crypto system resists all attacks**

Breaking Caesar cipher

Exhaustive search	Yes
ciphertext only	Statistical Analysis
known plaintext	Look at the first letter and get the shift
chosen plaintext	Choose “A” and get the shift
chosen ciphertext	Choose “A” and get the shift

Evolution of cryptosystems

A brief history of cryptography

~ 2000 years ago	Substitution ciphers (a.k.a mono alphabetic ciphers)
few centuries later	Transposition ciphers
Renaissance	Polyalphabetic ciphers
1844	Mechanization
1976	Public key cryptography

Substitution ciphers
(a.k.a mono alphabetic ciphers)

➡ Improvement over Caesar cipher

Algorithm : allow an arbitrary permutation of the alphabet

Key : set of substitutions

Key space : 26! possible substitutions ($4 \times 10^{26} \sim 89$ bits)

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

```
if we wish to replace letters
WI RF RWAJ UH YFTSDVF SFUUFYA
```

Breaking substitution ciphers

Exhaustive search	Doable with a computer
ciphertext only	Statistical analysis
known plaintext	Match letters together
chosen plaintext	Choose ABCDE … and match letters
chosen ciphertext	Choose ABCDE … and match letters

Polyalphabetic ciphers (a.k.a Renaissance Cipher)

➡ Vigenere cipher

Algorithm : combine the message and the key

Key : a word

Key space : the length of the word

$$\begin{array}{rl} & \texttt{wearediscoveredsaveyourself} \\ + & \texttt{deceptivedeceptivedeceptive} \pmod{26} \\ \hline & \texttt{ZICVTWQNGRZGVTWAVZHCQYGLMGJ} \end{array}$$

Advantage : Encryption of a letter is context dependent

Breaking Polyalphabetic Ciphers

exhaustive search	Small key length only
ciphertext only	Statistical analysis for small key length and significant amount of ciphertext
known plaintext	Subtract plaintext from ciphertext
chosen plaintext	Choose AAAAA … and match letters
chosen ciphertext	Choose AAAAA … and match letters

OTP - One Time Pad

➡ Improvement over Vigenere cipher

Algorithm : combine the message and the key

Key : an infinite random string

Key space : infinite

$$\begin{array}{rl} & \texttt{whatanicedaytoday} \\ \oplus & \texttt{yksuftgoarfwpfwel} \\ \hline & \texttt{ZZZJUCLUDTUNNWGQS} \end{array}$$

Advantage : this is the perfect cipher !

Disadvantage : hard to use in practice, how to transmit the key ?

XOR Cipher (a.k.a Vernham Cipher)

a modern version of Vigenere

Use ⊕ to combine the message and the key

$$E_k(m) = k \oplus m$$

$$D_k(c) = k \oplus c$$

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

Problem : known-plaintext attack

so $k = (k \oplus m) \oplus m$

$$x \oplus x = 0$$

$$x \oplus 0 = x$$

Mauborgne Cipher - a modern version of OTP

Use a random stream as encryption key

➡ Defeats the know-plaintext attack

Problem : Key-reused attack (a.k.a two-time pad)

$$C_1 = k \oplus m_1$$
$$C_2 = k \oplus m_2$$

so $C_1 \oplus C_2 = (k \oplus m_1) \oplus (k \oplus m_2)$

$$= (m_1 \oplus m_2) \oplus 0$$
$$= (m_1 \oplus m_2)$$

$$x \oplus x = 0$$
$$x \oplus 0 = x$$

The impossibility of breaking OTP

The ciphertext bears no statistical relationship to the plaintext

➡ No statistical analysis

For any plaintext and ciphertext, there exists a key mapping one to the other, and all keys are equally probable

➡ A ciphertext can be decrypted to any plaintext of the same length

The seeds of modern cryptography

1. **Diffusion**
   Mix-up symbols
   Transposition Cipher

2. **Confusion**
   Replace a symbol with another
   Polyaphabetic Cipher

3. **Randomization**
   Repeated encryption of the same text are different
   OTP

Types of Cryptographic Algorithms

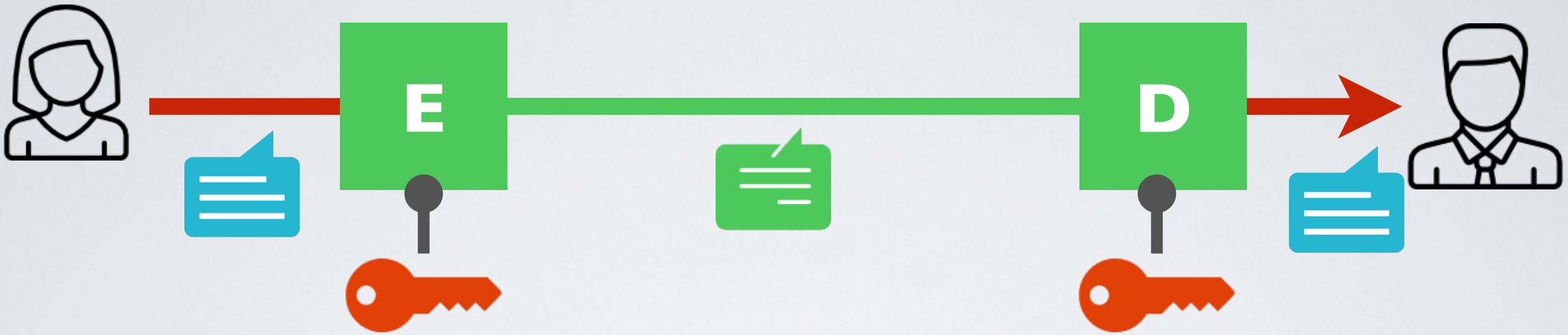
Definitions

One way algorithms
Also known as message digests. No keys involved. Encryption cannot be reversed.

Symmetric Key algorithms
The keys used for encryption and decryption are the same OR two-way mathematically related (can be derived from the other reliably)

Public Key algorithms
Also known as asymmetric algorithms. The keys used for encryption and decryption are different but one-way mathematically related.

Symmetric Key Encryption



➡ The same key $\mathbf{k}$ is used for encryption $\mathbf{E}$ and decryption $\mathbf{D}$

1. $\mathbf{D_k(E_k(m))=m}$ for every $\mathbf{k}$, $\mathbf{E_k}$ is an injection with inverse $\mathbf{D_k}$
2. $\mathbf{E_k(m)}$ is easy to compute (either polynomial or linear)
3. $\mathbf{D_k(c)}$ is easy to compute (either polynomial or linear)
4. $\mathbf{c = E_k(m)}$ finding $\mathbf{m}$ is hard without $\mathbf{k}$ (exponential)

Types of Symmetric Key Algorithms/Ciphers

Stream cipher

➡ Each bit is encrypted independently in a "stream"

RC4 - Rivest Cipher 4 (now deprecated)

Salsa20

Block cipher

➡ Blocks of data are encrypted in rounds

- Encryption standards

  DES (and 3DES) - Data Encryption Standard (now deprecated)

