



Cryptology - 01410  
Spring 2023

Christian Majenz & Carsten Baum  
Assistant Professors, DTU Compute,  
Technical University of Denmark

# Practicalities

# 01410 - Organisation

Mondays, 13.00 – 17.00

- Lectures 13:00-15:00, Building 308, Auditorium 13
- then exercises 15:00-17:00. Three groups in Building 308, Databars 001, 009 and 017
- 3 Homework sets. 1 over easter break, two “Homework weeks” with no planned lecture content (but there might be a class we are running behind schedule)

**Not** Mondays April 03&10 (Easter)

# Your teachers

## Carsten Baum

Assistant Professor at DTU and Aarhus University

### About me

I'm an Assistant Professor in the Cybersecurity Section at DTU Compute in Copenhagen and in the Computer Science Department at Aarhus University, Denmark. My research focus is on (applied) secure computation as well as (lattice-based) zero knowledge protocols. Furthermore, I am interested into security for machine learning and secure protocol design using public ledgers.

Before starting on this position I have been a Postdoc at Aarhus University and Bar Ilan University, Israel. I obtained my PhD in 2016 from Aarhus University.



Spring 2023

## Christian Majenz

Department of Applied Mathematics and Computer Science, Technical University of Denmark

About me	CV	Research	Publications	Presentations	Teaching	Links	Contact
----------	----	----------	--------------	---------------	----------	-------	---------



I currently hold a personal Veni grant from [NWO](#), which I have taken with me to Denmark via the "[Money Follows Researcher scheme](#)", and I am part of the MSCA doctoral training network "[Quantum-Safe Internet](#)". In addition I have been [awarded a Sapere Aude grant](#) from the Independent Research Fund Denmark, which will start September 2023.

### About me

I am an assistant professor of cryptography at [DTU Compute](#), the Department of Applied Mathematics and Computer Science at the [Technical University of Denmark](#). My research areas are post-quantum and quantum cryptography, and quantum information theory. I find the interplay of the algebraic structure of quantum mechanics and the computational framework of modern cryptography, as well as representation-theoretic and combinatorial techniques, particularly interesting.

I hold a PhD degree in mathematics, which I obtained from the [Department of Mathematical Sciences at University of Copenhagen](#) under the supervision of [Matthias Christandl](#). Before that, I studied [physics at University of Freiburg](#) where my M.Sc. thesis was supervised by [David Gross](#).

Nigel P. Smart

# Cryptography Made Simple

 Springer

# Exercises and Tutoring Classes

One Exercise sheet per week. Work on it during tutoring classes and in between classes!

Three tutoring class groups — three teaching assistants: Freja Elbro, Polly Nielsen Boutet-Livoff and Fabrizio Sisinni.

The exercise problems are very important for actually learning something in this course, please (try to) solve them!!!

# Exam

3 homeworks in course: 30% of the grade

Written examination: 70% of the grade

The practice problems and homework problems are indicative of how exam problems will look like (except the ones that obviously take too much time)

# Homework

3 Homework sheets. Each counts 10 percent of your grade.

You can submit in groups of up to 3 — please team up!

- TA time is valuable! Fairness demands that it is equally divided between you
- $\Rightarrow$  TAs have more time to give constructive written feedback on team submissions (but of course all submissions are graded according to the same standards)



# Changes compared to last year

## Old:

### Læringsmål

En studerende, der fuldt ud har opfyldt kursets mål, vil kunne:

- Foretage beregninger ved modulær aritmetik, herunder Euklids algoritmer og den kinesiske restklassesætning.
- Diskutere forskellene mellem klassisk (symmetrisk) kryptologi og public-key (asymmetrisk) kryptologi.
- Definere det diskrete logaritme problem modulo et primtal og demonstrere anvendelserne i kryptologi.
- Redegøre for hvordan man vælger store primtal til brug i public-key kryptologi.
- Definere egenskaberne ved en digital signatur og **forklare detaljerne i El Gamal's signatursystem**.
- Skitsere anvendelserne af kryptografiske hashfunktioner i kryptologi, og beskrive de ønskelige egenskaber med funktionerne i den pågældende anvendelse.
- Redegøre for hvordan de symmetriske krypteringssystemer, **DES** og AES, anvendes til kryptering og autentificering.
- Præsentere RSA public-key kryptosystemet i alle detaljer, samt forklare hvordan systemet kan bruges til kryptering og til at konstruere digitale signaturer.
- **Forklare hvad "secret-sharing" bruges til og hvordan en hemmelighed deles**.
- Demonstrere hvordan man udveksler en nøgle til symmetrisk kryptering på en sikker måde.

## New:

### Learning objectives

A student who has met the objectives of the course will be able to:

- Do calculations in modular arithmetic, including Euclid's algorithms and the Chinese Remainder Theorem.
- Discuss the differences between classical (symmetric) cryptology and public-key (asymmetrical) cryptology.
- **Explain the functionality and security properties required of symmetric-key and public-key encryption schemes, message authentication codes and digital signature schemes.**
- Describe the design principle of AES.
- Explain how block ciphers are used for encryption and authentication, and analyze the security of modes of operation.
- Define the discrete logarithm problem modulo a prime number and demonstrate the applications in cryptology.
- Explain how to find big prime numbers for use in public-key cryptology.
- Outline the applications of cryptographic hash functions in cryptology, and describe the desired properties of the functions in the particular application.
- Present the RSA public-key cryptosystem in all details, and explain how the system is used for encryption and to construct digital signatures.
- Demonstrate how to exchange a key for symmetric encryption securely using Diffie-Hellman key exchange.
- **Discuss the quantum threat to cryptography.**
- **Explain the Learning With Errors problem and Regev's encryption scheme.**

# I'm also learning...

Parts of this year's course will be the "Capstone Project" for a the DTU teacher training program I am enrolled in

⇒ I will ask for feedback a lot, please help me out and respond!

# Cryptography

# What is cryptography?



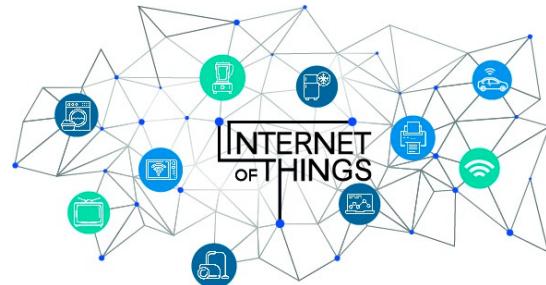
# Cryptography

cryptography(=cryptology): the science (and art) of constructing and analyzing cryptographic schemes

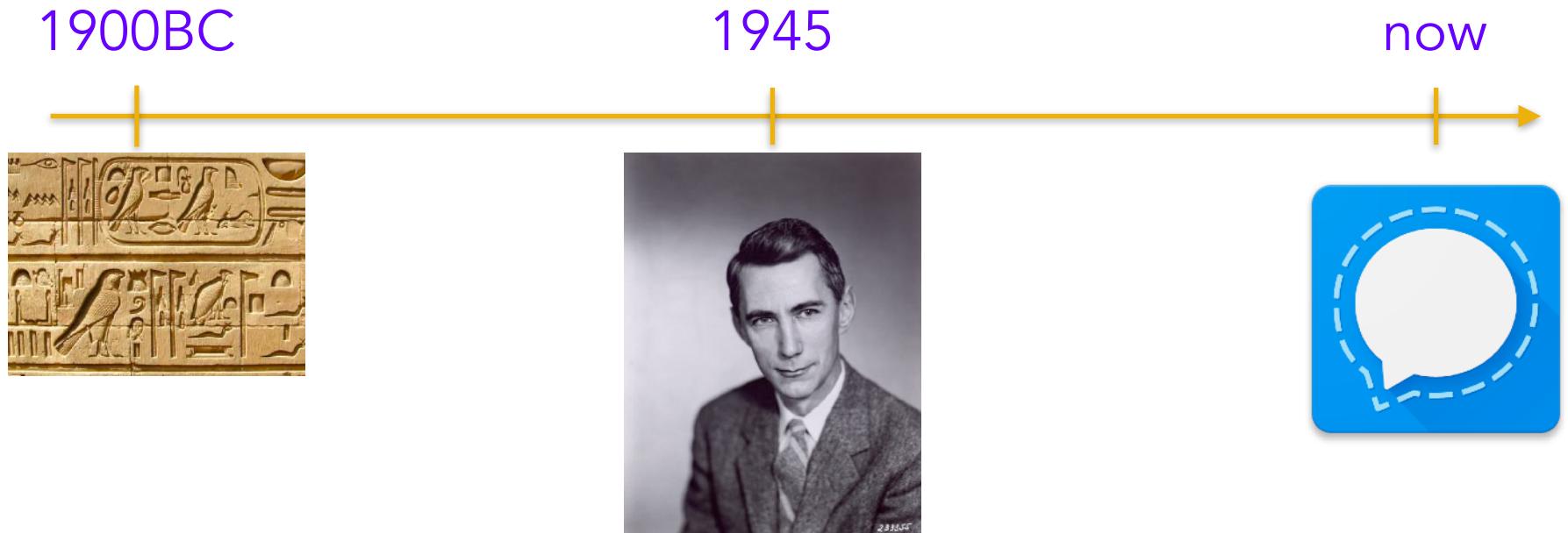
Examples:

- encryption schemes
- authentication schemes
- digital signatures
- hash functions
- zero-knowledge proofs, secure multiparty computation, program obfuscation, block chain, fully homomorphic/functional/identity-based/attribute-based/threshold encryption...

# Where is cryptography used?



# History of cryptography - seen from Mars



# Ciphers/Encryption schemes

Definition, historical ciphers

( $\Rightarrow$ blackboard)

# Cryptography 1

Luisa Siniscalchi

(These slides are taken from the lectures of Prof. Jonathan Katz)

## Defining Security of Encryption Scheme

If you don't understand what you want to achieve, how can you possibly know when (or if) you have achieved it? (Book:

Introduction to Modern Cryptography 2nd ed. CRC Press 2015)

# Defining security of encryption scheme

## On the need of formal definitions

- ▶ What does it mean for a scheme to be secure?
  - ▶ What do we want the adversary **to not** be able to achieve?
  - ▶ What are the capabilities of the adversary?

# Defining security of encryption scheme

## On the need of formal definitions

- ▶ What does it mean for a scheme to be secure?
  - ▶ What do we want the adversary to not be able to achieve?
  - ▶ What are the capabilities of the adversary?

## Formal definitions help because...

- ▶ Definitions enable meaningful analysis, evaluation, and comparison of schemes.

Formal Definition Encryption Scheme ...

## Private-key encryption

- ▶  $\mathbb{K}$  (key space): set of all possible keys
- ▶  $\mathbb{M}$  (message space): set of all possible messages
- ▶  $\mathbb{C}$  (ciphertext space): set of all possible ciphertexts

## Private-key encryption

A private-key encryption scheme is defined by a message space  $\mathbb{M}$ , (key space  $\mathbb{K}$ ) and algorithms  $e$  and  $d$ :

- ▶ **KeyGen** (key-generation algorithm): outputs  $\mathbf{k} \in \mathbb{K}$ .  
Usually:  $\mathbf{k} \in \mathbb{K}$  uniformly random. (This algorithm is sometimes left implicit in the book)

# Private-key encryption

A private-key encryption scheme is defined by a message space  $\mathbb{M}$ , (key space  $\mathbb{K}$ ) and algorithms  $(\text{KeyGen}, \text{e}, \text{d})$  :

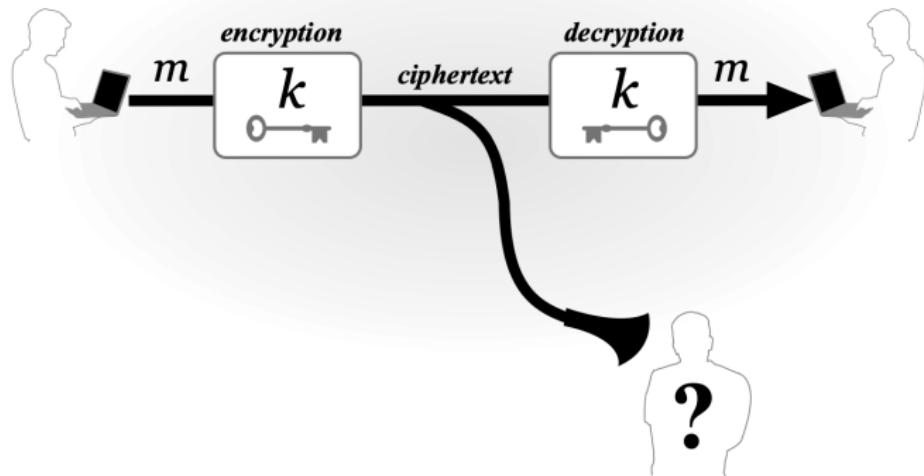
- ▶ **KeyGen** (key-generation algorithm): outputs  $\mathbf{k} \in \mathbb{K}$ .  
Usually:  $\mathbf{k} \in \mathbb{K}$  uniformly random. (This algorithm is sometimes left implicit in the book)
- ▶ **e** (encryption algorithm): takes as input key  $\mathbf{k}$  and message  $\mathbf{m} \in \mathbb{M}$ ; outputs ciphertext  $\mathbf{c} \leftarrow \text{e}_{\mathbf{k}}(\mathbf{m})$

## Private-key encryption

A private-key encryption scheme is defined by a message space  $\mathbb{M}$ , (key space  $\mathbb{K}$ ) and algorithms  $(\text{KeyGen}, \text{e}, \text{d})$  :

- ▶  $\text{KeyGen}$  (key-generation algorithm): outputs  $\mathbf{k} \in \mathbb{K}$ .  
Usually:  $\mathbf{k} \in \mathbb{K}$  uniformly random. (This algorithm is sometimes left implicit in the book)
- ▶  $\text{e}$  (encryption algorithm): takes as input key  $\mathbf{k}$  and message  $\mathbf{m} \in \mathbb{M}$ ; outputs ciphertext  $\mathbf{c} \leftarrow \text{e}_{\mathbf{k}}(\mathbf{m})$
- ▶  $\text{d}$  (decryption algorithm): takes as input key  $\mathbf{k}$  and ciphertext  $\mathbf{c}$ ; outputs  $\mathbf{m}$  or "error":  $\mathbf{m} = \text{d}_{\mathbf{k}}(\mathbf{c})$

# Private-key encryption



## What are the capabilities of the adversary $\mathcal{A}$ ?

- ▶ Ciphertext-only attack ( $\mathcal{A}$  has access only at ciphertext, specifically:
  - ▶ One ciphertext
  - ▶ Many ciphertexts
- ▶ Known-plaintext attack ( $\mathcal{A}$  has access to pairs of known plaintexts and their corresponding ciphertexts)
- ▶ Chosen-plaintext attack ( $\mathcal{A}$  has the ability to choose plaintexts and to view their corresponding ciphertexts)
- ▶ Chosen-ciphertext attack ( $\mathcal{A}$  has the ability to obtain the decryption of ciphertexts of its choice)

## Define secure encryption

- ▶ What does it mean for encryption scheme  $(\text{KeyGen}, e, d)$  to be **secure**?
- ▶ When  $\mathcal{A}$  has access only to one ciphertext.

# Define secure encryption. Attempt 1

**$\mathcal{A}$  does not learn the key**

- Consider the scheme  $e_k(m) = m$

## Define secure encryption. Attempt 2

**$\mathcal{A}$  does not learn the plaintext from the ciphertext**

- ▶ What if the adversary learns only a part of the plaintext?
- ▶ What if the adversary is able to learn some partial information about the plaintext? (e.g. is the salary **> 30.000 DKK**)

## Define secure encryption. Attempt 3:

### Perfect Secrecy

Regardless of any **prior information**, the adversary has about the plaintext, the ciphertext should leak **no additional information** about the plaintext

# Perfect Secrecy

Let us start with recalling some probability elements...

# Probability Review

## Random variable ( $RV$ )

Variable that takes on (discrete) values with certain probabilities

## Probability distribution ( $PD$ )

A  $PD$  for a  $RV$  specifies the probabilities with which the variable takes on each possible value

- ▶ Each probability must be between **0** and **1**
- ▶ The probabilities must sum to **1**

# Probability Review

## Event

A particular occurrence in some experiments:

- $\Pr[E]$ : probability of event  $E$

## Conditional probability

Probability that one event occurs, given that some other event occurred:

- $\Pr[A|B] = \Pr[A \text{ and } B]/\Pr[B] \equiv \Pr[AB]/\Pr[B]$

## Independence

Two  $RV$   $X, Y$  are **independent** if:

- $\forall x, y : \Pr[X = x|Y = y] = \Pr[X = x]$

# Probability Review

Law of total probability

Let  $E_1 \dots E_n$  are a partition of all possibilities. Then  $\forall A$ :

$$\begin{aligned}\Pr[A] &= \sum_i \Pr[AE_i] \\ &= \sum_i \Pr[A|E_i] \Pr[E_i]\end{aligned}$$

Note

$$\Pr[A|B] = \Pr[AB]/\Pr[B] \implies \Pr[AB] = \Pr[A|B]\Pr[B]$$

# Probability Distributions

## The random variable $M$

- $M$  is the *RV* denoting the value of the message
- $M$  ranges over  $\mathbb{M}$ ; context dependent
- Reflects the likelihood of different messages being sent, given the adversary's **prior knowledge**

## Example

$$\Pr[M = \text{attack today}] = 0.7$$

$$\Pr[M = \text{don't attack}] = 0.3$$

# Probability Distributions

## The random variable $K$

- $K$  is the *RV* denoting the key
- $K$  ranges over  $\mathbb{K}$
- Fix some encryption scheme  $(\text{KeyGen}, e, d)$
- $\text{KeyGen}$  defines a probability distribution for  $K$ :

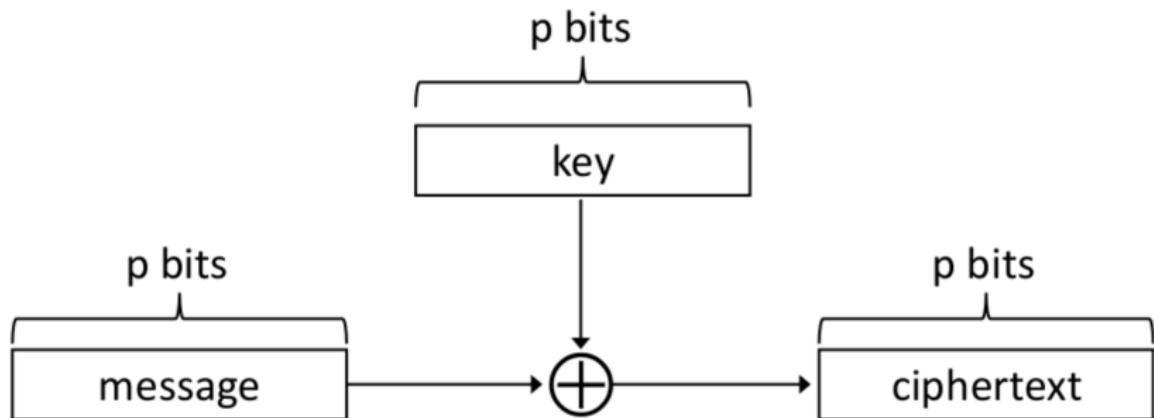
$$\Pr[K = k] = \Pr[\text{KeyGen outputs key } k]$$