  AES - Advanced Encryption Standard

- Block cipher modes of operation

Random Number Generator

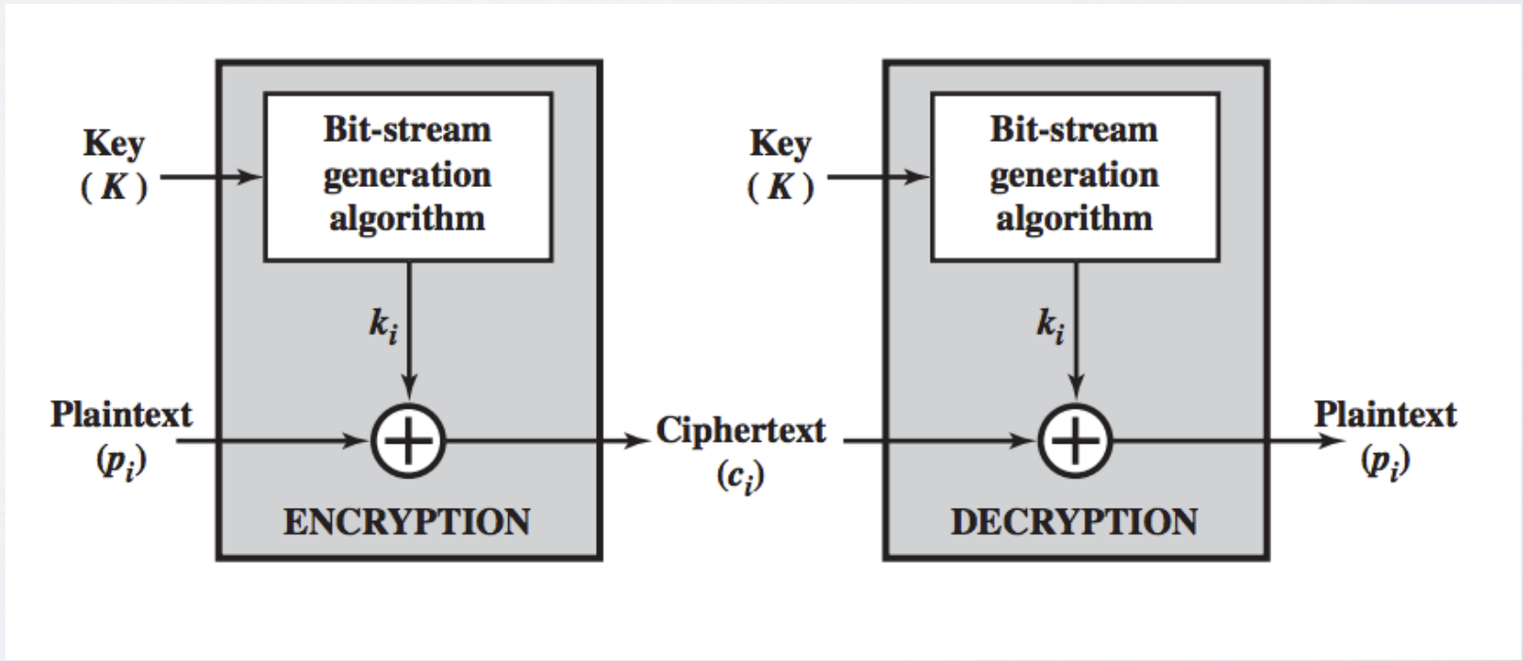
```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

True Random Number Generator

➡ No, because we want to be able to encrypt and decrypt

Pseudo-Random Generator

➡ Stretch a a fixed-size seed to obtain an unbounded random sequence



Stream cipher

Can we use $\mathbf{k}$ as a seed?

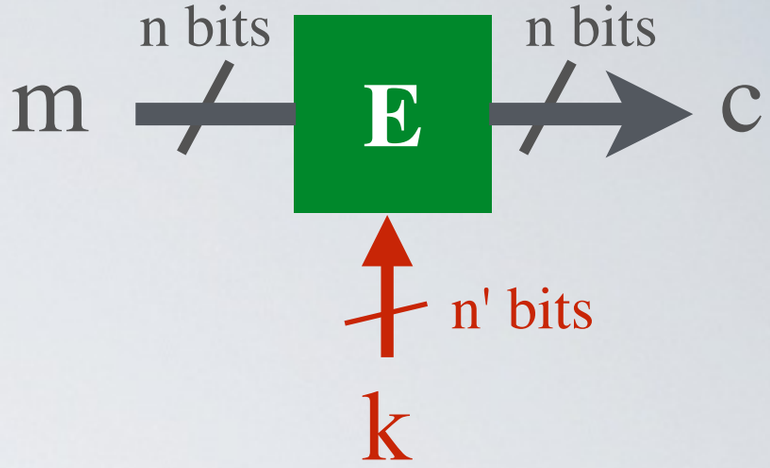
$$E_k(m) = m \oplus RNG(k)$$

➡ Be careful of key reused attack !

Break

Block Cipher

Ideal block cipher



- Combines confusion (substitution) and diffusion (permutation)
- Changing single bit in plaintext block or key results in changes to approximately half the ciphertext bits

➡ Completely obscure statistical properties of the original message

➡ A known-plaintext attack does not reveal the key

DES - Data Encryption Standard

Block size	64 bits
Key Size	56 bits
Speed	~ 50 cycles per byte
Algorithm	Feistel Network

Timeline

- **1972** NBS call for proposals
- **1974** IBM Lucifer proposal
  analyzed by DOD and enhanced by NSA
- **1976** adopted as standard
- **2004** NIST withdraws the standard

Security of DES - DES Challenges (brute force contests)

1998 *Deep Crack*, the EFF's DES cracking machine used 1,856 custom chips

- Speed : matter of days
- Cost : $250,000

2006 *COPACOBANA, the* COst-optimized Parallel COdeBreaker used 120 FPGAs

- Speed : less than 24h
- Cost : $10,000

3DES (Triple DES)

$$3DES_{k1,k2,k3}(m) = E_{k3}(D_{k2}(E_{k1}(m)))$$

- ➡ Uses three keys; some same or all distinct
- ➡ Effective key length (entropy) : 112 bits or 168 bits
- ✓ Very popular, used in PGP, TLS (SSL) …
- ◉ But terribly slow