# Probability distributions

## The random variable $C$

- ▶ Fix some encryption scheme  $(\text{KeyGen}, \text{e}, \text{d})$  , and some  $PD$  for  $M$
- ▶ Consider the following (randomized) experiment:
  - ▶ Generate a key  $k$  using  $\text{KeyGen}$
  - ▶ Choose a message  $m$ , according to the given  $PD$
  - ▶ Compute  $c \leftarrow \text{e}_k(m)$
- ▶ **This defines a distribution on the ciphertext**
- ▶ Let  $C$  be a  $RV$  denoting the value of the ciphertext in this experiment

# One-time Pad



## Private-key encryption

A private-key encryption scheme is defined by a message space  $\mathbb{M}$ , (key space  $\mathbb{K}$ ) and algorithms  $(\text{KeyGen}, \text{e}, \text{d})$  :

- ▶  $\text{KeyGen}$  (key-generation algorithm): outputs  $\mathbf{k} \in \mathbb{K}$ .  
Usually:  $\mathbf{k} \in \mathbb{K}$  uniformly random. (This algorithm is sometimes left implicit in the book)
- ▶  $\text{e}$  (encryption algorithm): takes as input key  $\mathbf{k}$  and message  $\mathbf{m} \in \mathbb{M}$ ; outputs ciphertext  $\mathbf{c} \leftarrow \text{e}_{\mathbf{k}}(\mathbf{m})$
- ▶  $\text{d}$  (decryption algorithm): takes as input key  $\mathbf{k}$  and ciphertext  $\mathbf{c}$ ; outputs  $\mathbf{m}$  or "error":  $\mathbf{m} = \text{d}_{\mathbf{k}}(\mathbf{c})$

# One-time Pad

- ▶ Let  $\mathbb{M} = \{0, 1\}^n$
- ▶ KeyGen: choose a uniform key  $k \in \{0, 1\}^n$
- ▶  $e_k(m) = k \oplus m$
- ▶  $d_k(c) = k \oplus c$
- ▶  $d_k(e_k(m)) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = m$

# Perfect Secrecy of One-time Pad

## Theorem

*The One-time Pad satisfies perfect secrecy.*

## Intuition

- Having observed a ciphertext, the attacker cannot conclude for certain which message was sent

# One-time Pad and Brute-force Attacks

The same ciphertext	Decrypted with this key...	...gives this plaintext
SMAIJIZJSIFPSTWFI	→ STHIHZQRBPIONWP	→ ATTACKATBREAKFAST
	→ BIHRFIGIODRYOGIRV	→ RETREATBEFORENOON
	→ MYARVOMGKVDHBRLBQ	→ GOAROUNDINCIRCLES
	→ ATAVGOGQORURAAOUX	→ STANDUTTERLYSTILL
	→ AENCQMLCSTQRAFJZQ	→ SINGTWOHAPPYSONGS
	→ AFMOQIHYEOPCAEINQ	→ SHOUTASLOUDASPOSS
	→ IIWTQUGJHXHXQMDLW	→ KEEPTOTALLYSCHTUM
	→ SBPUKKPKZTRXALVUE	→ ALLOUTPUTPOSSIBLE

- ▶ OTP resists even a brute-force attack
- ▶ Decrypt a ciphertext with every key returns every possible plaintext (incl. every ASCII/English string)
- ▶ No way of telling the correct plaintext

## Perfect Secrecy of One-time Pad

Proof.

- ▶ Fix arbitrary distribution over  $\mathbb{M} = \{0, 1\}^n$ , and choose arbitrary  $m, c \in \{0, 1\}^n$
- ▶ Check if

$$\Pr[M = m | C = c] = \Pr[M = m]$$

# Perfect Secrecy of One-time Pad

Proof.

- Recall (Bayes' theorem)

$$\Pr[M = m | C = c] = \frac{\Pr[C = c | M = m] \Pr[M = m]}{\Pr[C = c]}$$

- We can see that  $\forall c, m$

$$\begin{aligned}\Pr[C = c | M = m] &= \Pr[M \oplus K = c | M = m] = \\ &= \Pr[m \oplus K = c] = \Pr[K = c \oplus m] = 2^{-n}\end{aligned}$$

# Perfect Secrecy of One-time Pad

Proof.

By law of total probability:

$$\begin{aligned}\Pr[C = c] &= \\ &= \sum_{m'} \Pr[C = c | M = m'] \Pr[M = m'] \\ &= \sum_{m'} \Pr[K = m' \oplus c | M = m'] \Pr[M = m'] \\ &= \sum_{m'} 2^{-n} \Pr[M = m'] \\ &= 2^{-n} \sum_{m'} \Pr[M = m'] = 2^{-n}\end{aligned}$$

# Perfect Secrecy of One-time Pad

Proof.

$$\begin{aligned}\Pr[M = m | C = c] &= \\ &= \frac{\Pr[C = c | M = m] \Pr[M = m]}{\Pr[C = c]} \\ &= \frac{\Pr[K = m \oplus c | M = m] \Pr[M = m]}{2^{-n}} \\ &= \frac{2^{-n} \Pr[M = m]}{2^{-n}} \\ &= \Pr[M = m]\end{aligned}$$

□

# One-time Pad

- ▶ The One-time Pad achieves perfect secrecy!
- ▶ Resists even a brute-force attack
- ▶ Not currently used! Why?

# One-time Pad

## Limitations of OTP

1. The key is as long as the message
2. A key must be used only once
  - Only secure if each key is used to encrypt a single message
  - (Trivially broken by a known-plaintext attack)

⇒ Parties must share keys of (total) length equal to the (total) length of all the messages they might ever send

## Using the Same Key Twice?

- ▶ Say

$$c_1 = k \oplus m_1$$

$$c_2 = k \oplus m_2$$

- ▶ Attacker can compute

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

- ▶ This leaks information about  $m_1, m_2$

# Using the Same Key Twice?

$m_1 \oplus m_2$  leaks information about  $m_1, m_2$

Is this significant?

- ▶  $m_1 \oplus m_2$  reveals where  $m_1, m_2$  differ
- ▶ No longer perfectly secret!
- ▶ Exploiting characteristics of ASCII...

# ASCII table (recall)

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	8	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	:	91	5B	[	123	7B	(
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	)
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	□	127	7F	□

<https://hubpages.com/technology/What-Are-ASCII-Codes>

# Using the Same Key Twice: recall ASCII

## Observations

- ▶ Letters begin with 0x4, 0x5, 0x6 or 0x7
  - ▶ **⇒ letters all begin with 01...**
- ▶ ASCII code for the space character 0x20 = **00100000**
  - ▶ **⇒ the space character begins with 00...**
- ▶ XOR of two letters gives **00...**
- ▶ XOR of letter and space gives **01...**
- ▶ **Easy to identify XOR of letter and space!**

# One-time Pad

## Drawbacks

- ▶ Key as long the message
- ▶ Only secure if each key is used to encrypt once
- ▶ Trivially broken by a known-plaintext attack

## Perfect Secrecy (PS)

Is the notion too strong?

PS requires that absolutely **no information** about the plaintext is leaked, even to eavesdroppers with **unlimited computational power**

- ▶ Has some inherent drawbacks
- ▶ Seems **unnecessarily strong**

# Computational Secrecy (CS)

A weaker, yet practical notion

- ▶ Still fine if a scheme **leaks information** with tiny probability to eavesdroppers with **bounded computational resources**
- ▶ i.e. we can **relax perfect secrecy** by
  1. Allowing security to "fail" with tiny probability
  2. Restricting attention to "efficient" attackers

## Tiny probability of failure?

- ▶ Say security fails with probability  $2^{-60}$
- ▶ Should we be concerned about this?
- ▶ With probability  $> 2^{-60}$ , the sender and receiver will both be struck by lightning in the next year...
- ▶ Something that occurs with probability  $2^{-60}/\text{sec}$  is expected to occur once every **100** billion years

## Bounded attackers?

- ▶ Consider brute-force search of key space; assume one key can be tested per clock cycle
- ▶ Desktop computer  $\approx 2^{57}$  keys/year
- ▶ Supercomputer  $\approx 2^{80}$  keys/year
- ▶ Supercomputer since Big Bang  $\approx 2^{112}$  keys
- ▶ Therefore restricting attention to attackers who can try  $2^{112}$  keys is fine!
- ▶ Modern key space:  $2^{128}$  keys or more...

# Symmetric Encryption

February 13, 2023

# AES - Advanced Encryption Standard

- US governmental encryption standard
- Keys: choice of 128-bit, 192-bit, and 256-bit keys
- Blocks: 128 bits
- Open (world) competition announced January 97
- Standard: FIPS 197, November 2001

# AES=Rijndael

- Designed by Joan Daemen and Vincent Rijmen
- Simple design, only byte operations
- S-box, substitutes one byte by another byte
- Iterated cipher

Key size	128	192	256
Number of rounds	10	12	14

# AES round transformation

Arrange the 16 input bytes in a  $4 \times 4$  matrix

## Subfunctions

- ① AddRoundKey
- ② SubBytes (byte substitution via S-box)
- ③ ShiftRows
- ④ MixColumns

# AddRoundKey (bit-wise XOR)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

 $\oplus$ 

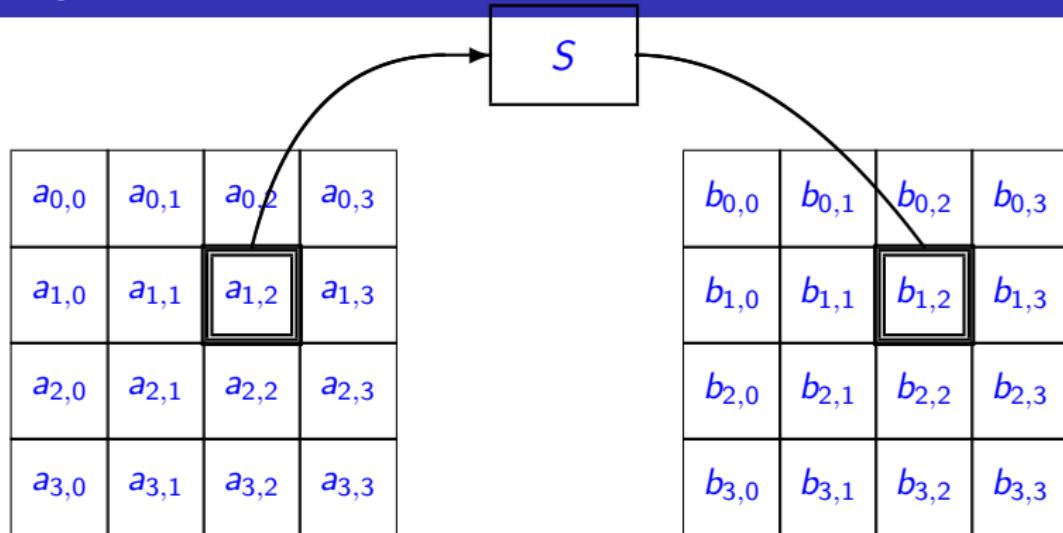
$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

 $=$ 

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

$$b_{i,j} = a_{i,j} \oplus k_{i,j}$$

# SubBytes



$$b_{i,j} = S(a_{i,j})$$

$S : \{0, 1\}^8 \rightarrow \{0, 1\}^8$  is the invertible S-box

$S$  is a very simple non-linear function (field inversion)

# ShiftRows

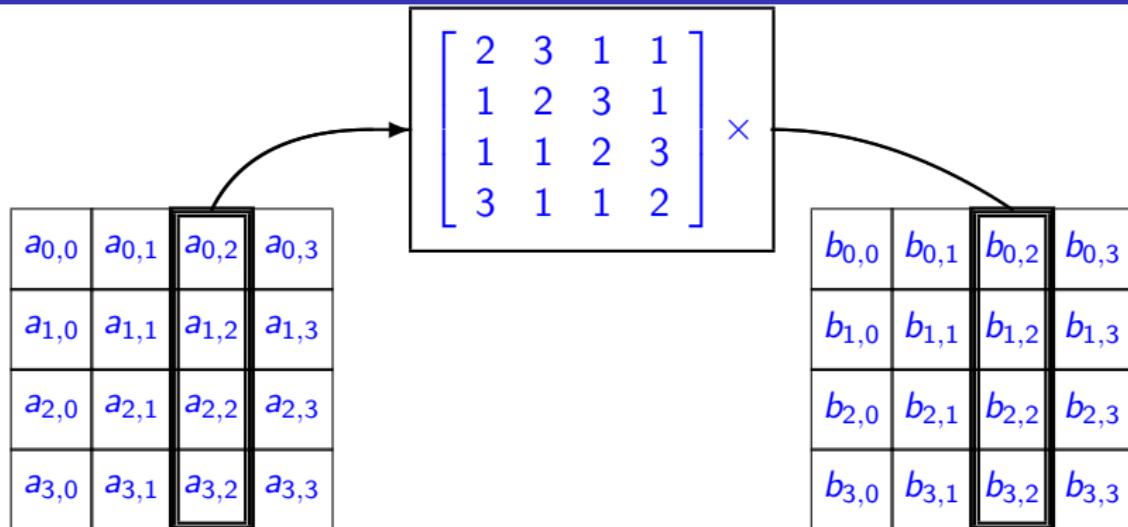
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$



$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

Rows shifted over different offsets: 0,1,2, and 3

# MixColumns



Bytes in columns are combined linearly

$$b_{0,2} = \{2\} \times a_{0,2} + \{3\} \times a_{1,2} + \{1\} \times a_{2,2} + \{1\} \times a_{3,2}$$

Multiplication is a special field-multiplication

# AES - 10-round version

Arrange the 16 input bytes in a  $4 \times 4$  matrix

- AddRoundKey
- Do nine times
  - SubBytes (byte substitution via S-box)
  - ShiftRows
  - MixColumns
  - AddRoundKey
- SubBytes
- ShiftRows
- AddRoundKey

# Byte mixing in AES


Shift  
→  
Rows


Mix  
→  
Col.

♠			
♠			
♠			
♠			

Shift  
→  
Rows

♠			
			♠
			♠
	♠		

Mix  
→  
Col.

♠	♠	♠	♠
♠	♠	♠	♠
♠	♠	♠	♠
♠	♠	♠	♠

# Modes of operation for block ciphers

Block cipher with  $n$ -bit blocks, e.g. DES:  $n = 64$ , AES:  $n = 128$

Message  $m$  split into blocks of  $n$  bits, i.e.,

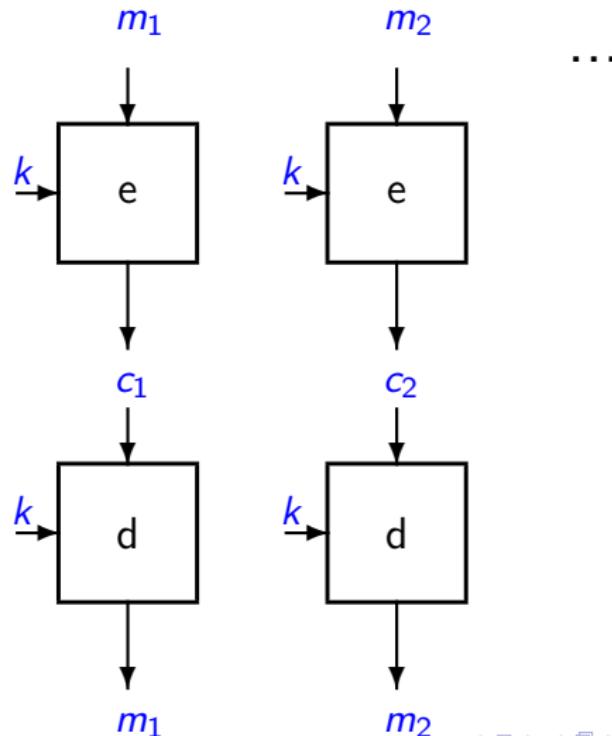
$$m = m_1, m_2, \dots, m_t,$$

where  $|m_i| = n$

Many modes of operation: ECB (dangerous, don't use), CBC, CFB, OFB, CTR, GCM...

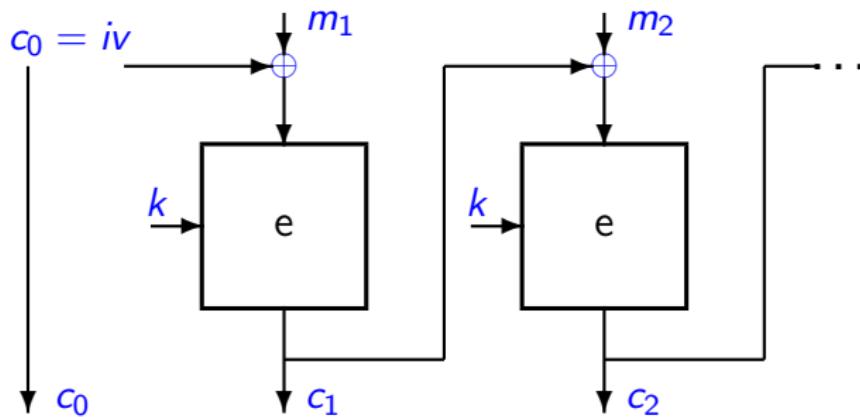
# ECB mode (dangerous, don't use)

Encryption and decryption



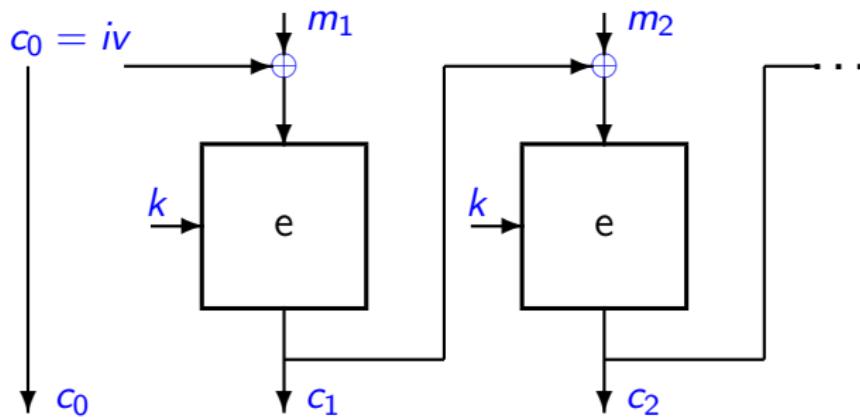
# CBC mode

## Encryption



# CBC mode

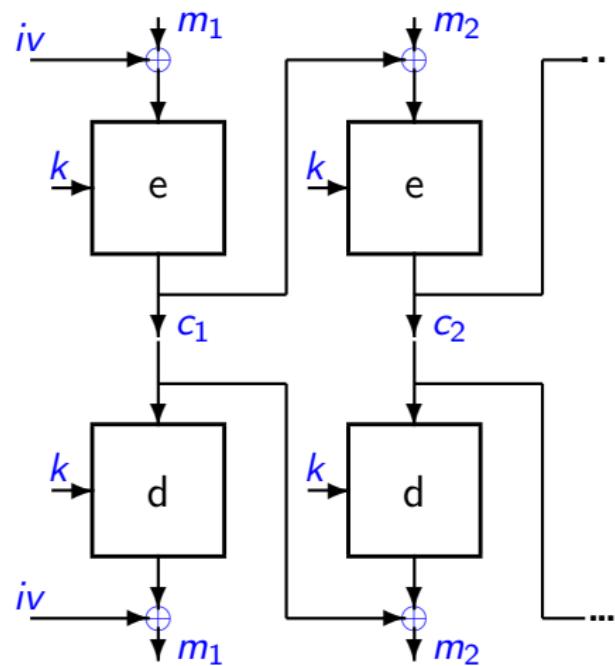
## Encryption



How does decryption work?