AES - Advanced Encryption Standard

Timeline

- **1996** NIST issues public call for proposal
- **1998** 15 algorithms selected
- **2001** winners were announced

Rijndael by *J. Daemen and V. Rijmen*

Block size	128 bits
Key Size	128, 192, 256 bits
Speed	~18-20 cycles / byte
Mathematical Foundation	Galois Fields
Implementation	• Substitution-permutation network • Basic operations : ⊕, + , shift • Small code : 98k

Adopted by the NIST in December 2001

(pure) Encryption Modes
a.k.a. how to encrypt long messages

ECB - Electronic Code Book

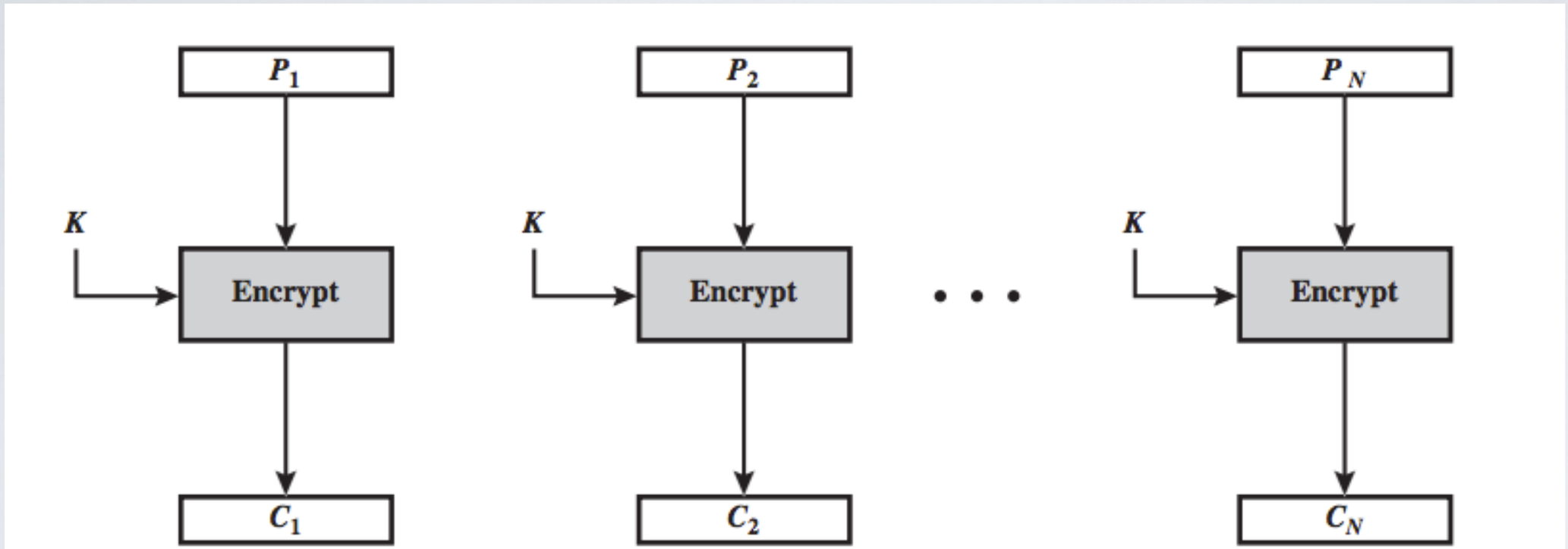
CBC - Cipher Block Chaining

CFB - Cipher Feedback

OFB - Output Feedback

CTR - Counter

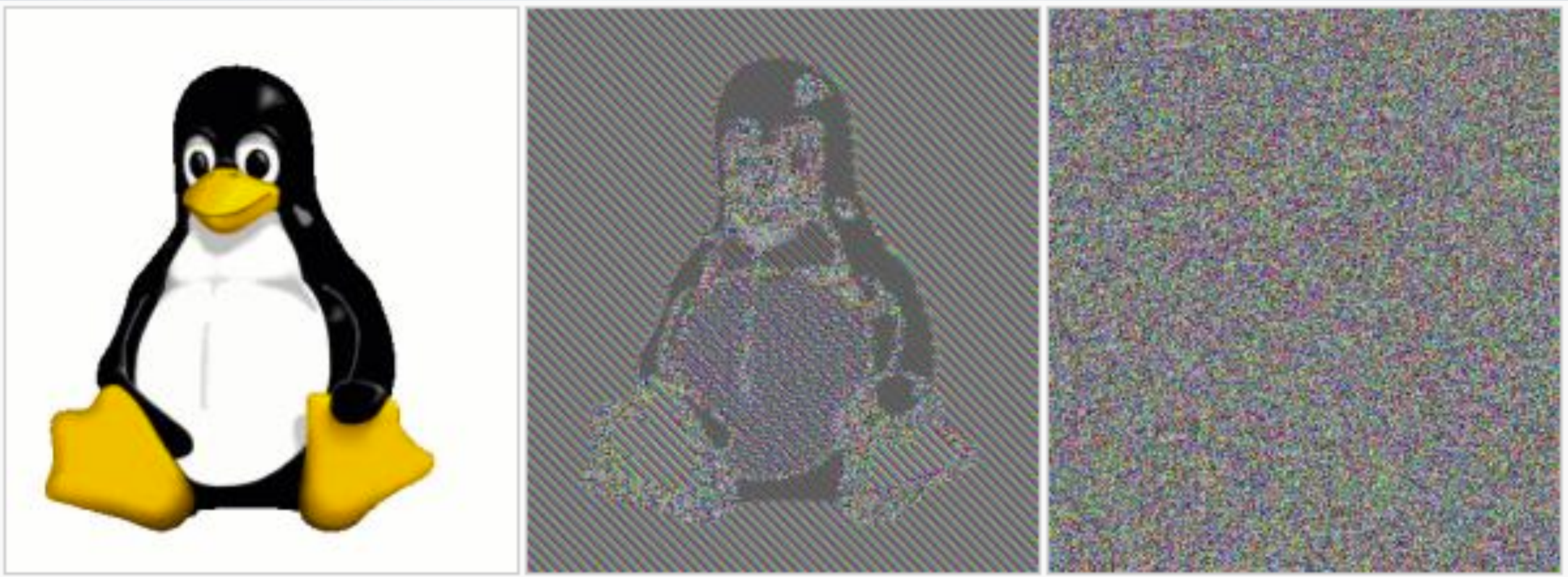
ECB - Electronic Code Book



Each plaintext block is encrypted <u>independently with the key</u>

- ✓ Block can be encrypted in parallel
- ◉ The same block is encrypted to the same ciphertext

How bad is ECB mode with a large data?

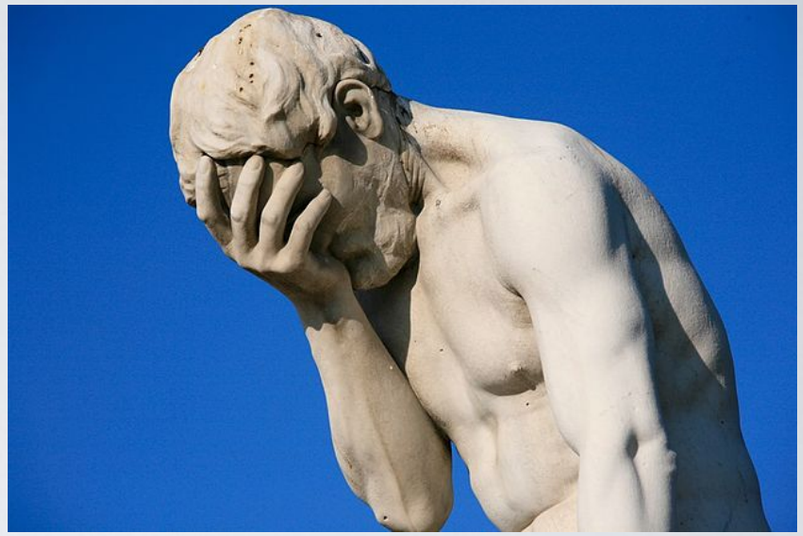


Original image

Encrypted using ECB mode

Modes other than ECB result in pseudo-randomness

source: *Wikimedia*



Simple Illustration of Zoom Encryption Failure

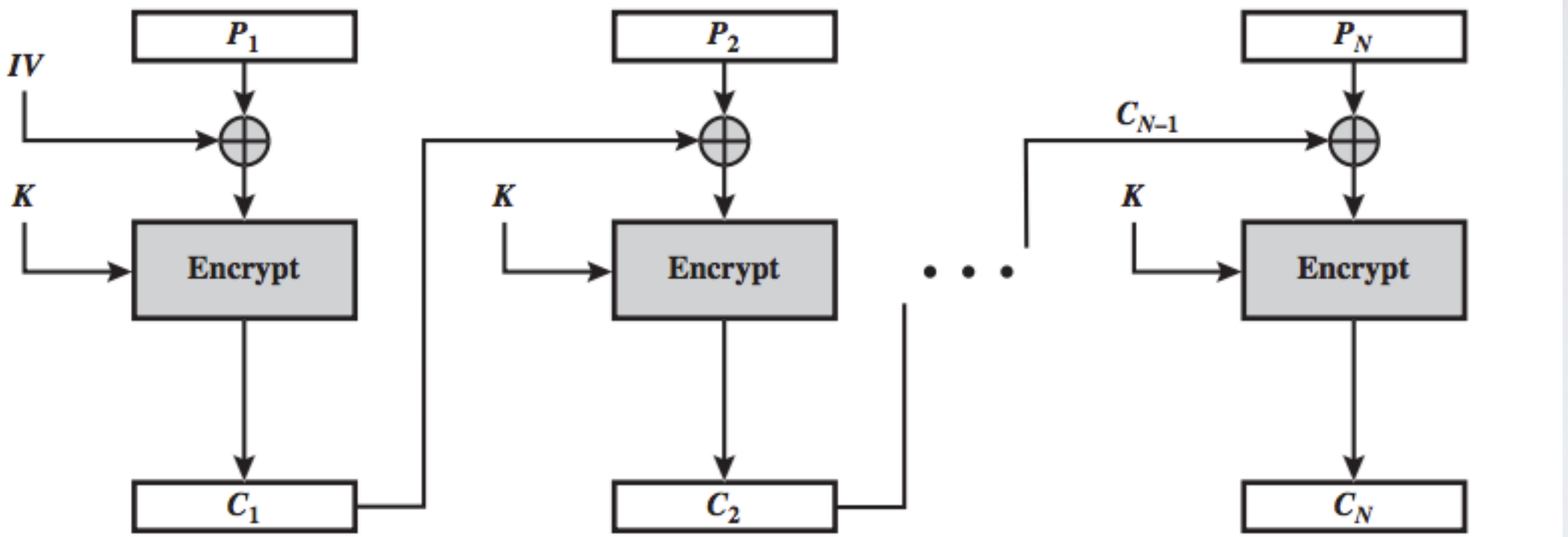
by Davi Ottenheimer on April 10, 2020

The Citizen Lab April 3rd, 2020 report broke the news on Zoom using weak encryption and gave this top-level finding:

> Zoom documentation claims that the app uses “AES-256” encryption for meetings where possible. However, we find that in each Zoom meeting, a single AES-128 key is used in ECB mode by all participants to encrypt and decrypt audio and video. The use of ECB mode is not recommended because patterns present in the plaintext are preserved during encryption.

source: *Security Boulevard*

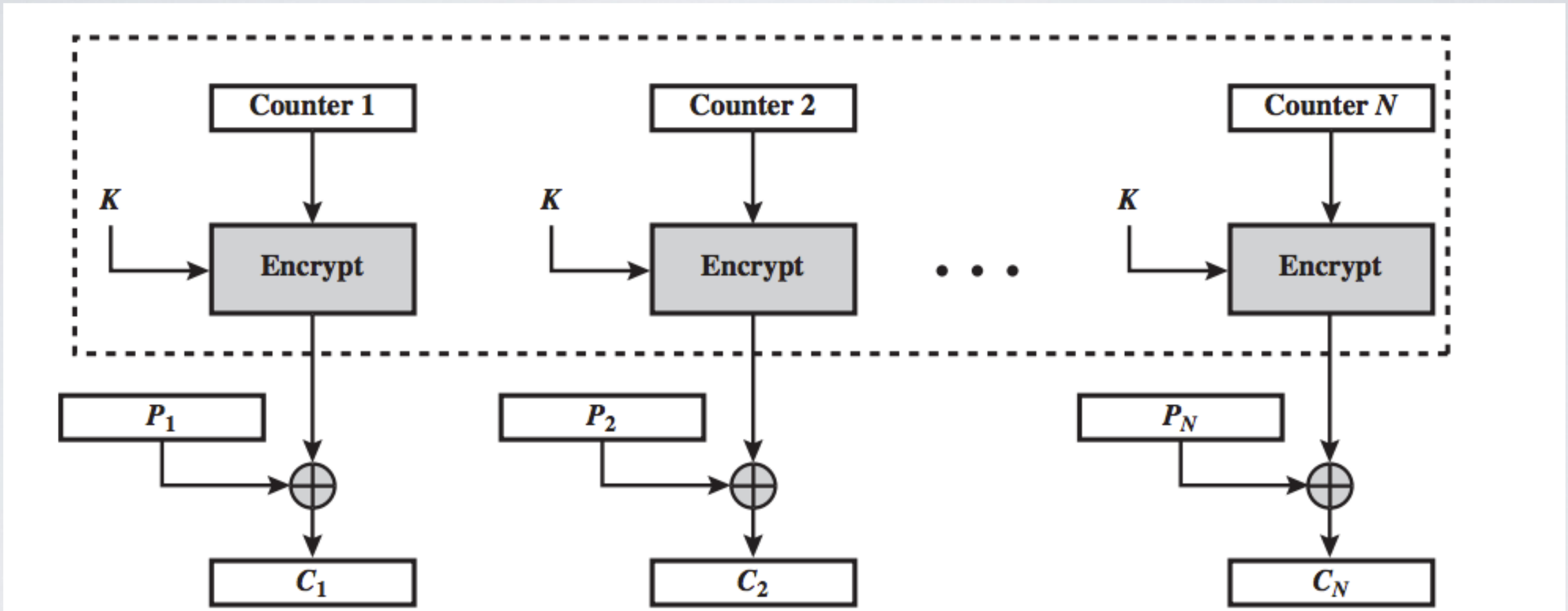
CBC - Cipher Block Chaining



Introduce some <u>randomness</u> using the previous ciphertext block

- ✓ Repeating plaintext blocks are not exposed in the ciphertext
- ◉ No parallelism
- ➡ The Initialization Vector should be known by the recipient

CTR - Counter

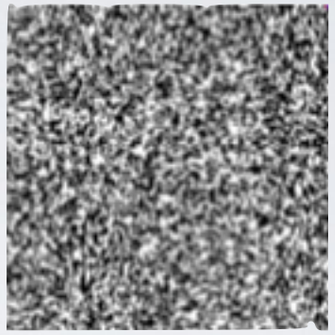


Introduce some randomness using a counter

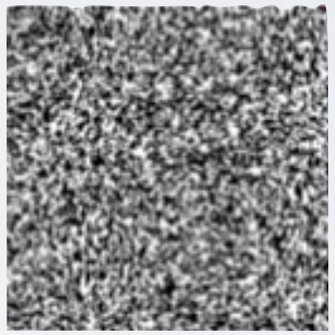
- ✓ High entropy and parallelism
- ◉ Sensitive to key-reused attack

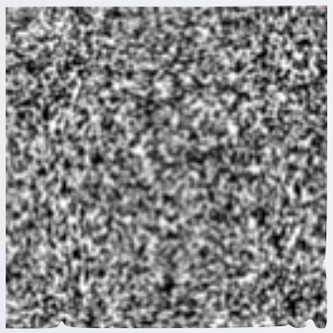
Key-reused attack on CTR

$\oplus$ K =

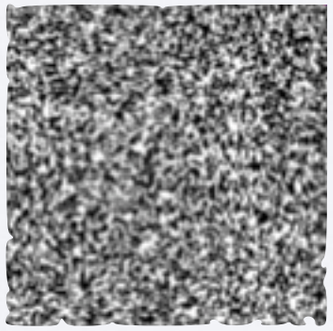


$\oplus$ K =





$\oplus$



=

Stream Cipher vs Block Cipher

	Stream Cipher	Block Cipher
Approach	Encrypt one symbol of plaintext directly into a symbol of ciphertext	Encrypt a group of plaintext symbols as one block
Pro	Fast	High diffusion
Cons	Low diffusion Key reused attack	Slow

Stream cipher and block cipher are often used together

- Stream cipher for encrypting large volume of data
- Block cipher for encrypting fresh pseudo-random seeds

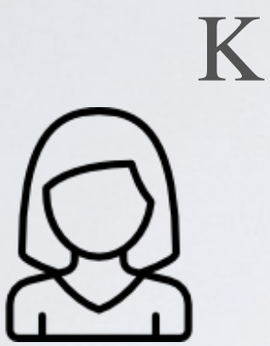
Latest trends

AES is now hardware accelerated (AES-NI native instruction)

➡ AES is fast enough (~1.3 cycles per byte)
to be used as the go-to cipher for any application

https://security.stackexchange.com/questions/22905/how-long-would-it-take-a-single-processor-with-the-aes-ni-instruction-set-to-bru

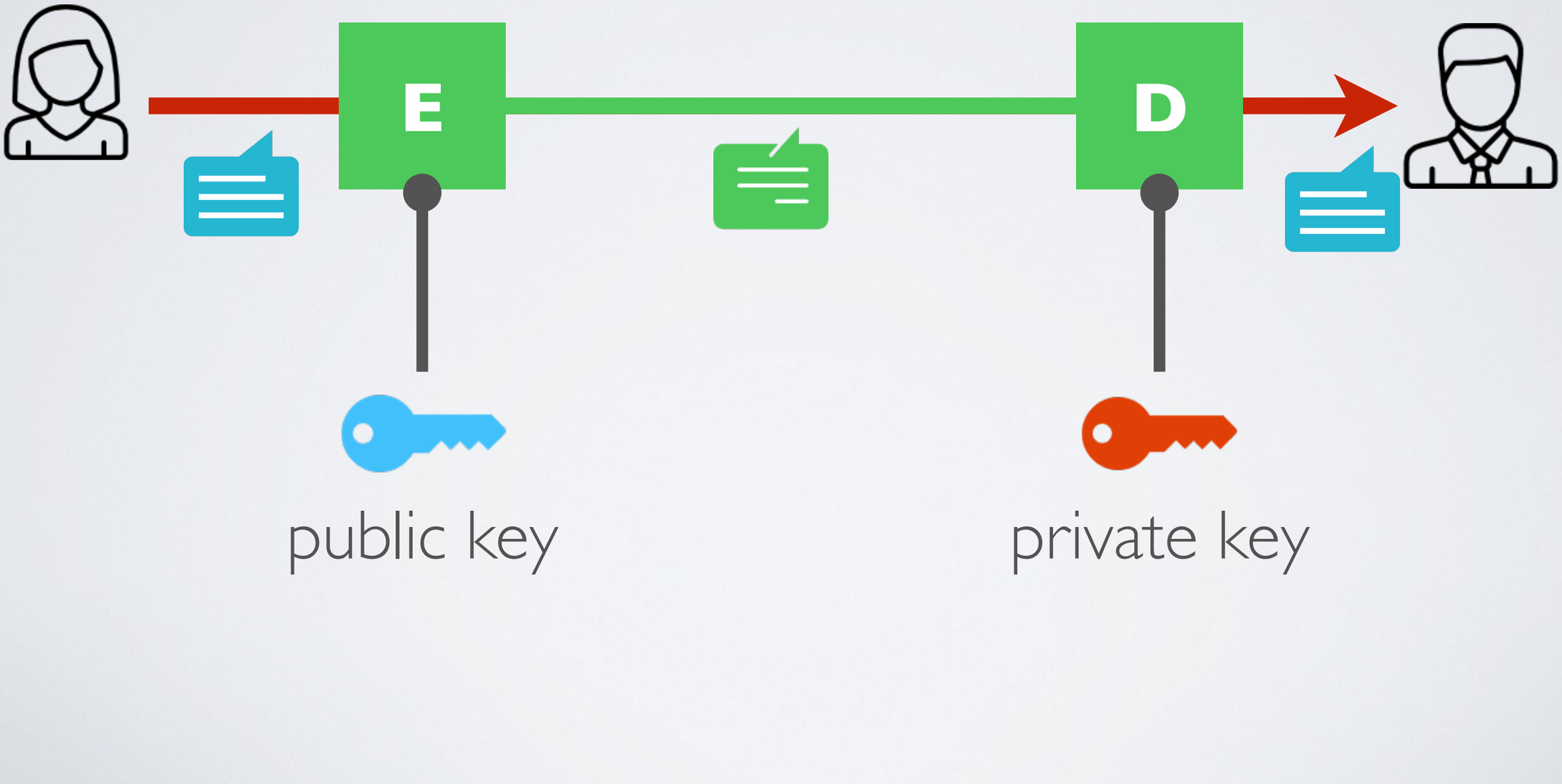
An issue ...



$E_k(m)$ = `tkS3bffBp...`

- How does Alice and Bob agree on a symmetric key?

Asymmetric encryption
a.k.a Public Key Cryptography

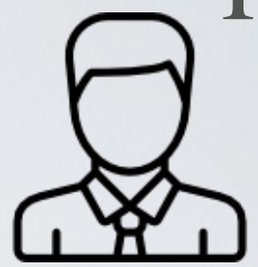


Asymmetric keys

$\mathbf{Ks_A}, \mathbf{Kp_A}$