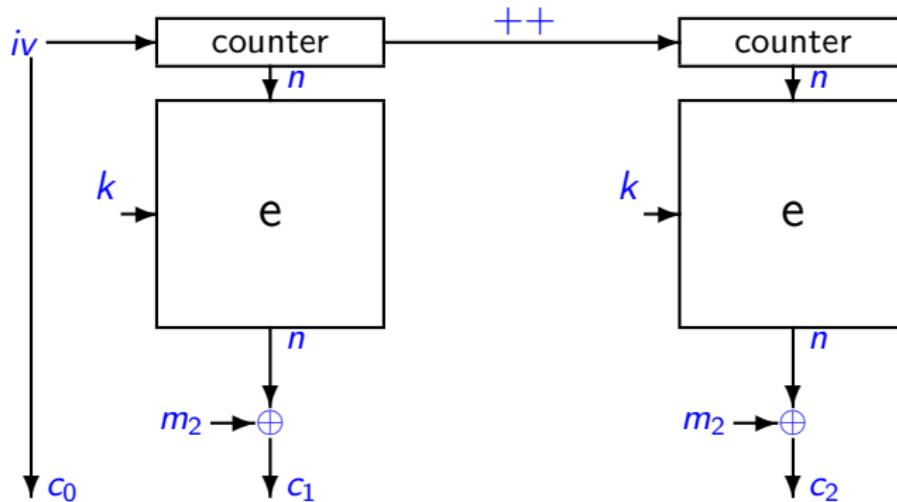
# CBC mode

## Encryption and decryption



# CTR mode

$n$  and  $m$  are sizes in bits



# Logistics

Carsten Baum

Building 322, room 210

Exercises as usual

Next Monday: Homework (sheet is online),  
i.e. no lecture

Homework is due on 20.03

# Public Key Cryptography

And the RSA cryptosystem

# Schedule for today

## Math recap

1. Modular arithmetic
2. Gcd and the (extended) Euclidean algorithm
3. Coprimality and Euler's totient function
4. Multiplicative inverses and how to compute them
5. Lagrange's Theorem

## Actual cryptography

1. What is Public Key Encryption?
2. Defining Security of PKE
3. The RSA cryptosystem
4. Why RSA decryption works

# Modular arithmetic

Let  $N$  be a positive integer, called **modulus**

If we divide  $a$  by  $N$  over the integers, then  $a = b + kN$  where  $0 \leq b < N$  is unique

We call  $b$  the remainder of the division

Two integers  $a, b$  are called **congruent** if  $N|(b - a)$  and we write  $a = b \pmod{N}$

Since  $0 = N \pmod{N}$ ,  $1 = N + 1 \pmod{N}$  every integer is equal to  $0, \dots, N - 1$  modulo  $N$

We write the remainders as  $Z_N = \{0, \dots, N - 1\}$

# Examples

Modular arithmetic  $mod\ 11$

$$24 = 2 + 2 \cdot 11, \text{ so } 24 = 2 \bmod 11 \rightarrow 11|(24 - 2)$$

Any integer when divided by 11 must be a unique number between 0 and 10

$$\mathbb{Z}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

# Examples continued

Modular arithmetic  $mod\ 11$

$$24 = 2 \ mod\ 11$$

$$24 + 3 = 27 = 5 \ mod\ 11, 2 + 3 = 5 \ mod\ 11$$

$$24 \cdot 3 = 72 = 6 \ mod\ 11, 2 \cdot 3 = 6 \ mod\ 11$$

If we add (or multiply)  $mod\ 11$  it does not matter if we start from 24 or 2.

# Rules of modular arithmetic

1. If  $a = x \bmod N$  and  $b = y \bmod N$  then
  1.  $a + b = x + y \bmod N$
  2.  $a \cdot b = x \cdot y \bmod N$
2. Associativity:  
$$(a + b) + c = a + (b + c) \bmod N \text{ and } (a \cdot b) \cdot c = a \cdot (b \cdot c)$$
3. Commutativity:  
$$a + b = b + a \bmod N \text{ and } a \cdot b = b \cdot a \bmod N$$

# Rules of modular arithmetic

4. Identity elements:

$$a + 0 = a \bmod N \text{ and } a \cdot 1 = a \bmod N$$

5. Distributivity:

$$(a + b) \cdot c = a \cdot c + b \cdot c \bmod N$$

6.  $a + (N - a) = 0 \bmod N$

# Computing the Greatest Common Divisor

$\gcd(x, y)$  : largest positive integer  $d$  such that  $d|x$  and  $d|y$

Example:  $\gcd(12,8) = 4$ ,  $\gcd(9,3) = 3$ ,  $\gcd(11,12) = 1$

If  $\gcd(x, y) = 1$  then we say  $x, y$  are coprime

An algorithm to compute the  $gcd$ : the Euclidean algorithm

# How the Euclidean algorithm works

To compute  $\gcd(x, y)$ :

1. Define  $r_0 = x, r_1 = y$
2. Iteratively in round  $i \in \{1, \dots\}$ 
  1. Divide  $r_{i-1}$  by  $r_i$ , obtaining remainder  $r_{i+1}$  (i.e.  $r_{i+1} = r_{i-1} \bmod r_i$ )
  2. If  $r_{i+1} = 0$  output  $r_i$

The algorithm always terminates, because  $r_{i+1} < r_i$  but division remainder never  $< 0$

The output is correct, because  $\gcd(r_0, r_1) = \gcd(r_1, r_2) = \dots = \gcd(r_{i-1}, r_i)$

# Computing the gcd - example

$\text{gcd}(12,8)$ :

$$1. \ 4 = 12 - 8 \cdot 1$$

$$2. \ 0 = 8 - 4 \cdot 2$$

$$\rightarrow \text{gcd}(12,8) = 4$$

$\text{gcd}(12,7)$ :

$$1. \ 5 = 12 - 7 \cdot 1$$

$$2. \ 2 = 7 - 5 \cdot 1$$

$$3. \ 1 = 5 - 2 \cdot 2$$

$$4. \ 0 = 2 - 1 \cdot 2$$

# Extended Euclidean Algorithm

In addition to Euclidean Algorithm, keep track of linear combinations

$\gcd(12,7)$ :

$$1. \ 5 = 1 \cdot 12 - 7 \cdot 1$$

$$2. \ 2 = 7 - 5 \cdot 1 = 7 - (1 \cdot 12 - 7 \cdot 1) \cdot 1 = -1 \cdot 12 + 7 \cdot 2$$

$$\begin{aligned}3. \ 1 &= 5 - 2 \cdot 2 \\&= (1 \cdot 12 - 7 \cdot 1) - (-1 \cdot 12 + 7 \cdot 2) \cdot 2 \\&= 3 \cdot 12 - 7 \cdot 5\end{aligned}$$

$$4. \ 0 = 2 - 1 \cdot 2$$

# Formal Extended Euclidean Algorithm

$egcd(a, b)$ :

1.  $s \leftarrow 0, s' \leftarrow 1, t \leftarrow 1, t' \leftarrow 0, r \leftarrow b, r' \leftarrow a$
2. While  $r \neq 0$ 
  1.  $q = \lfloor \frac{r'}{r} \rfloor$
  2.  $(r', r) \leftarrow (r, r' - q \cdot r)$
  3.  $(s', s) \leftarrow (s, s' - q \cdot s)$
  4.  $(t', t) \leftarrow (t, t' - q \cdot t)$
3.  $d \leftarrow r', x \leftarrow t, y \leftarrow s$
4. Output  $d, x, y$  such that  $d = \gcd(a, b) = x \cdot a + y \cdot b$

# Coprinality and Euler's Totient function

$a$  is coprime to  $N$  iff  $\gcd(a, N) = 1$

Euler's totient function  $\varphi(N)$

How many numbers  $1 \leq a < N$  fulfill  $\gcd(a, N) = 1$

Given  $N = p_1^{e_1} \cdots p_n^{e_n}$  prime factorization it is known that  
 $\varphi(N) = p_1^{e_1-1}(p_1 - 1) \cdots p_n^{e_n-1}(p_n - 1)$

If  $N = p \cdot q$  then  $\varphi(N) = (p - 1) \cdot (q - 1)$

# Coprimality and Euler's totient function

Let  $N = 6$  then which numbers are coprime?

$$\gcd(1,6) = 1$$

$$\gcd(2,6) = 2$$

$$\gcd(3,6) = 3$$

$$\gcd(4,6) = 2$$

$$\gcd(5,6) = 1$$

But we need to know factorization of  $N$

Faster:  $N = 6 = 2 \cdot 3$  so  $\varphi(N) = (2 - 1) \cdot (3 - 1) = 2$

# How does coprimality help us?

Consider we want to solve  $a \cdot x = 1 \bmod N$  by finding  $x$

Does a solution exist?

If  $a \cdot x = 1 \bmod N$  then  $a \cdot x + k \cdot N = 1$ :

If  $\gcd(a, N) = 1$  then  $\text{egcd}(a, N)$  can compute  $x$

Also: if  $\gcd(a, N) = 1$  then  $x$  is unique

# Example

We know that  $\gcd(12,7) = 1$  so we can solve  $12 \cdot x = 1 \bmod 7$

$\gcd(12,7) :$

$$1. \ 5 = 12 - 7 \cdot 1$$

$$2. \ 2 = 7 - 5 \cdot 1 = 7 - (12 - 7 \cdot 1) \cdot 1 = -12 + 7 \cdot 2$$

$$3. \ 1 = 5 - 2 \cdot 2 = (12 - 7 \cdot 1) - (-12 + 7 \cdot 2) \cdot 2 = 3 \cdot 12 - 7 \cdot 5$$

$$4. \ 0 = 2 - 1 \cdot 2$$

Correct!  $3 \cdot 12 = 36 = 1 + 5 \cdot 7 = 1 \bmod 7$

# Lagrange's Theorem

For positive integer  $N$  define  $Z_N^* = \{x \in Z_N \mid \gcd(x, N) = 1\}$

Euler Totient function:  $|Z_N^*| = \varphi(N)$

Then for all  $x \in Z_N^*$ :  $x^{\varphi(N)} = 1 \bmod N$

Meaning:  $x^{\varphi(N)-1}$  is the inverse for  $x \bmod N$  for every coprime number!

# Example

$$N = 12 = 3 \cdot 2^2$$

$$\varphi(N) = 2 \cdot 2 = 4$$

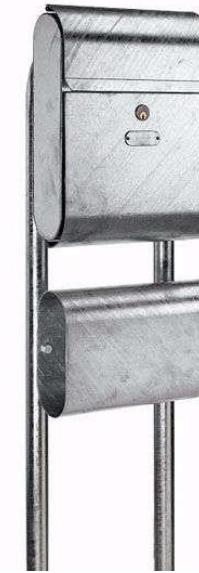
Coprime numbers:  $\mathbb{Z}_N^* = \{1, 5, 7, 11\}$

$$1^4 = 1 \cdot 1^3 = 1 \bmod 12$$

$$5^4 = 5 \cdot 125 = 5 \cdot (10 \cdot 12 + 5) = 25 = 2 \cdot 12 + 1 = 1 \bmod 12$$

$$7^4 = 7 \cdot 343 = 7 \cdot (28 \cdot 12 + 7) = 49 = 4 \cdot 12 + 1 \bmod 12$$

# Public Key Encryption



# Disadvantages of Symmetric Cryptography

- The **chicken-and-egg** problem
  - You need a shared key  $k$  to establish a secure channel
  - You need a secure channel to share the key
- **Scalability** problems
  - A network of  $n$  users needs  $n(n - 1)/2$  exchanged keys
    - $O(n^2)$  for  $n$  nodes
  - Collaborative networks (e.g. sensor networks) may use a single network-wide key
    - If one node gets compromised, whole network get compromised

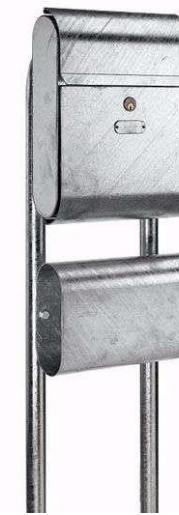
# Public key (asymmetric) encryption

Involves two separate but mathematically related keys **per user**

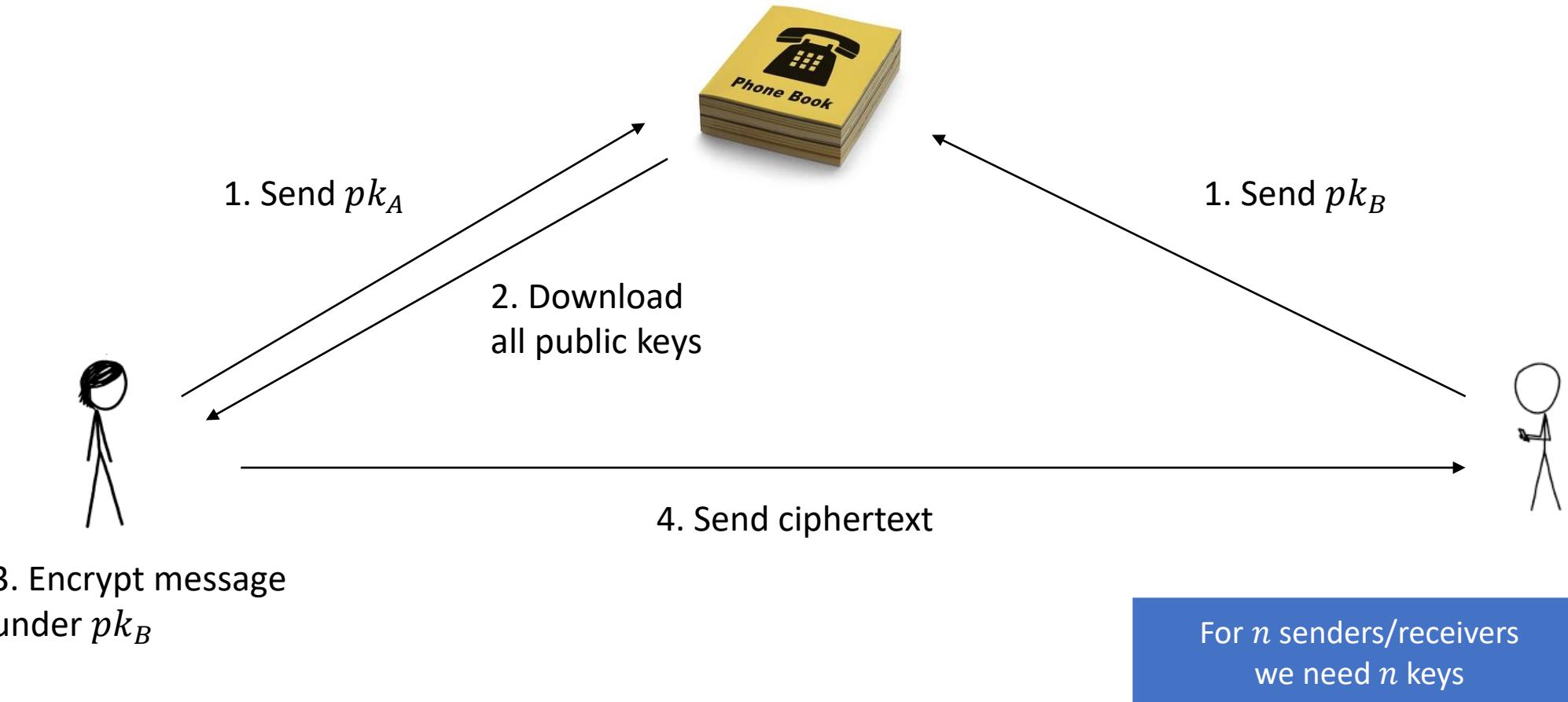
- One **private** and one **public**
- Given public key, it is hard to compute private key

## Confidentiality

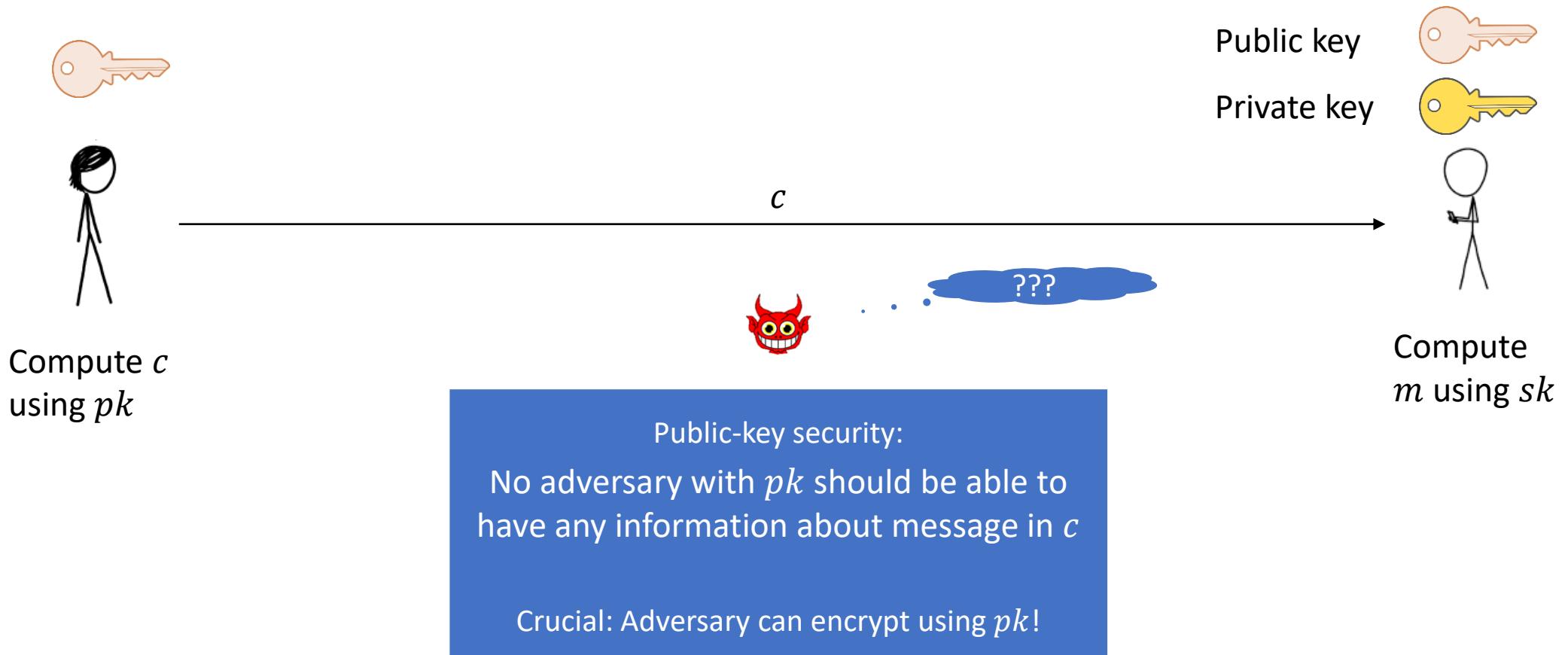
- The sender encrypts the message with the **public key** of the receiver
- Only receiver can decrypt it using private key