$\mathbf{Kp_A}$

$\mathbf{Kp_A}$



Alice generates a pair of asymmetric keys

- $\mathbf{Ks_A}$ is the secret key that Alice keeps for herself
- $\mathbf{Kp_A}$ is the public key that Alice gives to everyone (even Mallory)

➡ These two keys $\mathbf{Ks_A}$ and $\mathbf{Kp_A}$ work together

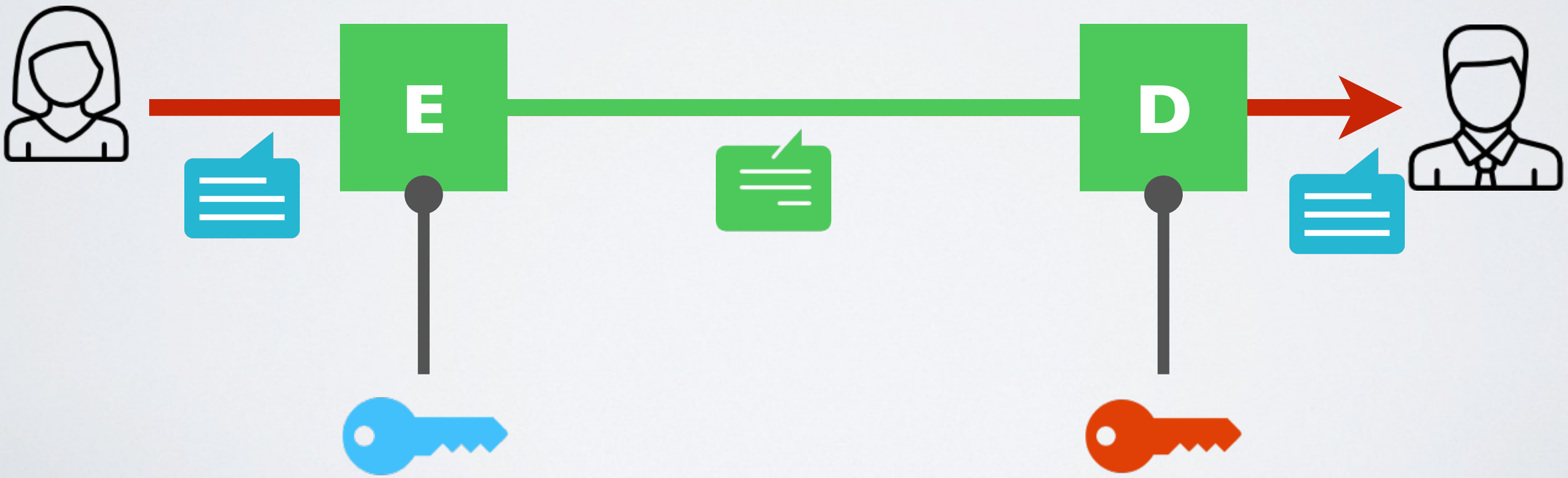
Asymmetric Keys - Functional Requirements

$D_{Ks}(E_{Kp}(m)) = m$ and $D_{Kp}(E_{Ks}(m)) = m$ for every pair (Kp, Ks)

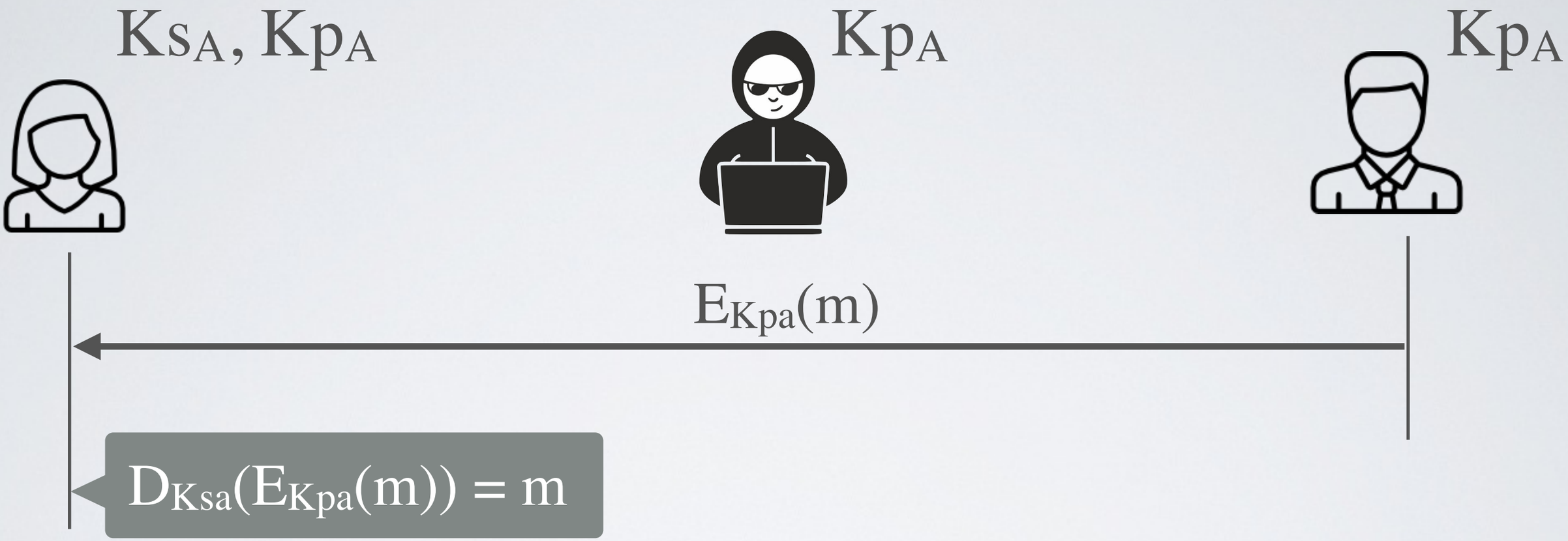
- ✓ Generating a pair (Kp, Ks) is easy to compute (polynomial)
- ✓ Encryption is easy to compute (either polynomial or linear)
- ✓ Decryption is easy to compute (either polynomial or linear)
- ◉ Finding a matching key Ks for a given Kp is hard (exponential)
- ◉ Decryption without knowing the corresponding key is hard (exponential)

Asymmetric encryption
a.k.a Public Key Cryptography

- ➡ The public key for encryption
- ➡ The private key for decryption



Asymmetric encryption for **confidentiality**



Bob encrypts a message $\mathbf{m}$ with Alice's public key $\mathbf{Kp_A}$

➡ Nobody can decrypt $\mathbf{m}$, except Alice with her private key $\mathbf{Ks_A}$

✓ Confidentiality without the need to exchange a secret key

RSA - Rivest, Shamir and Alderman

Key Size	1024 - 4096
Speed	~ factor of 10^6 cycles / operation
Mathematical Foundation	Prime number theory

Most widely used to secure network traffic

Adopted in 1977

Computational Complexity

Easy problems with prime numbers

- Generating a prime number p
- Addition, multiplication, exponentiation
- Inversion, solving linear equations

Hard problem with prime numbers

- Factoring primes
  e.g. given $\mathbf{n}$ find $\mathbf{p}$ and $\mathbf{q}$ such that $\mathbf{n = p \cdot q}$