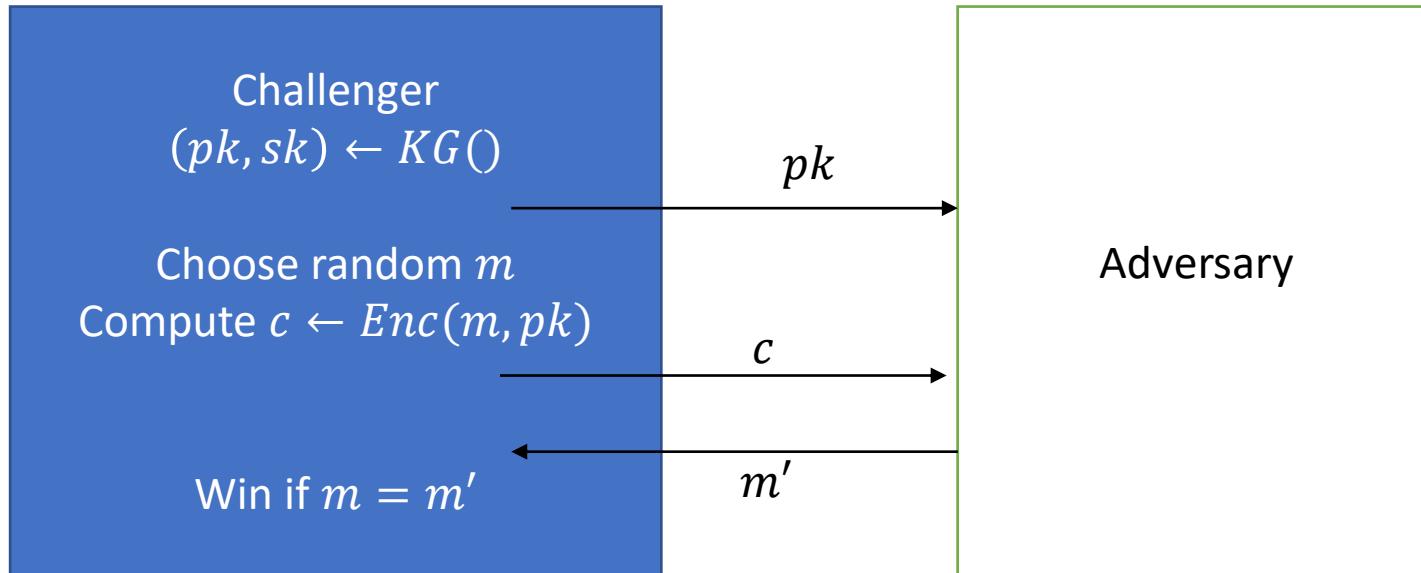
# How to use Public key encryption



# Public key encryption

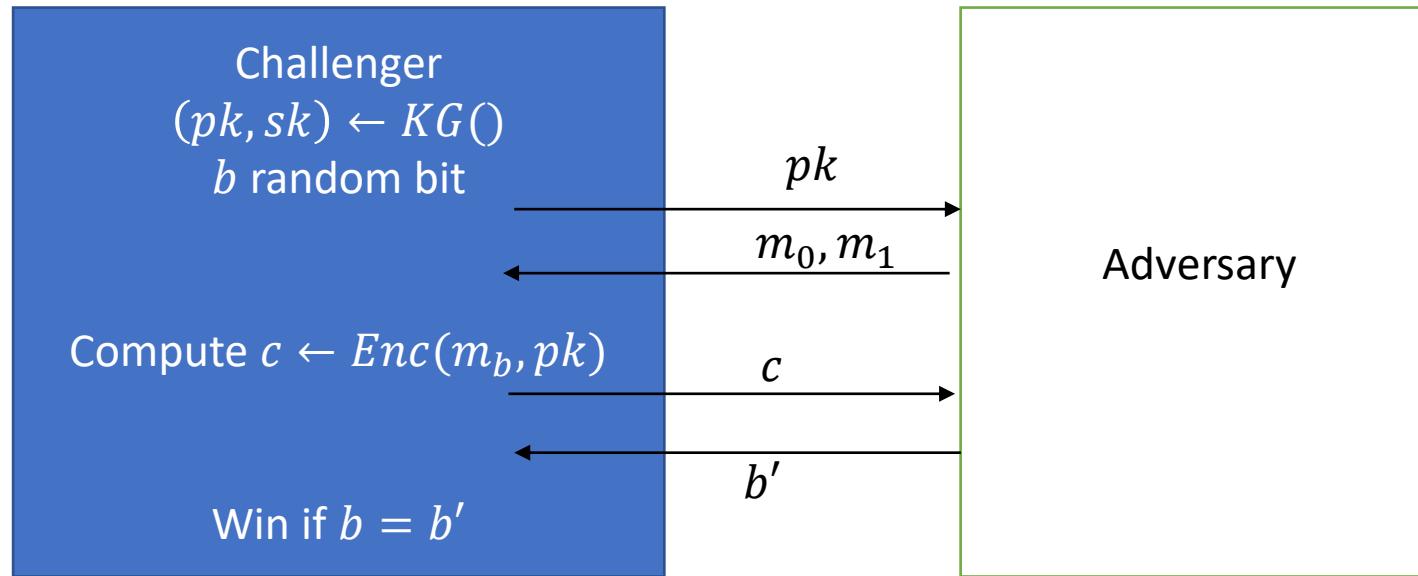


# Defining security of PKE

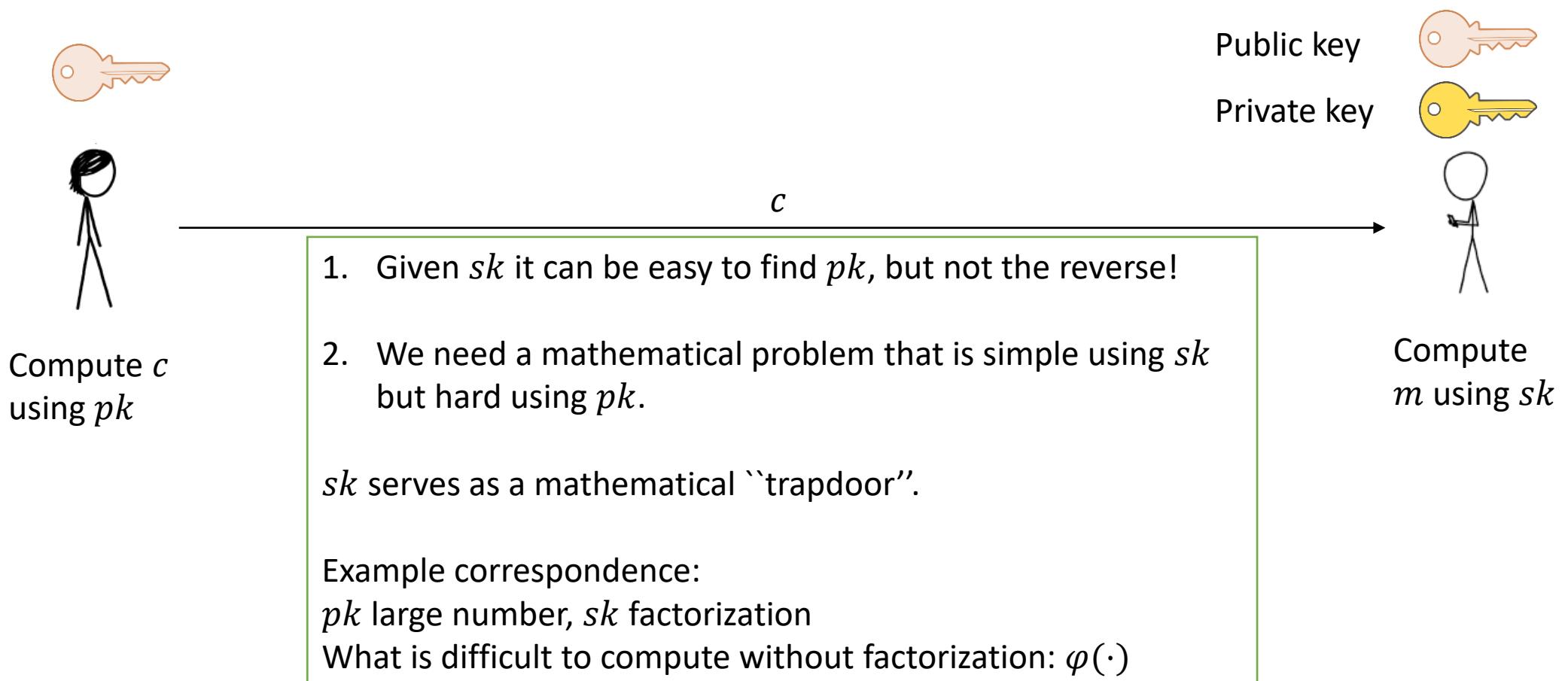


Problems with this definition?

# Defining security of PKE properly – IND-CPA



# Building Public key encryption



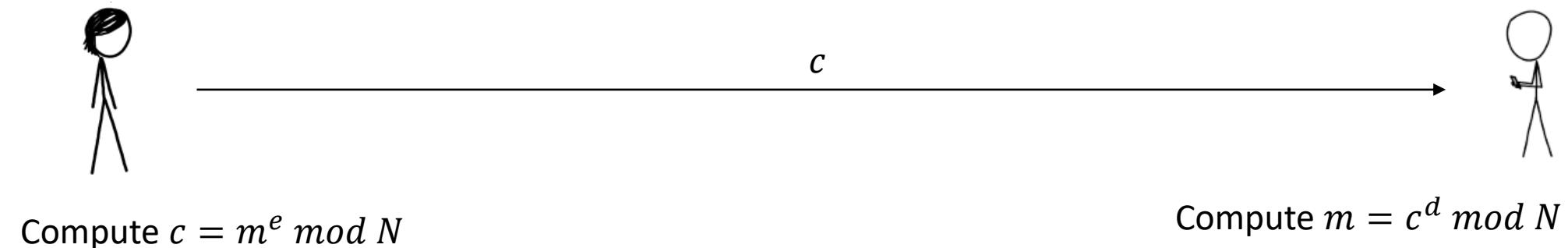
# The RSA cryptosystem

Invented 1977 by Rivest, Shamir & Adleman

Turing Award in 2002

## Key Generation

1. Find two large primes  $p, q$  and  $e$  with  $\gcd(e, (p - 1) \cdot (q - 1)) = 1$
2. Compute  $N = p \cdot q$
3. Find  $d$  such that  $d \cdot e = 1 \bmod (p - 1)(q - 1)$



# RSA – an example

## Key Generation

1. Find two large primes  $p = 13, q = 17$  and  $e = 5$ .  $\gcd(5, 12 \cdot 16) = 1$
2. Compute  $N = p \cdot q = 221$
3. Find  $d$  such that  $d \cdot e = 1 \bmod (p-1)(q-1)$   
Solve  $5d = 1 \bmod 192 \rightarrow d = 77$
4.  $pk = (N, e) = (221, 5), sk = (d) = (77)$

## Encrypt:

1. Message  $m \in \mathbb{Z}_N^*$ . Set  $m = 17$ .
2. Set  $c \leftarrow m^e \bmod N = 153$

## Decrypt:

1. Ciphertext  $c \in \mathbb{Z}_N^*$
2. Set  $m' = c^d \bmod N = 153^{77} \bmod 221 = 17$

# Why it works

- $N = p \cdot q, \varphi(N) = (p - 1) \cdot (q - 1)$
- We choose  $\gcd(e, (p - 1) \cdot (q - 1)) = 1$   
so  $d = e^{-1} \bmod \varphi(N)$  always exists
- Lagrange's Theorem:  $\forall x \in Z_N^*: x^{\varphi(N)} = 1 \bmod N$
- $Dec(Enc(x, pk), sk) = (x^e)^d \bmod N = x^{1+k\varphi(N)} = x \bmod N$

# Summary

We looked at the mathematics necessary for RSA

Modular arithmetic, computing modular inverses, Lagrange's theorem

Public Key encryption

1. What is Public Key Encryption?
2. Defining Security of PKE

The RSA cryptosystem

1. The RSA cryptosystem
2. Why RSA decryption works

# RSA and Digital Signatures

# Schedule for today

## Recap

## More on RSA

1. Making RSA IND-CPA/IND-CCA secure
2. Hybrid encryption

## Digital Signatures

1. What are Digital Signatures?
2. Formalizing security
3. Simple RSA signatures and why they are not secure
4. The RSA-FDH signature scheme
5. Proving RSA-FDH secure

# What we did last time



# Public Key Encryption

*True or false?*

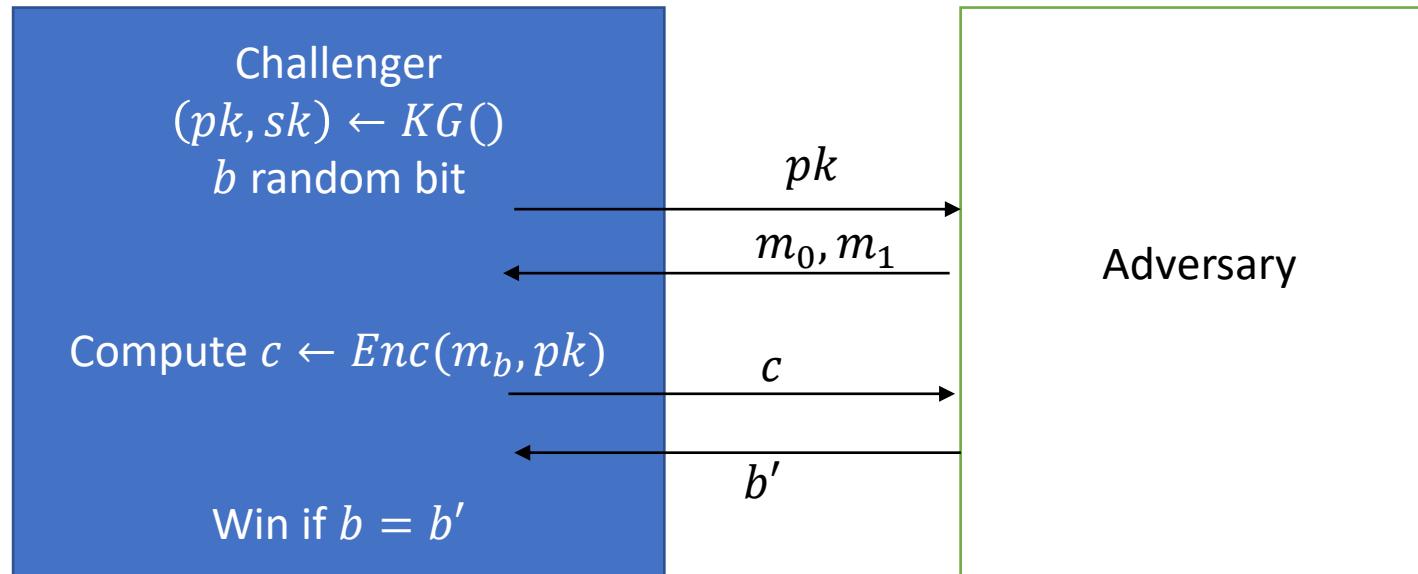
Public Key Encryption allows any two parties to confidentially communicate with each other, without knowing anything about each other.

# Security of PKE

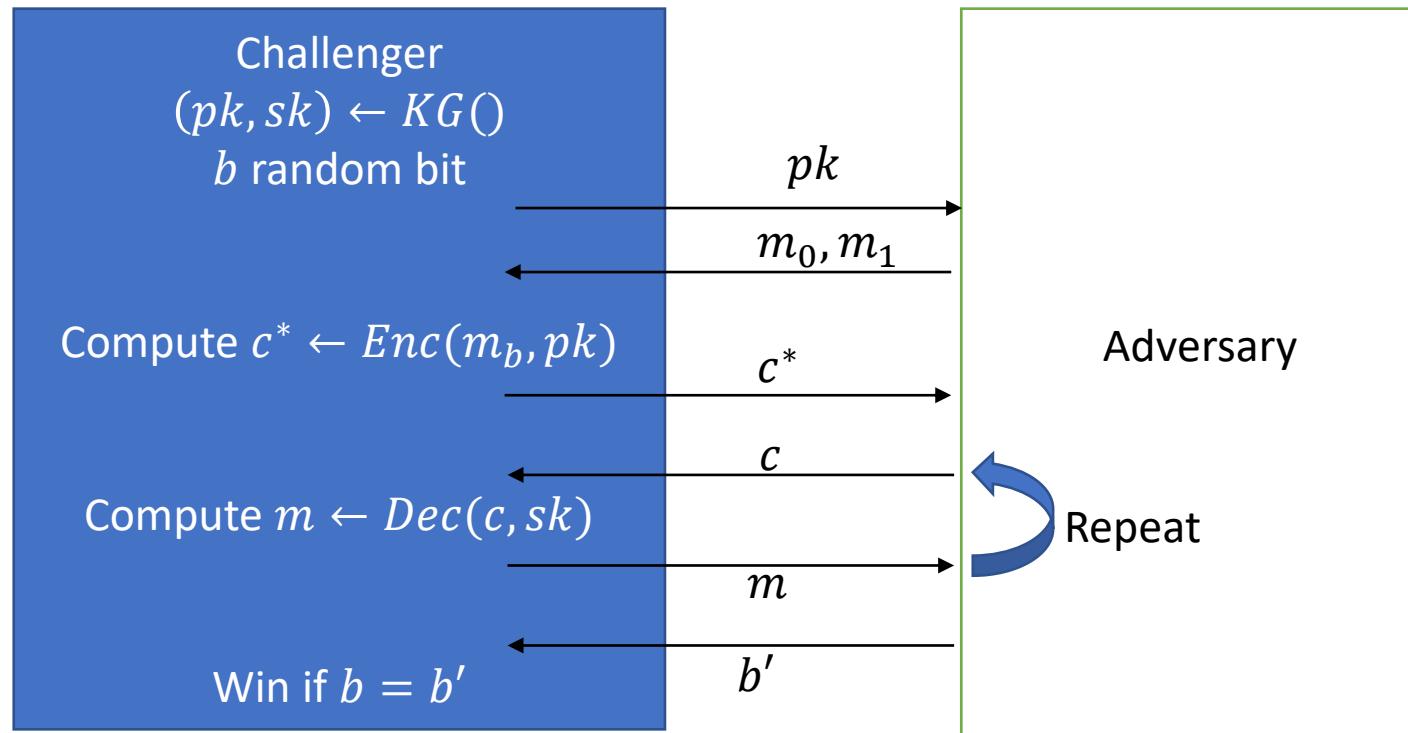
*True or False?*

IND-CPA is the strongest security notion for Public Key encryption schemes we know of.