RSA - generating the key pair

1. Pick $\mathbf{p}$ and $\mathbf{q}$ two large prime numbers and calculate $\mathbf{n = p \,.\, q}$ (see primality tests)
2. Compute $\mathbf{z = (p\text{-}1).(q\text{-}1)}$
3. Pick a prime number $\mathbf{e < z}$ such that $\mathbf{e}$ and $\mathbf{z}$ are relative primes

➡ (e,n) is the **public key**

4. Solve the linear equation $\mathbf{e * d = 1\ (\ mod\ z\ )}$ to find $\mathbf{d}$

➡ (d,n) is the **private key**
however p and q must be kept secret too

RSA - encryption and decryption

Given $\mathbf{Kp} = (\mathbf{e}, \mathbf{n})$ and $\mathbf{Ks} = (\mathbf{d}, \mathbf{n})$

➡ Encryption : $E_{kp}(m) = m^e \bmod n = c$

➡ Decryption : $D_{ks}(c) = c^d \bmod n = m$

➡ **$(m^e)^d \bmod n = (m^d)^e \bmod n = m$**

Other asymmetric cryptography schemes

Diffie-Hellman (precursor)

➡ No Authentication but good for key-exchange

El-Gamal

➡ Good properties for homomorphic encryption

Elliptic Curve Cryptography (widely used nowadays)

➡ Fast and small keys (190 bits equivalent to 1024 bits RSA)

Asymmetric vs Symmetric

	Symmetric	Asymmetric
pro	Fast	No key agreement
cons	Key agreement	Very slow

The best of both worlds

- Use RSA to encrypt a shared key
- Use AES to encrypt message

$$E_{Kp}(m) = RSA_{Kp}(k), AES_k(m)$$

Naive approach

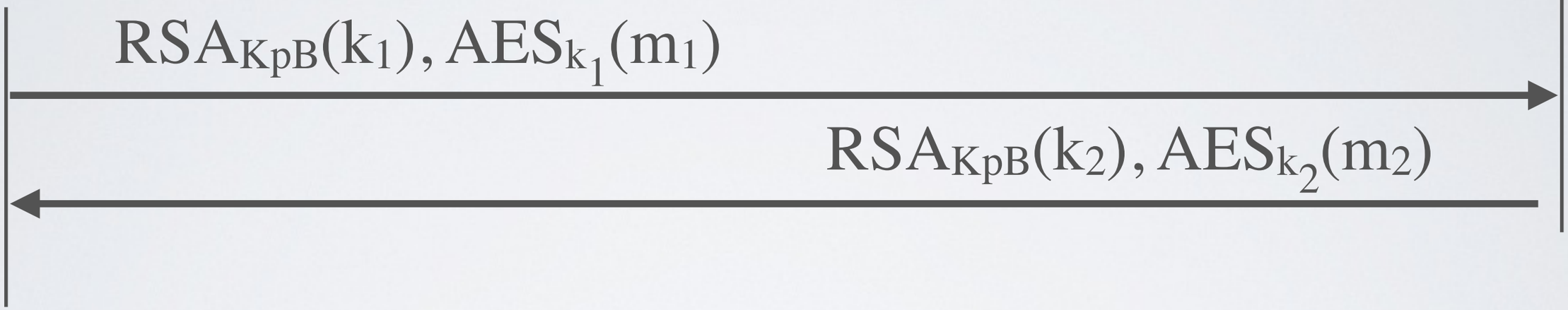
But not perfect yet

Ks_A, Kp_A

Kp_A, Kp_B

Ks_B, Kp_B

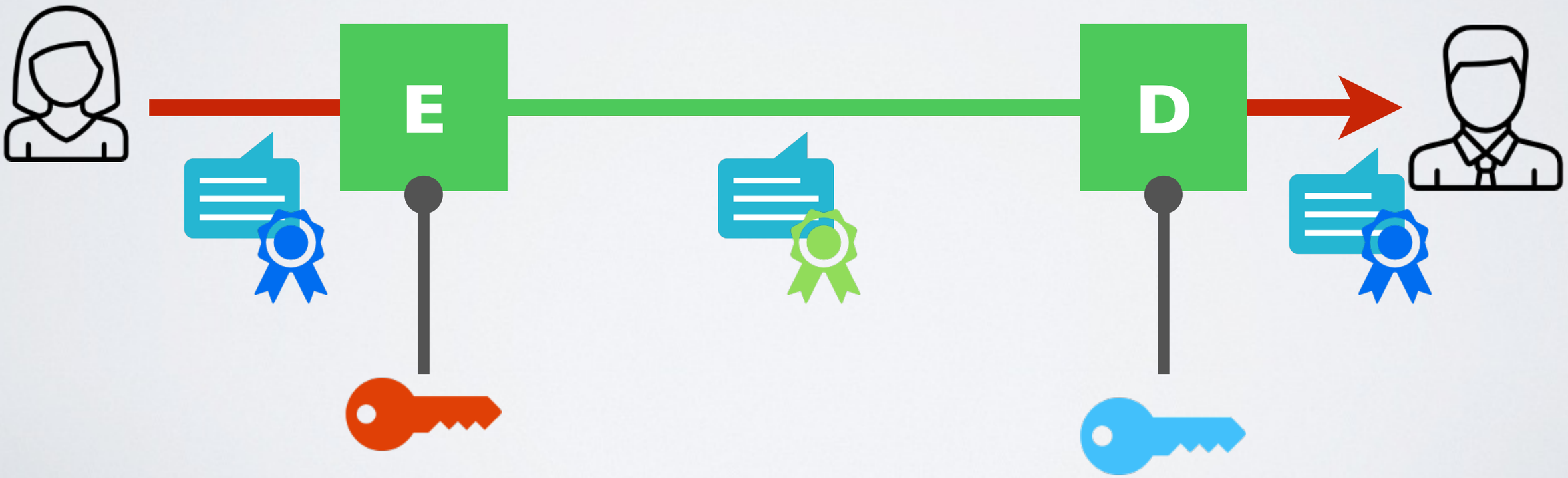
Kp_A



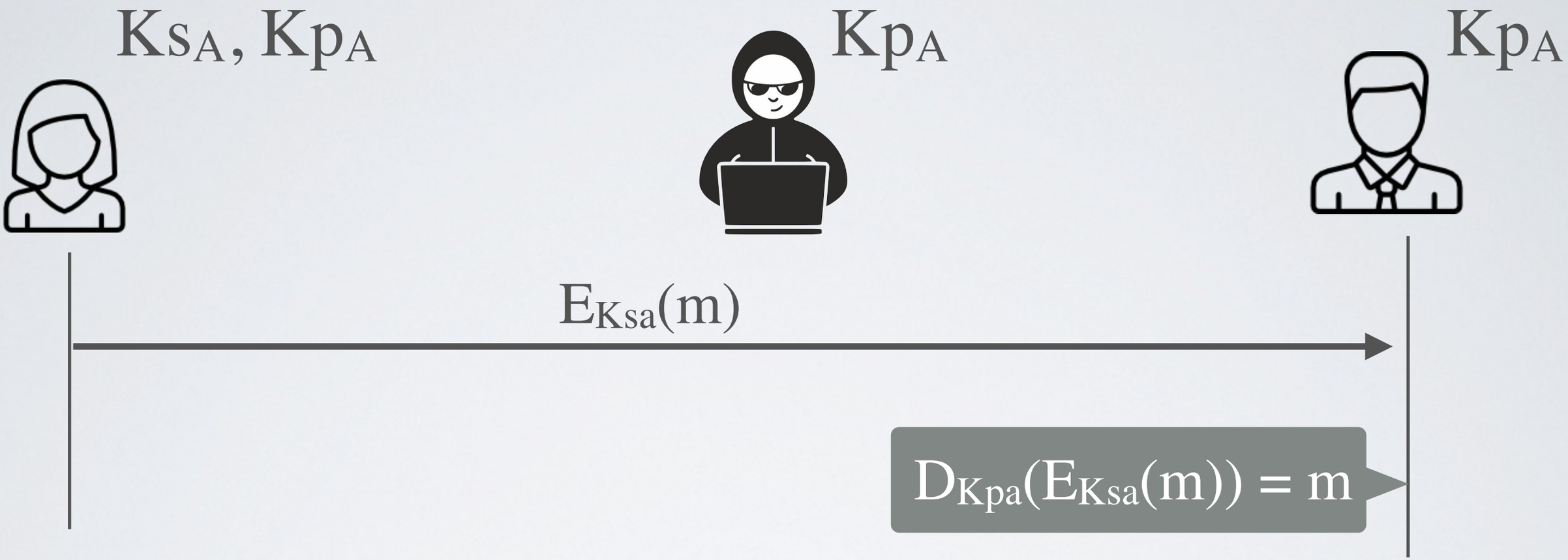
- ✓ Does ensure the confidentiality of the communication
- ◉ Does not authenticate Alice or Bob

Asymmetric encryption: Digital Signature

- The private key for encryption
- The public key for decryption