# Defining security of PKE – IND-CPA



# Defining security of PKE – IND-CCA



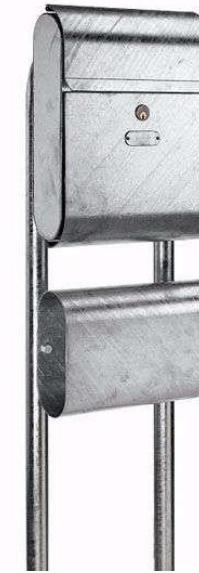
# RSA

*Which statement/statements is/are true?*

RSA is secure if

1. It is hard to compute  $e$ 'th roots modulo a biprime  $N$ .
2. It is hard to compute  $e$ 'th powers modulo a biprime  $N$ .
3. It is hard to factor a number  $N$  into its prime factors.

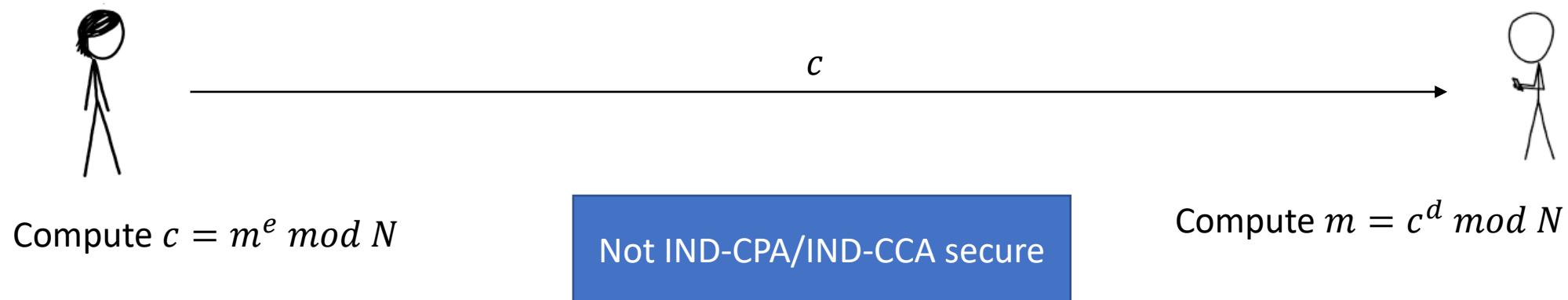
# More about RSA



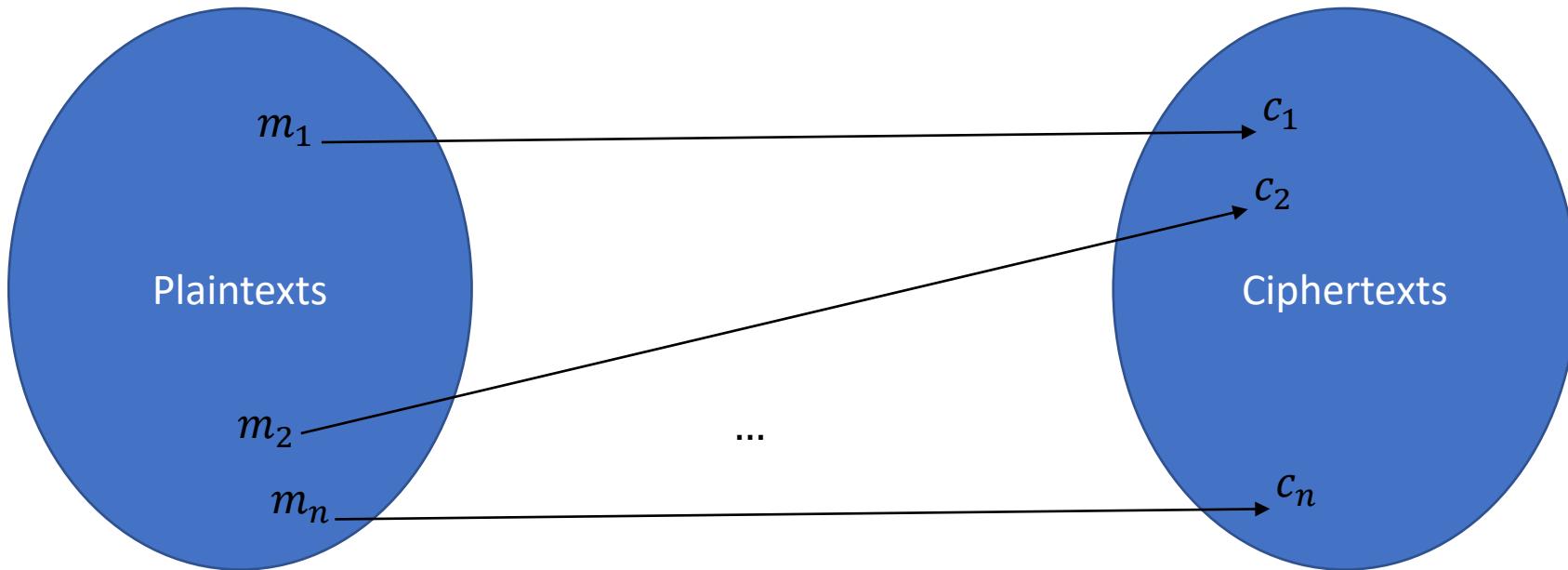
# The RSA cryptosystem

## Key Generation

1. Find two large primes  $p, q$  and  $e$  with  $\gcd(e, (p - 1) \cdot (q - 1)) = 1$
2. Compute  $N = p \cdot q$
3. Find  $d$  such that  $d \cdot e = 1 \bmod (p - 1)(q - 1)$

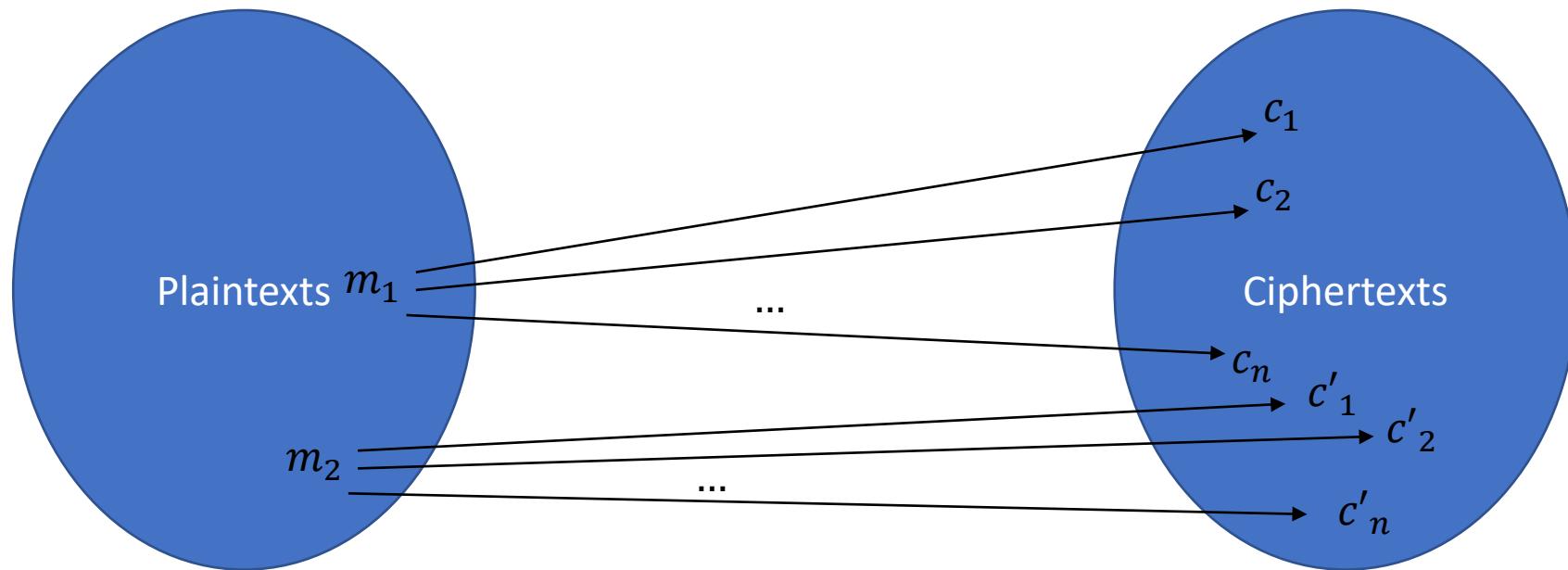


# Core of the problem for IND-CPA: no randomness

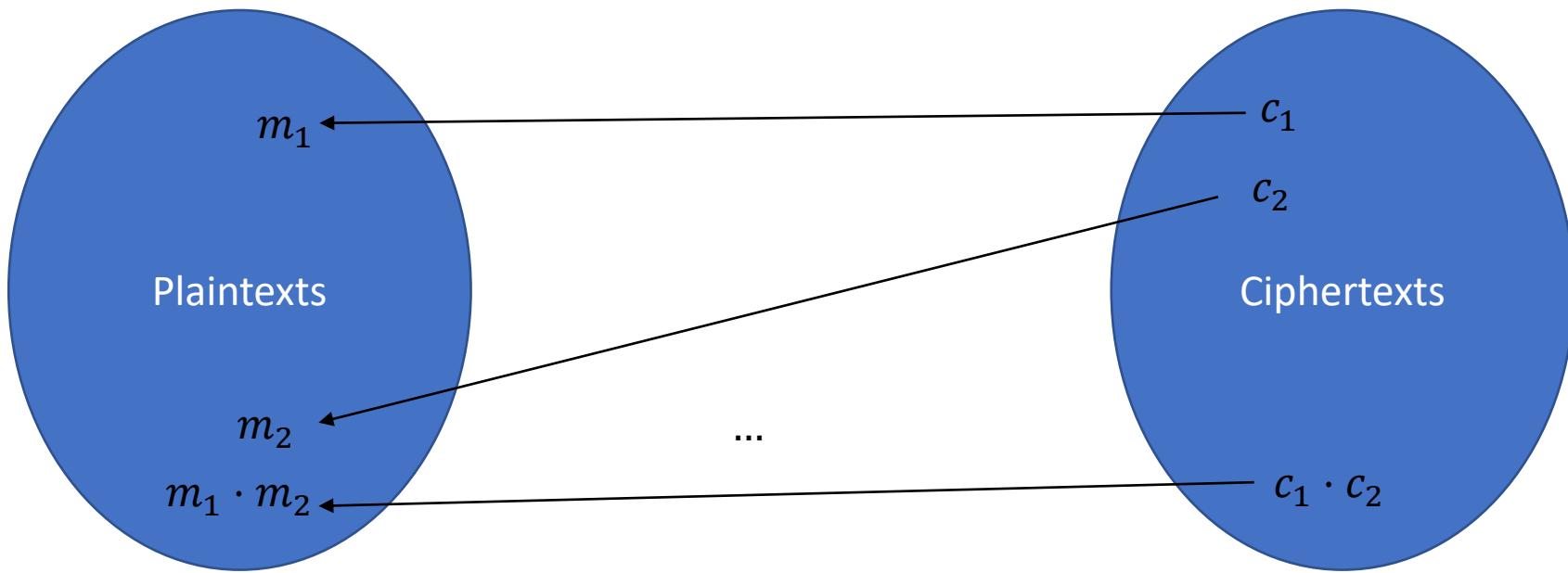


As many messages as ciphertexts, encryption/decryption is bijection

# Solving the IND-CPA problem

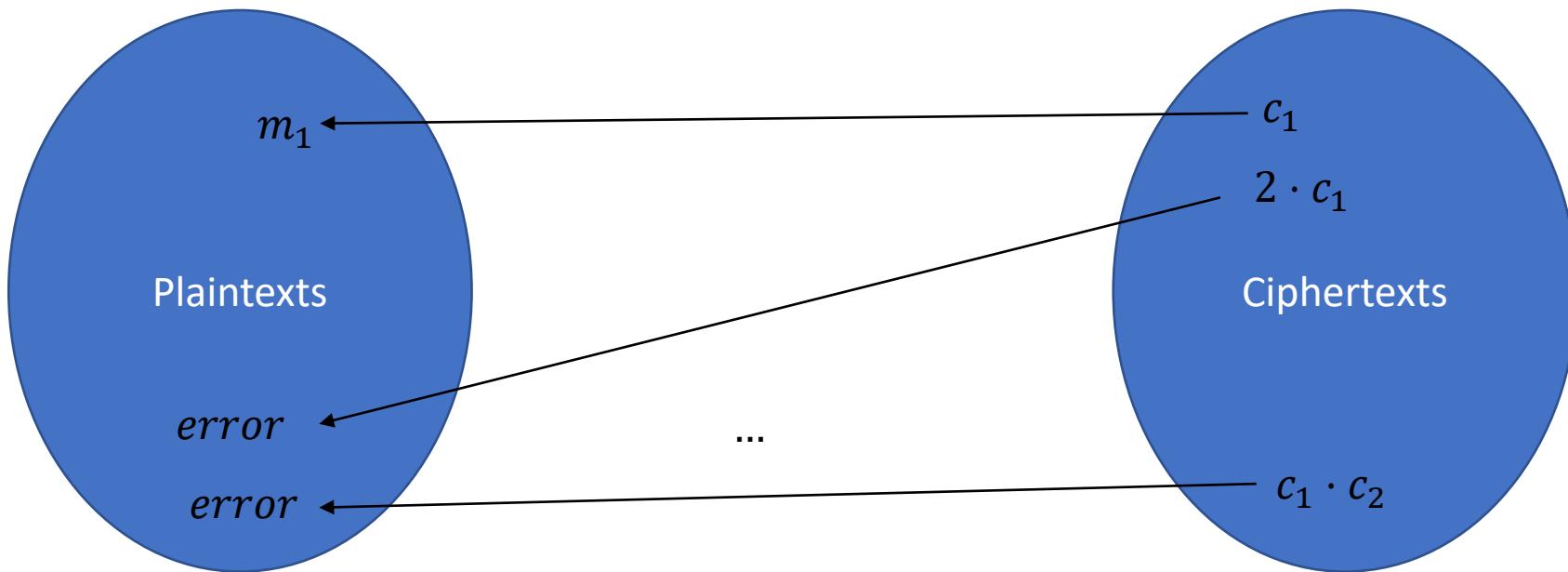


# Core of problem for IND-CCA: Malleable ciphertexts



In RSA we can change ciphertexts such that their plaintexts change in a predictable way

# Solving the IND-CCA problem



# RSA-OAEP

Let  $k = 8 \cdot \lceil \log_8 N \rceil$  and  $k_0, k_1 > 128, n = k - k_0 - k_1$

Messages  $m \in \{0,1\}^n$

Hash functions  $G: \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}, H: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$

Encryption for  $m, N, e$ :

1. Sample random bit string  $R \in \{0,1\}^{k_0}$
2. Compute  $A = [(m|0^{k_1}) \oplus G(R), R \oplus H((m|0^{k_1}) \oplus G(R))]$
3. Set  $c = A^e \bmod N$

# Decrypt RSA-OAEP

Messages  $m \in \{0,1\}^n$

Hash functions  $G: \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$ ,  $H: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$

Format:  $[(m|0^{k_1}) \oplus G(R), R \oplus H((m|0^{k_1}) \oplus G(R))]$

Decryption for  $c, N, d$ :

1. Compute  $A' = c^d \bmod N$  and check if  $A' < 2^k$
2. Let  $A' = [B_0, B_1]$  and compute  $R' = H(B_0) \oplus B_1$
3. Compute  $m' = B_0 \oplus G(R')$ . If  $m'$  ends with  $k_1$  0s then recover  $m$

# Why RSA-OAEP works

Messages  $m \in \{0,1\}^n$

Hash functions  $G: \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$ ,  $H: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$

Format:  $A = [(m|0^{k_1}) \oplus G(R), R \oplus H((m|0^{k_1}) \oplus G(R))]$

IND-CPA: every choice of  $R, m$  gives different  $A$  ( $2^{k_0}$  many)

# Why RSA-OAEP works

Messages  $m \in \{0,1\}^n$

Hash functions  $G: \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$ ,  $H: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$

*Format:*  $A = [(m|0^{k_1}) \oplus G(R), R \oplus H((m|0^{k_1}) \oplus G(R))]$



Changing 1 bit in  $m$  or  $R$  creates  
entirely different block

Related messages don't have an algebraic relation!

# Hybrid encryption

AES vs. RSA on a modern AMD Ryzen 9 5950X from 2020. 16 cores but we only use 1.

## AES-128

1. It takes around 16 cycles per byte (<https://bench.cr.yp.to/results-stream.html>) to encrypt/decrypt AES-128. For a whole ciphertext of 16 bytes, that is around 256 cycles.
2. The processor runs at 3.400 MHz, i.e. it performs 3.400.000.000 cycles per second.
3. One core can approximately encrypt/decrypt 13.300.000 ciphertexts per second.