Asymmetric encryption for **integrity**

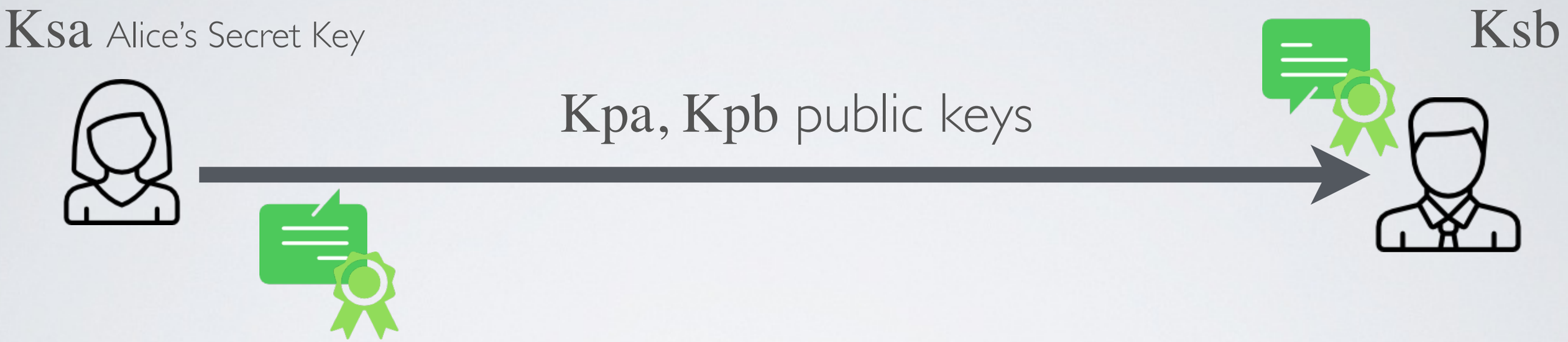


Alice encrypts a message $\mathbf{m}$ with her private key $\mathbf{Ks_A}$

➡ Everybody can decrypt $\mathbf{m}$ using Alice's public key $\mathbf{Kp_A}$

✓ Authentication with non-repudiation (a.k.a Digital Signature)

Digital Signature



➡ Use public cryptography to **sign and verify**

$$m \,||\, SIG_{Ksa}(m)$$

$$SIG_{Ksa}(m) = E_{Ksa}(H(m))$$

How to verify your Ubuntu download

NOTE: You will need to use a terminal app to verify an Ubuntu ISO image. These instructions assume basic knowledge of the command line, checking of SHA256 checksums and use of GnuPG.

Verifying your ISO helps insure the data integrity and authenticity of your download. The process is fairly straightforward, but it involves a number of steps. They are:

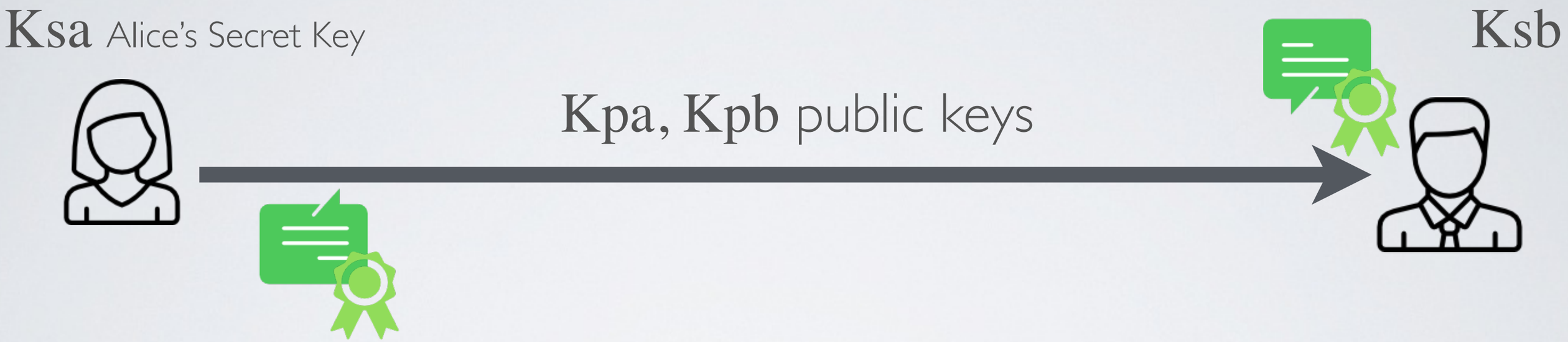
1. Download SHA256SUMS and SHA256SUMS.gpg files
2. Get the key used for the signature from the Ubuntu key server
3. Verify the signature
4. Check your Ubuntu ISO with sha256sum against the downloaded sums

After verifying the ISO file, you can then either install Ubuntu or run it live from your CD/DVD or USB drive.

Non-repudation as a special case of integrity

	MAC	Digital Signature
Integrity	✓	✓
Non-repudiation	✗	✓

Digital Signatures and Confidentiality



1. Alice generates an asymmetric session key k
2. Use both symmetric and asymmetric cryptography to **encrypt, sign and verify** the message and the key

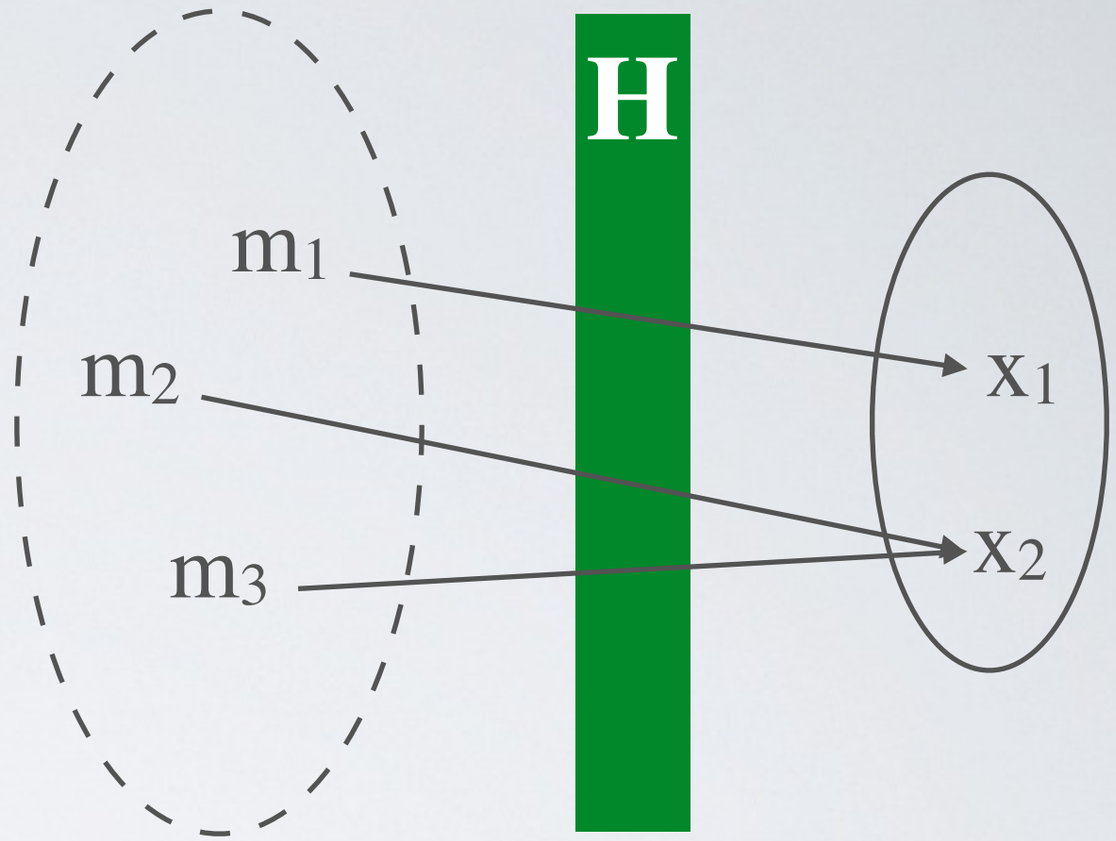
$$E_{Kpb}(k) \,||\, E_k(m \,||\, E_{Ksa}(H(m)))$$

Message digests

Message digests are meant for creating fingerprints of messages

- Un-keyed message digest : hashes, checksum