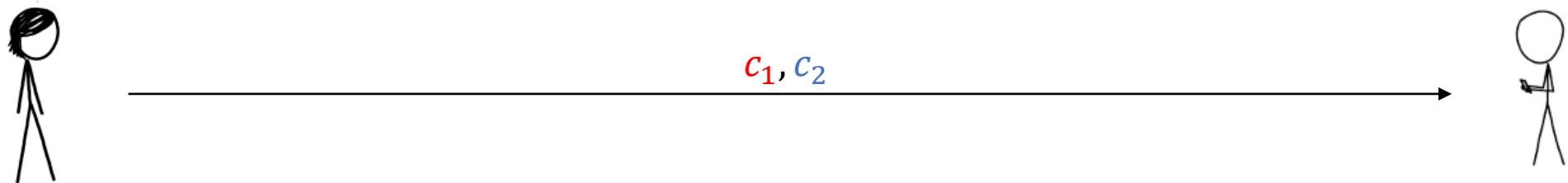
## RSA w/ 2048 bit keys

1. Encryption with small exponent:  $\approx 12.000$  cycles
  2. Decryption:  $\approx 2.400.000$  cycles
- (from <https://bench.cr.yp.to/results-kem.html> )



# Hybrid encryption (key encapsulation)

Use PKE scheme  $Enc^{Pub}, Dec^{Pub}$   
together with SKE scheme  $Enc^{Sym}, Dec^{Sym}$



1. Choose symmetric key  $k$ .
2. Compute  $c_1 \leftarrow Enc^{Pub}(k, pk)$ .
3. Compute  $c_2 \leftarrow Enc^{Sym}(m, k)$ .

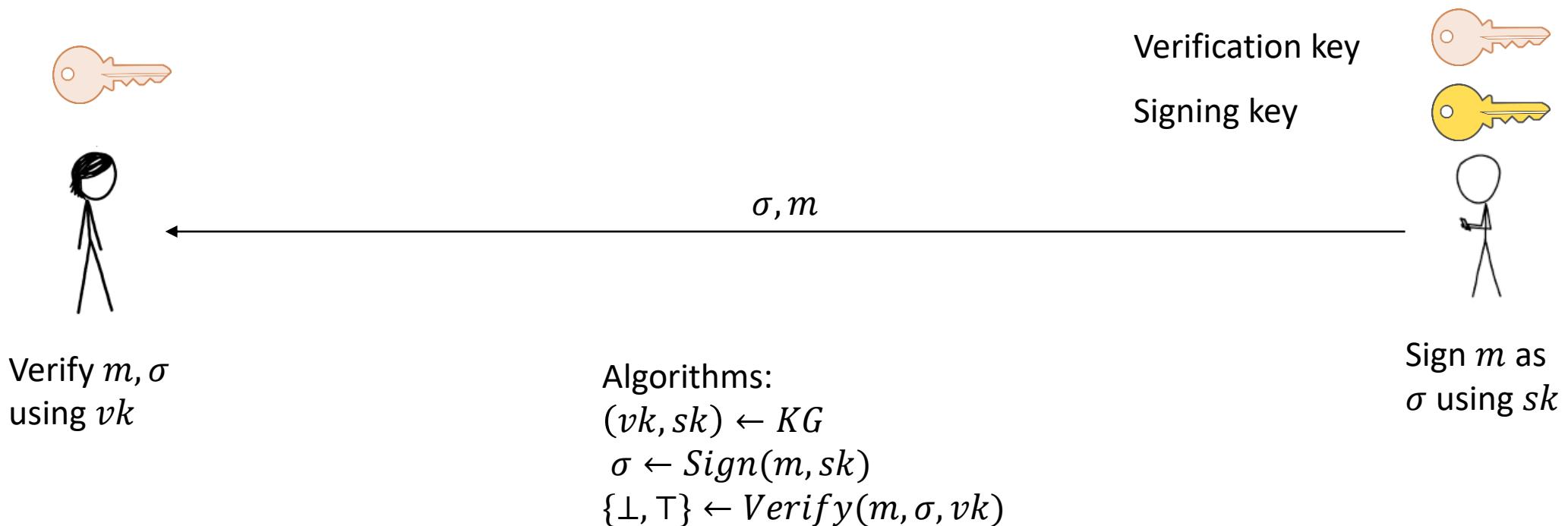
1. Recover  $k \leftarrow Dec^{Pub}(c_1, sk)$ .
2. Decrypt  $m \leftarrow Dec^{Sym}(c_2, k)$ .

# Digital Signatures

---



# Digital Signatures



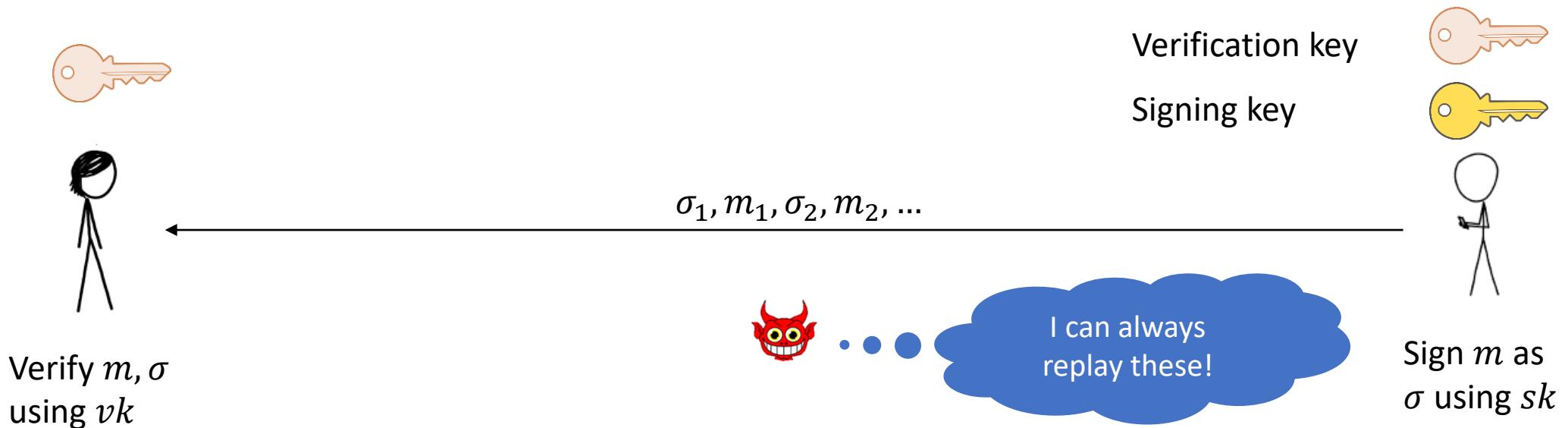
# Use cases of digital signatures

- ``digital'' equivalent of signing a contract (NemID/MitID)
- Building authenticated channels over insecure network
- Software integrity
- Transactions in cryptocurrencies



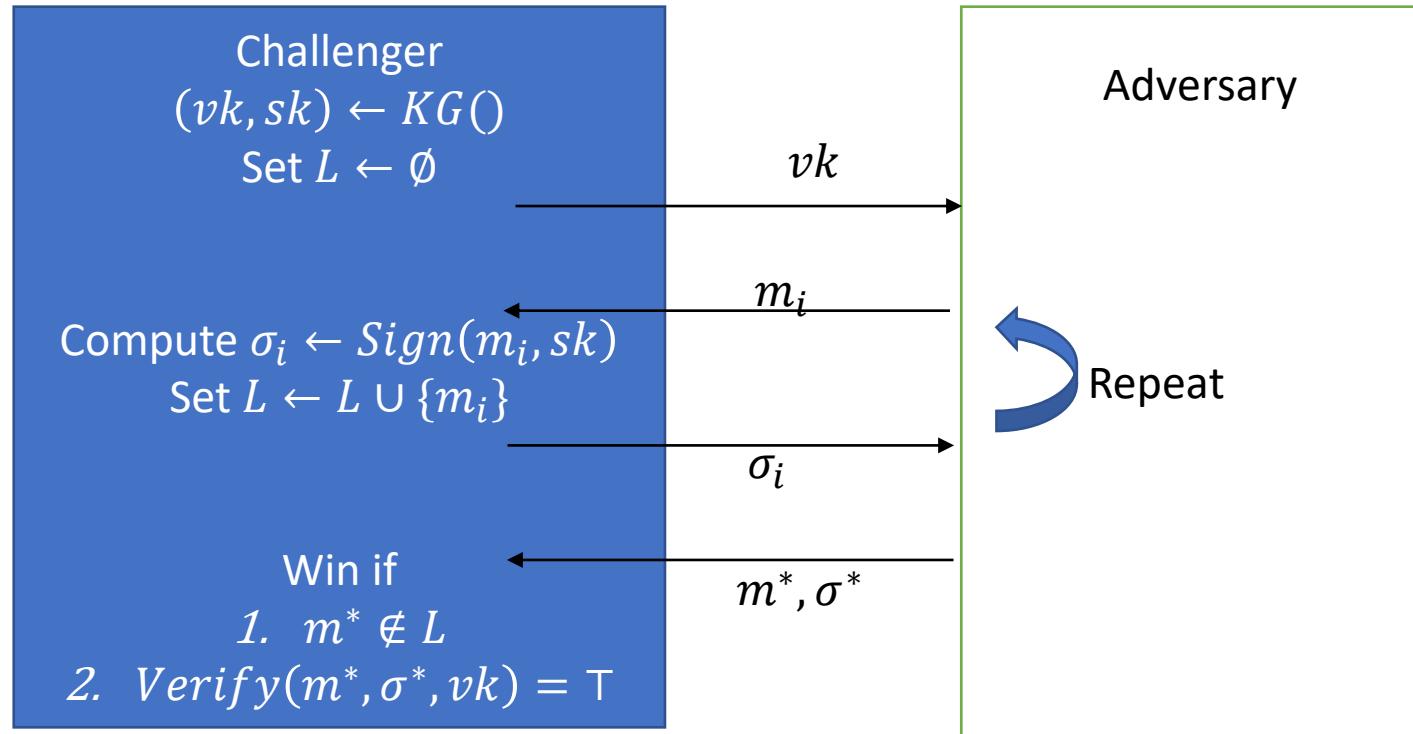
# Defining Security

MACs for public key setting!



Unforgeability:  
No adversary with  $vk$  and  
message/signature pairs  $m_1, \sigma_1, \dots$   
should be able to make new  $m, \sigma$

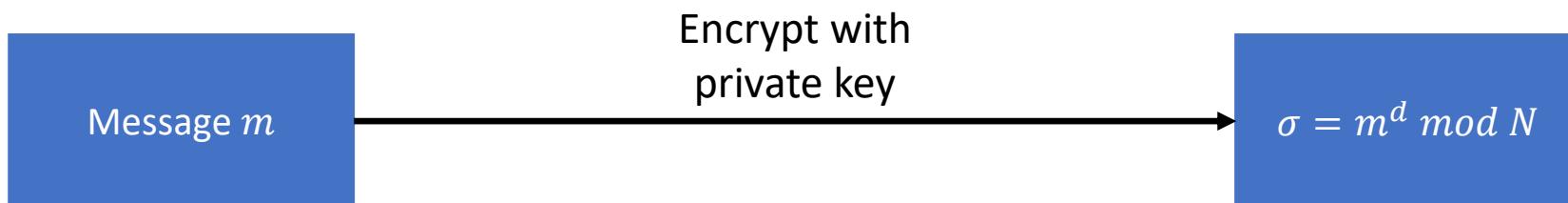
# EUF-CMA for Signatures



# Signatures from RSA: the wrong way

Signing key: secret  $d$

Verification key:  $N, e$



Verify  $m, \sigma$ :  
Check that  $m = \sigma^e \bmod N$

This clearly fails!

# Counterexample 1

Generate signature on “random” message:

1. Let  $\text{pk} = (N, e)$
2. Fix a random element  $\sigma \in Z_N^*$
3. Compute  $m = \sigma^e \bmod N$

$(m, \sigma)$  is valid by construction

# Counterexample 2 – Inspired by Homework 2

We want to forge a signature on  $m$

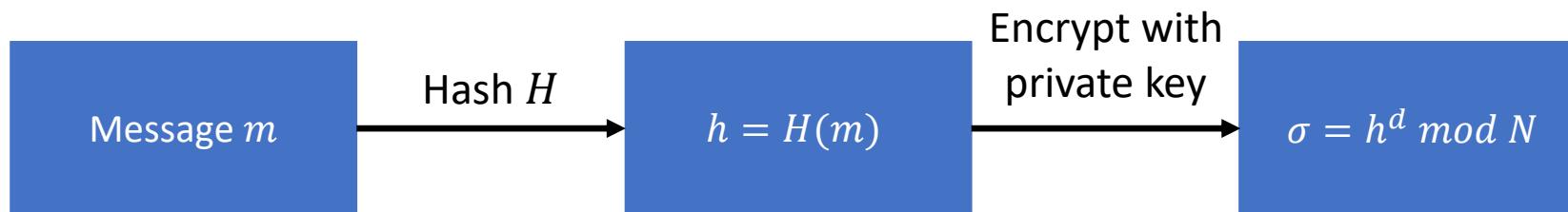
1. Choose  $m_1 \in \mathbb{Z}_N^*$ , compute  $m_2 \leftarrow \frac{m}{m_1} \bmod N$
2. Ask EUF-CMA oracle to compute  $\sigma_1 \leftarrow \text{Sign}(m_1, sk), \sigma_2 \leftarrow \text{Sign}(m_2, sk)$
3. Then  $\sigma = \sigma_1 \cdot \sigma_2 = m_1^d \cdot m_2^d = m^d$  is a valid signature on  $m$ !

# Digital Signatures using RSA: RSA-FDH

Signing key: secret  $d$

Verification key  $N, e$

Cryptographic hash  $H: \{0,1\}^* \rightarrow \mathbb{Z}_N^*$

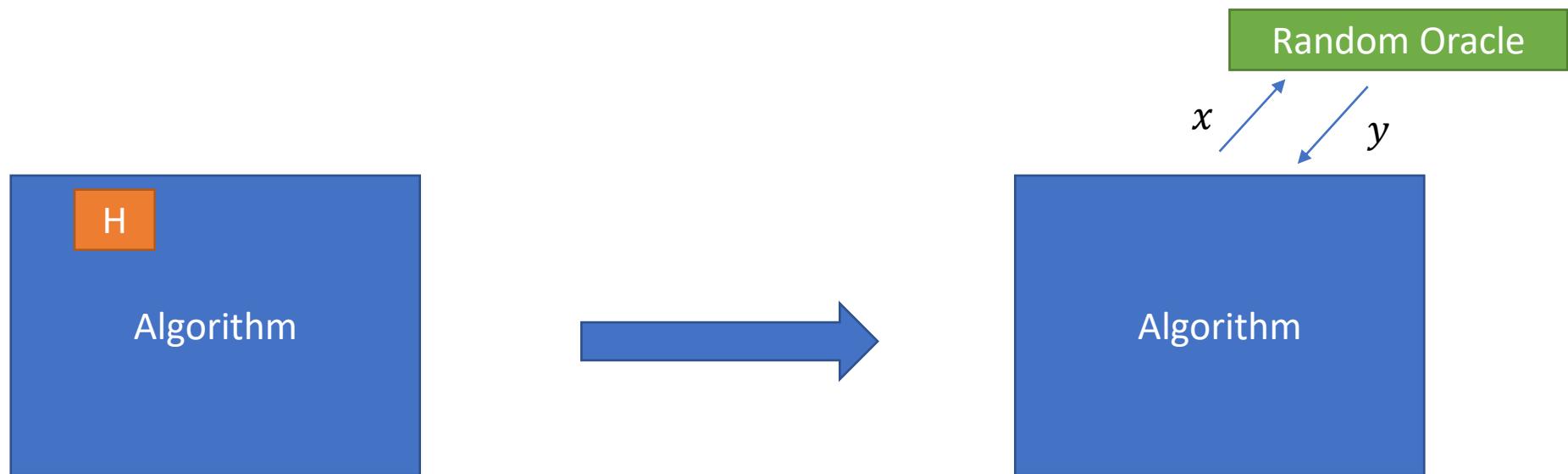


Verify  $m, \sigma$ :  
Check that  $H(m) = \sigma^e \bmod N$

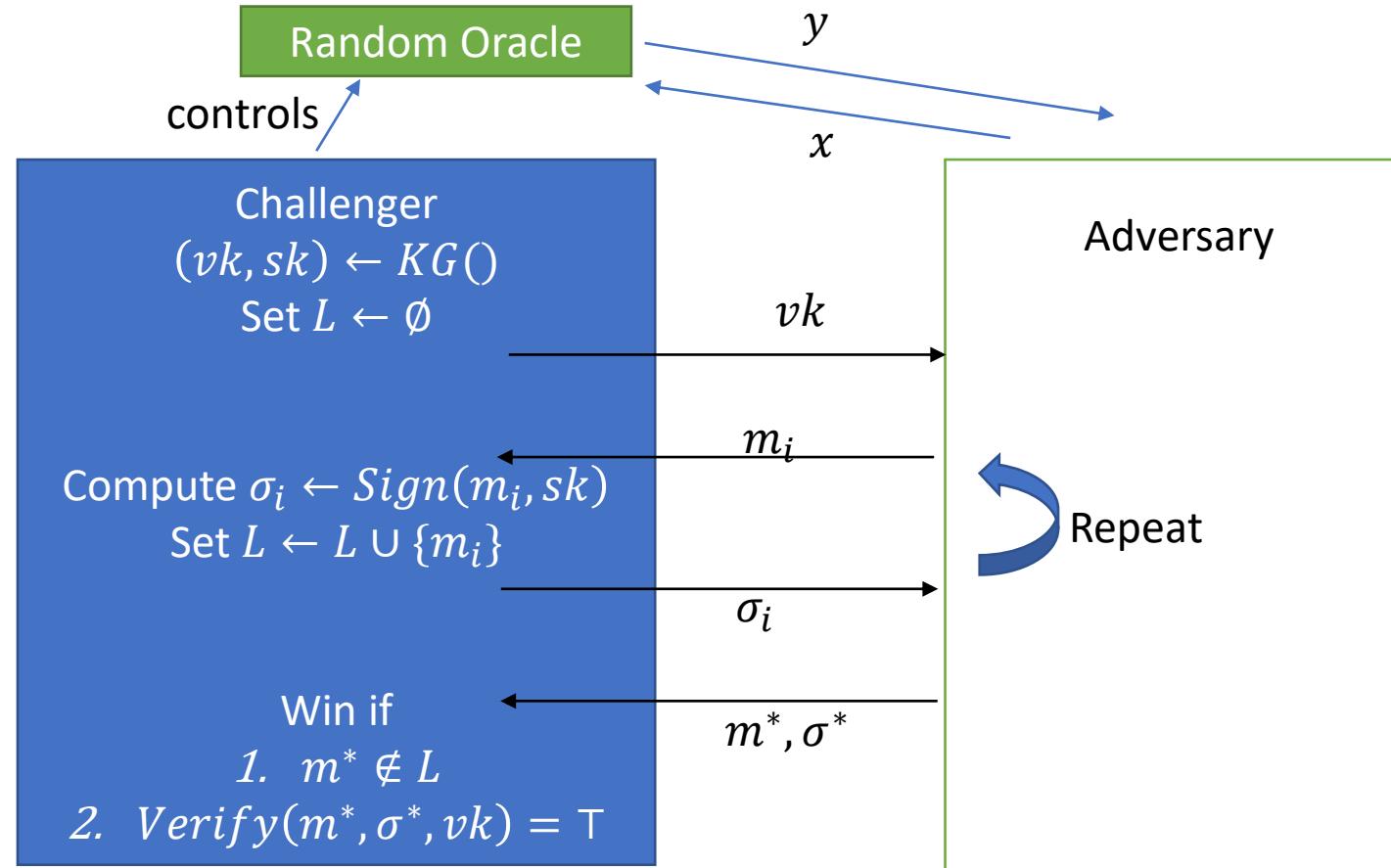
Any RSA instance for encryption can also be used for signing!

# EUF-CMA security

Recap from Problem Sheet 5: the Random Oracle Model

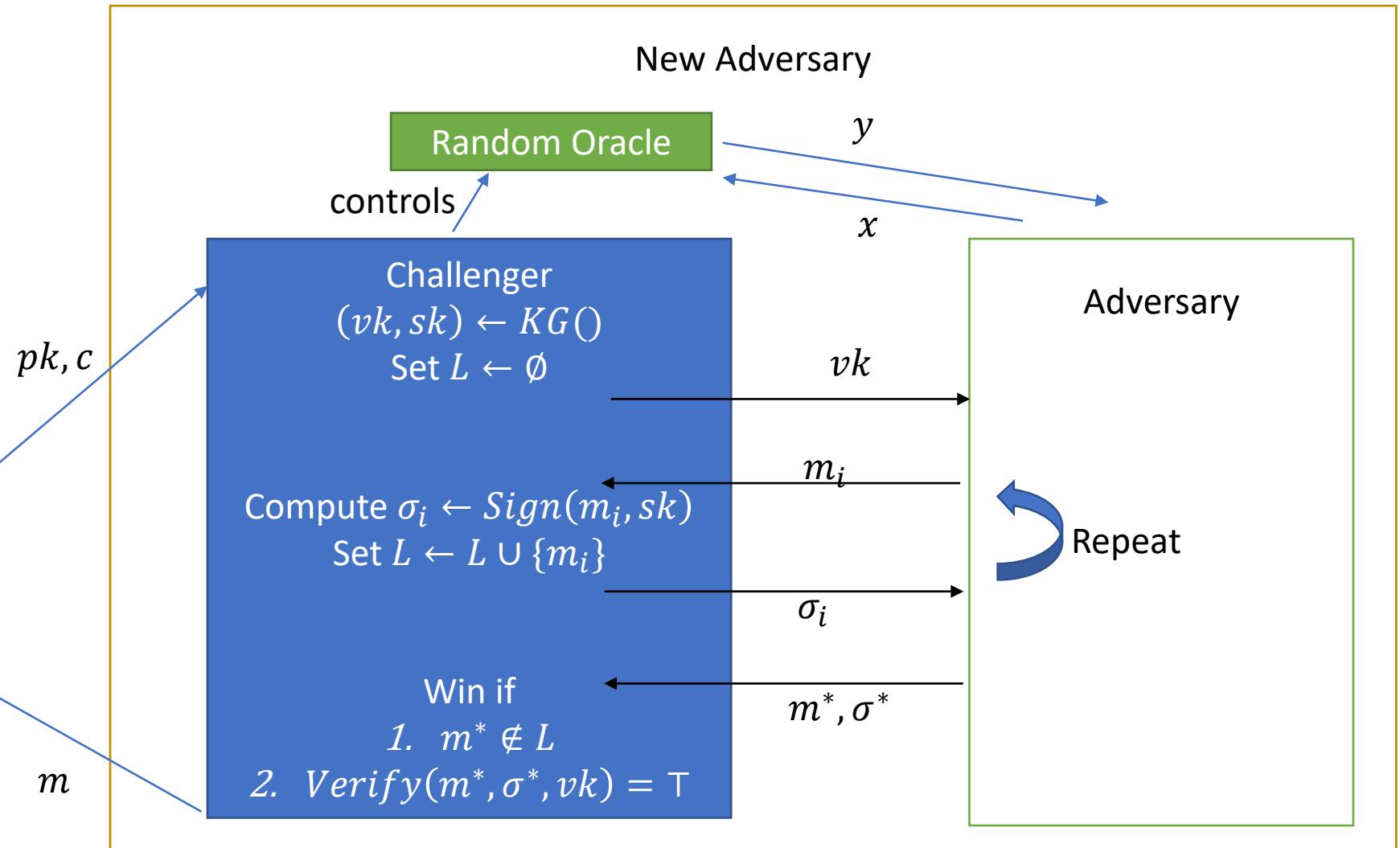
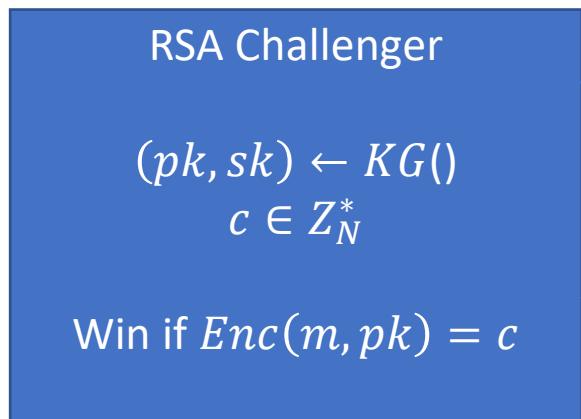


# Looking at EUF-CMA



# What we prove

Assuming  $H$  is a random oracle. Then given the RSA problem is hard (Problem Sheet 6), RSA-FDH is EUF-CMA secure.



# Summary

When using RSA, use RSA-OAEP to make it IND-CCA secure (16.2.1 in the book)

Hybrid encryption for long messages

Digital signatures

RSA signatures using RSA-FDH

# Digital Signatures and Primality Testing

# Schedule for today

## Recap

### Digital Signatures

1. The RSA-FDH signature scheme
2. Proving RSA-FDH secure

### Primality Testing

1. Prime numbers and the prime number theorem
2. Trial division
3. The Fermat test and Carmichael numbers
4. Miller-Rabin test
5. The AKS test

# What we did last time



# Question 1

RSA-OAEP only allows us to encrypt a message  $m$  substantially shorter than  $\log_2 N$ . Why is this unavoidable if we want IND-CPA security and correctness?

## Question 2

In the EUF-CMA security game, what if the attacker can come up with a fresh  $\sigma'$  on a message  $m$  that it has not seen before?

# Question 3

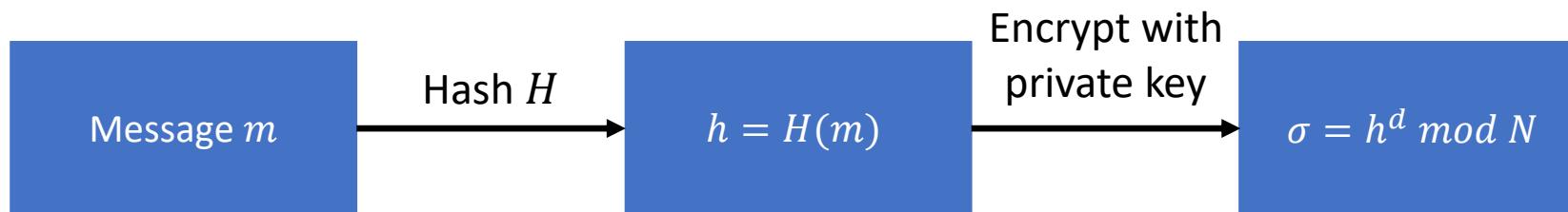
Could an attack where an attacker finds a new  $\sigma'$  for a previously queried  $m$  work for RSA-FDH?

# Digital Signatures using RSA: RSA-FDH

Signing key: secret  $d$

Verification key  $N, e$

Cryptographic hash  $H: \{0,1\}^* \rightarrow \mathbb{Z}_N^*$

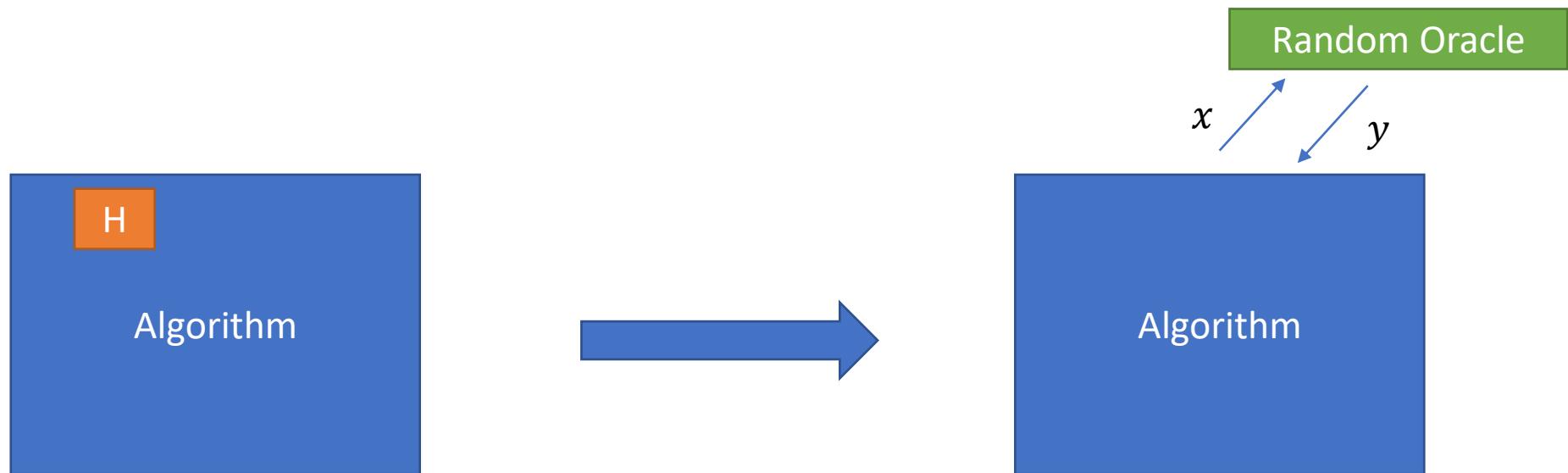


Verify  $m, \sigma$ :  
Check that  $H(m) = \sigma^e \bmod N$

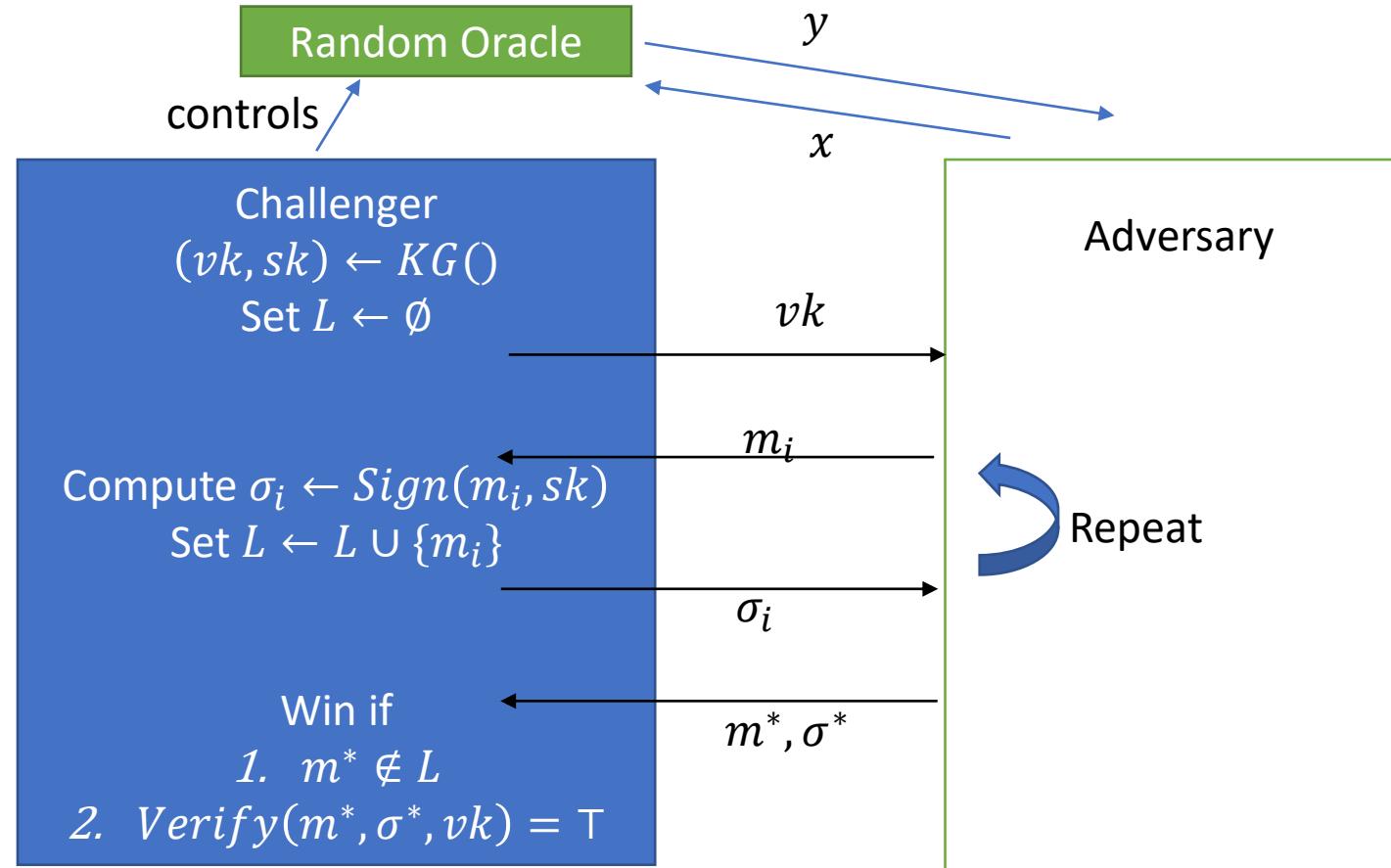
Any RSA instance for encryption can also be used for signing!

# EUF-CMA security

Recap from Problem Sheet 5: the Random Oracle Model

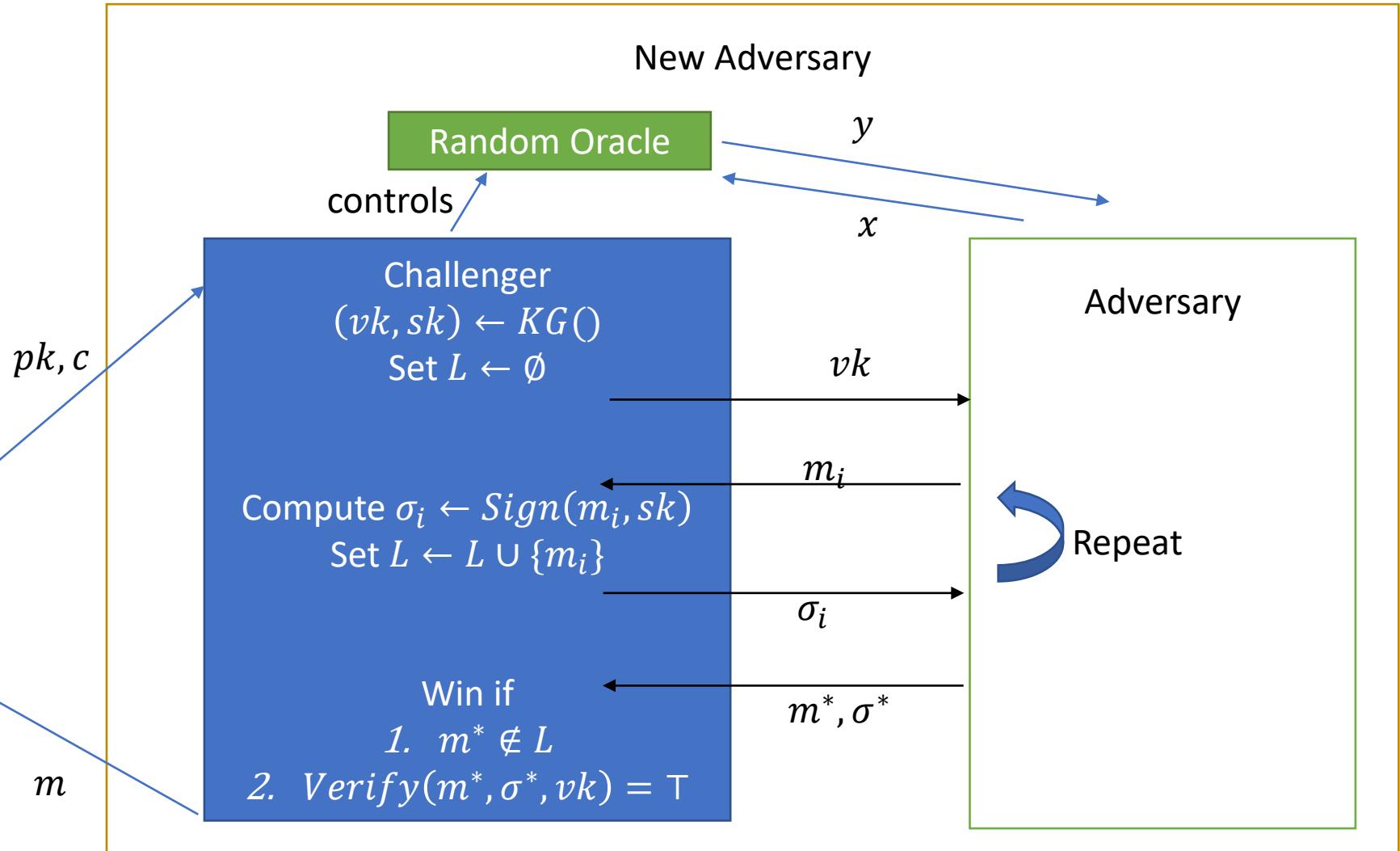
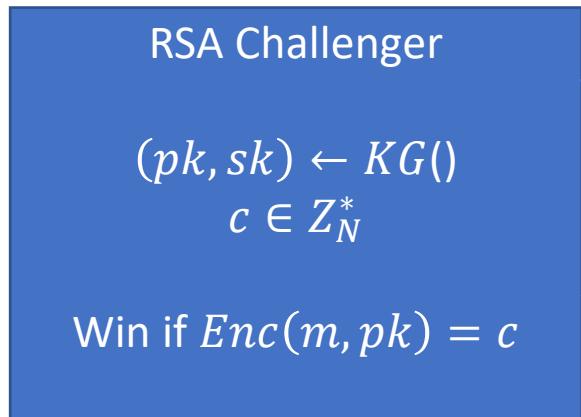


# Looking at EUF-CMA



# What we prove

Assuming  $H$  is a random oracle. Then given the RSA problem is hard (Problem Sheet 6), RSA-FDH is EUF-CMA secure.