- Keyed message digests : MACs

Cryptographic hashing



$H(m) = x$ is a hash function if

- H is one-way function
- m is a message of any length
- x is a message digest of a fixed length

➡ H is a lossy compression function
necessarily there exists x, m_1 and m_2 | $H(m_1) = H(m_2) = x$

Computational complexity

- Given H and $\mathbf{m}$, computing $\mathbf{x}$ is **easy** (polynomial or linear)

- Given H and $\mathbf{x}$, computing $\mathbf{m}$ is **hard** (exponential)

➡ H is **not invertible**

Preimage resistance and collision resistance

PR - Preimage Resistance (a.k.a One Way)

➡ given H and x, hard to find m
e.g. password storage

2PR - Second Preimage Resistance (a.k.a Weak Collision Resistance)

➡ given H, m and x, hard to find m' such that $H(m) = H(m') = x$
e.g. virus identification

CR - Collision Resistance (a.k.a Strong Collision Resistance)

➡ given H, hard to find m and m' such that $H(m) = H(m') = x$
e.g. digital signatures

CR → 2PR and CR → PR

Security of hash functions

Brute-forcing a hash function

CR - Collision Resistance

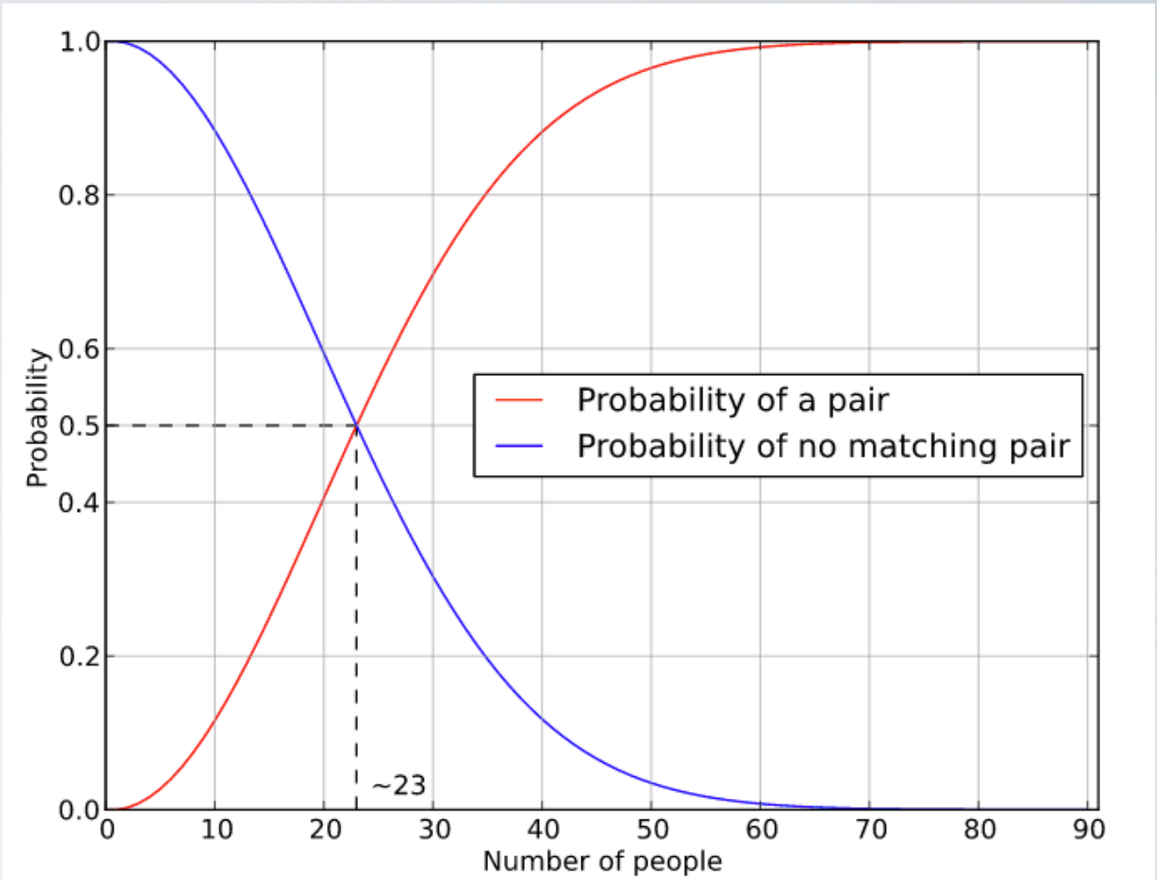
➡ given H, hard to find m and m' such that $H(m) = H(m') = x$

Given a hash function H of n bits output

- Reaching all possibilities 2^n cases
- ~~On average, an attacker should try half of them~~ ~~2^{n-1} cases~~

Birthday Paradox

"There are 50% chance that 2 people have the same birthday in a room of 23 people"



N-bits security

➡ Given a hash function **H** of **n** bits output,
a collision can be found in around $2^{n/2}$ evaluations
e.g SHA-256 is 128 bits security

Broken hash functions beyond the birthday paradox

	Year	Collision
MD5	2013	2^{24} evaluations (2^{39} with prefix)
SHA-1	2015	2^{57} evaluations