# Proof

Blackboard ☺

# Primality testing

# How many prime numbers are there?

Let  $\pi(x) = |\{p \text{ prime} \mid p < x\}$ . Then  $\pi(x) \approx x/\ln(x)$

$x$	$x/\ln(x)$	$\pi(x)$
$10^3$	145	168
$10^4$	1,086	1,229
$10^5$	8,686	9,592
$10^6$	72,382	78,498
$10^7$	620,420	664,579

# How many prime numbers are there?

Let  $\pi(x) = |\{p \text{ prime} \mid p \leq x\}|$ . Then  $\pi(x) \approx x/\ln(x)$

Assuming the primes are equally distributed in interval,

$$\Pr[p \text{ prime}] \approx 1/\ln p$$

# How to check that $p$ is prime?

Idea 1:  $p$  prime iff only divisible by 1 and  $p$

Trial-division by all numbers  $k \in \{1, \dots, \sqrt{p}\}$

Why is  $\sqrt{p}$  sufficient?

## Runtime estimate

1. Assume trial division by  $k$  each is one unit of time
2.  $\sqrt{2^{1024}} = 2^{512}$  units of time needed
3. To break AES-128, we only need  $2^{128}$  operations...

# But!

Trial division is efficient for small numbers and to eradicate non-prime candidates early!

Any random number is divisible

1. by 2 with probability  $\frac{1}{2}$
2. by 3 with probability  $\frac{1}{3}$
3. by 5 with probability  $\frac{1}{5}$
4. ...

A random number is divisible by 2, 3 or 5 with probability 0.73

Use to sieve before using ``the big guns''

# Fermat's Test: idea

## Fermat's little theorem

For any prime  $p$ ,

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

More generally:  $a^{\phi(n)} = 1 \bmod n$  for  $a \in \mathbb{Z}_N^*$

Hope: if  $n$  not prime, then  $\phi(n) \neq n - 1$  and  
very often  $a^{n-1} \neq 1 \bmod n$

# Fermat's Test

The algorithm for input  $n$

1. For  $i \in \{1, \dots, k\}$ :
  1. Pick  $a \in \{2, \dots, n - 1\}$  uniformly at random
  2. Compute  $b = a^{n-1} \bmod n$
  3. If  $b \neq 1$  then output "Not prime"
2. Output "Probably prime"

How to choose  $k$ ?

What test shows: if  $a^{n-1} \neq 1 \bmod n$  then  $n$  not prime

What it doesn't show:  $n$  is prime

# Example

$n = 17$ :

- $3^{16} = 43046721 = 1 \bmod 17$
- $2^{16} = 65536 = 1 \bmod 17$

$n = 16$ :

- $2^{15} = 32768 = 0 \bmod 16$

# More examples

$n = 561 = 3 \cdot 11 \cdot 17$ :

- $5^{560} = 1 \bmod 561$
- $17^{560} = 1 \bmod 561$
- $235^{560} = 1 \bmod 561$

# Carmichael Numbers

A composite  $n$  such that  $\forall a \in \mathbb{Z}_n^*: a^{n-1} = 1 \bmod n$

Examples:

- 561
- 1105
- 1729
- 2465
- ...

Theorem (Erdos): There are infinitely many Carmichael numbers ☺

# Fixing Fermat's Test

Testing that  $a^{n-1} = 1 \bmod n$  is necessary, but not sufficient

Additional idea: roots of unity

$$x^2 - 1 = 0 \bmod n \leftrightarrow (x + 1)(x - 1) = 0 \bmod n$$

If  $n$  is prime then  $\pm 1$  are only roots of  $1 \bmod n$

# Fixing Fermat's Test

If  $n$  is odd, then  $n - 1 = 2^s d$  where  $d$  is odd

Consider  $a^d \bmod n, a^{2d} \bmod n, \dots, a^{2^s d} \bmod n$  for  $a \in \mathbb{Z}_n^*$ , then

- either  $a^d \equiv 1 \pmod n$
- or  $a^{2^i d} \equiv -1 \pmod n$

i.e. it cannot be that  $a^{2^j d} \notin \{-1, 1\} \bmod n$  but  $a^{2^{j+1} d} \equiv 1 \pmod n$

# Miller-Rabin Test

The algorithm for input  $n$

1. Let  $n - 1 = 2^s d$  where  $d$  is odd
2. For  $i \in \{1, \dots, k\}$ :
  1. Pick  $a \in \{2, \dots, n - 1\}$  uniformly at random
  2. Compute  $b = a^d \bmod n$
  3. If  $b \notin \{-1, 1\}$ 
    1. Set  $i \leftarrow 1$
    2. While  $i < s$  and  $b \neq -1$ 
      1.  $b \leftarrow b^2 \bmod n$
      2. If  $b = 1$  return "Composite"
      3.  $i \leftarrow i + 1$
    3. If  $b \neq -1$  return "Composite"
  3. Output "Probably prime"

# Can we fool Miller-Rabin?

Short answer: No!

Less short answer

For every composite  $n$  there exist more than 2 roots of unity, which the test may choose!

Full answer

If  $n$  is composite, then  $\geq 3/4$  of all  $a$  will make the test detect a composite! (e.g. <https://shoup.net/ntb/ntb-v2.pdf> Theorem 10.3)

# Certificates of primality

Trial division: none

Fermat: well, Carmichael numbers...

Miller-Rabin: if  $a_i$  truly random, then yes\*!

\*repeating the test  $k$  times gives failure  $\frac{1}{2^{2k}}$

# Deterministic Poly-Time test of Primality

Long-standing open question: can we get exact primality test in polynomial time?

Agrawal, Kayal, Saxena 2002: YES!

Their approach:  $n \geq 2$  is prime iff  $(X - a)^n = X^n - a \bmod n$  for some integer  $a$  coprime to  $n$

Their algorithm is accurate, but in practice slower than Miller-Rabin.

# Summary

1. RSA-FDH is EUF-CMA secure
2. The Fermat primality test can be fooled
3. Miller-Rabin is more reliable

# Discrete Logarithms

# Schedule for today

## Recap

### Discrete Logarithms & Key Agreement

The DLOG problem

Key Agreement

DH Key Agreement

### Security of Diffie Hellman

The DDH problems and relation to DLOG

Passive Security of DHKA

MITM attacks on DH

Generalizing DH & Elliptic Curve Groups

# What we did last time



# Questions

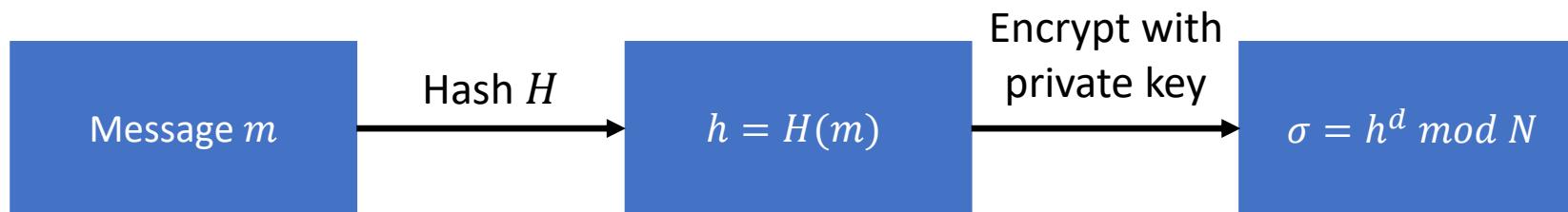
The RSA-FDH signature scheme allows multiple signatures for the same message.

# Digital Signatures using RSA: RSA-FDH

Signing key: secret  $d$

Verification key  $N, e$

Cryptographic hash  $H: \{0,1\}^* \rightarrow \mathbb{Z}_N^*$



Verify  $m, \sigma$ :  
Check that  $H(m) = \sigma^e \bmod N$

Any RSA instance for encryption can also be used for signing!

# Questions

There exist prime numbers  $p$  for which the Fermat test falsely claims that they are not prime.

# Questions

The Miller-Rabin test only probabilistically determines if a number is prime (i.e. with high probability the output is correct).

# Discrete Logarithms & Key Agreement

# Order of a group element

Let  $p$  be a prime and  $g \in Z_p^*$ .

Then the smallest positive  $a \in N$  such that  $g^a = 1 \bmod p$  is called *order of  $g \bmod p$* .

E.g.  $p = 31$ .

Element	1	2	3	4
Order $\bmod p$	1	5	30	5

# Lagrange's Theorem

For any finite group  $G$  and subgroup  $H \subseteq G$ :  $|H|$  divides  $|G|$ .

Corollary:

For any prime  $p$  and  $g \in Z_p^*$  the order of  $g$  must divide  $\phi(p) = p - 1$

Fact:

For any prime  $p$  there are  $\phi(\phi(p))$  many elements  $g \in Z_p^*$  that have maximal order  $\phi(p) = p - 1$ .

# Finding elements of maximal order

## Easy mode:

Let  $p$  be a prime such that  $p - 1 = 2q$  where  $q$  is also prime.

1. Pick  $g \in \mathbb{Z}_p^*$  such that  $g \notin \{1, p - 1\}$
2. If  $g^q \neq 1 \pmod{p}$  then output  $g$ , otherwise go to step 1

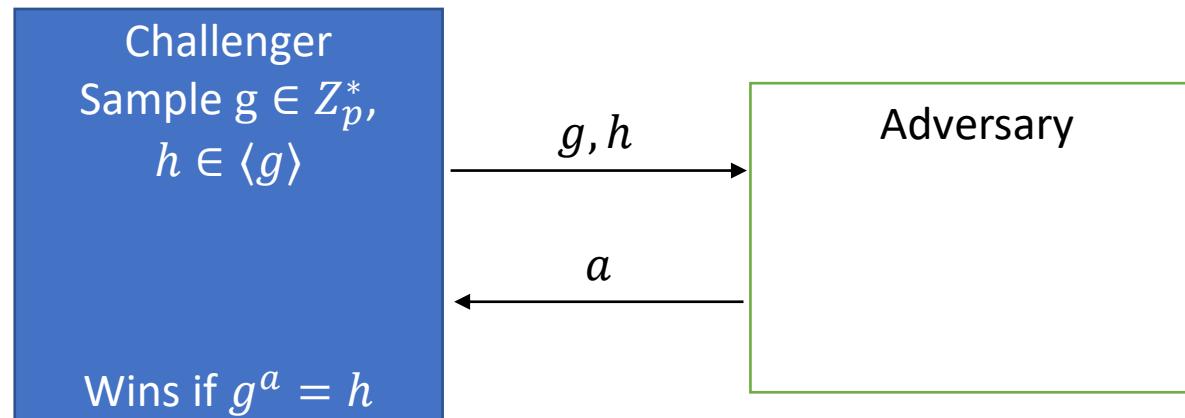
## General:

Let  $p$  be any prime.

1. Pick  $g \in \mathbb{Z}_p^*$ .
2. For every  $q$  that divides  $p - 1$  test that  $g^q \neq 1 \pmod{p}$ . If so then output  $g$ , otherwise go to step 1

# Discrete logarithms modulo a prime

Let  $p$  be a prime



# Discrete logarithms modulo a prime

Example

$$p = 17, g = 3, h = 14$$

$a$	1	2	3	4	5	6	7	8	9
$3^a \text{ mod } 17$	3	9	10	13	5	15	11	16	14

# Hardness of DLOG

When is DLOG difficult?

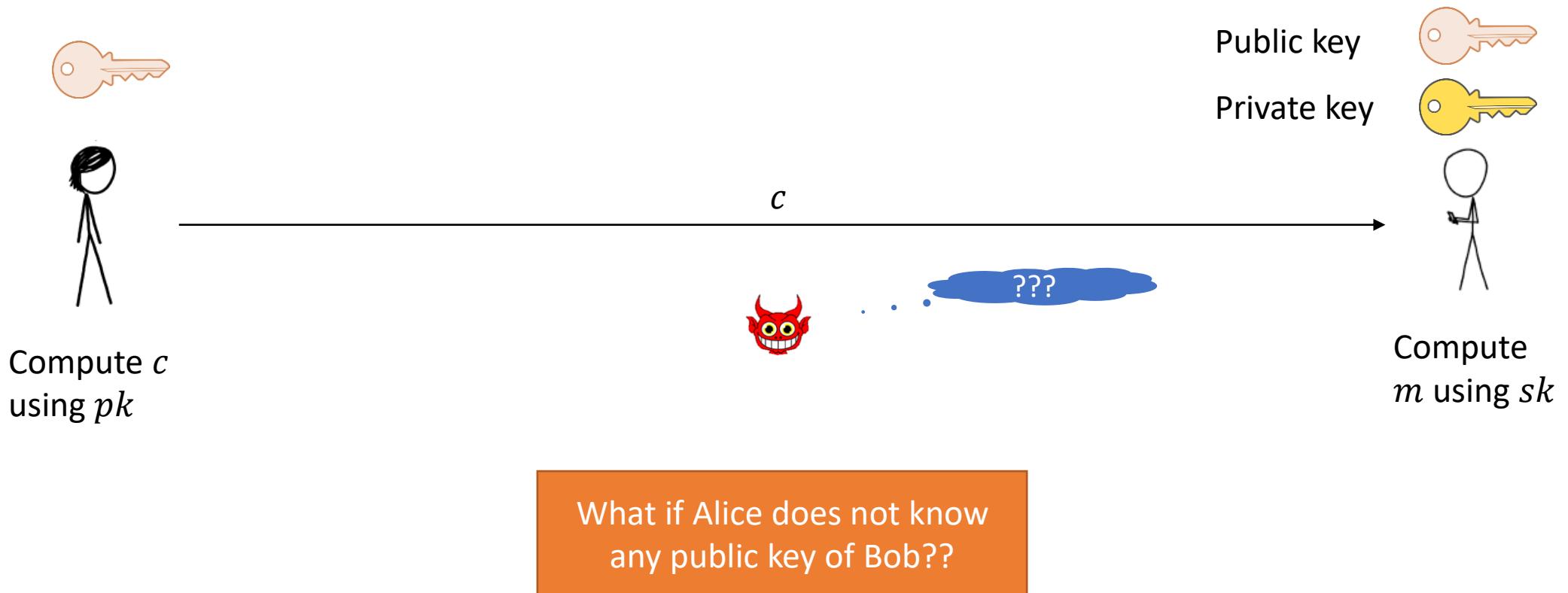
1.  $p$  is large prime (brute force attacks)
2.  $p - 1$  must have at least one large prime factor (exercise!)
3.  $q$  must be large (brute force attacks)

Difficulty until 2030:

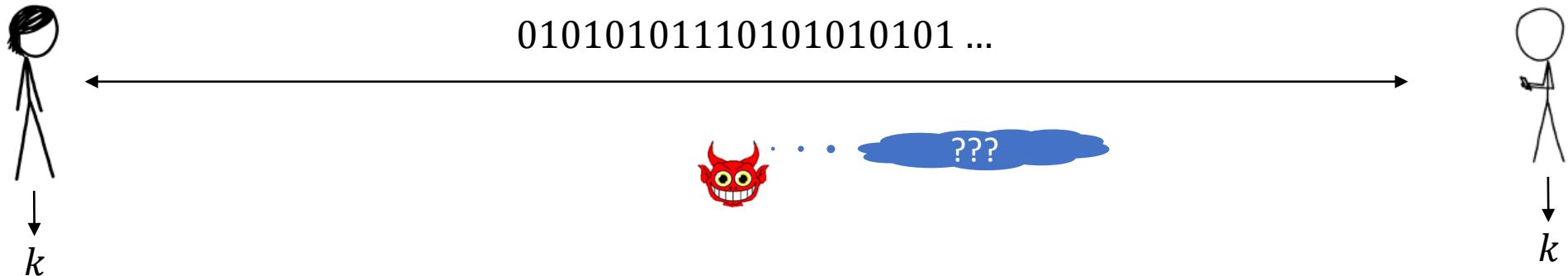
Keylength.com: use  $\log_2 p \approx 15.000$

What does DLOG  
have to do with  
Cryptography?

# So far: Public key encryption



# The key-agreement problem

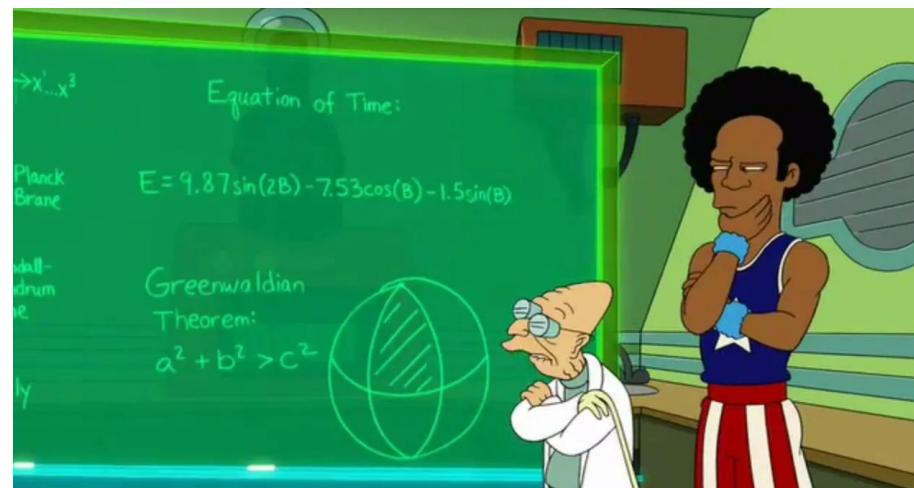


Alice has no special secret information about Bob and vice-versa

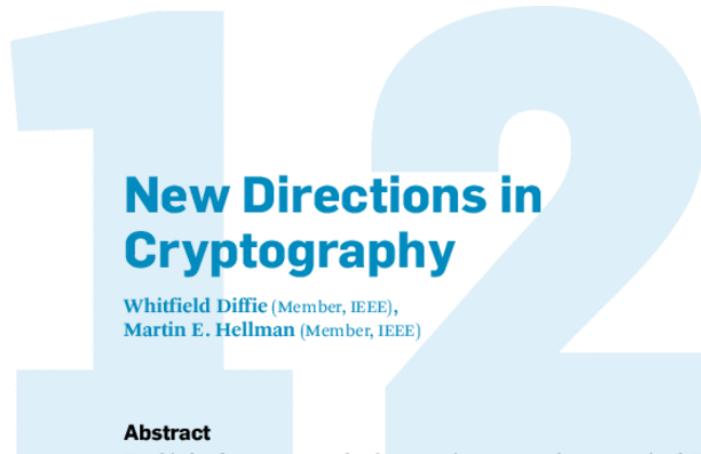
# The key-agreement problem

Seems impossible:  
how to agree on something private over public channel?

Solution:  
Discrete Logarithms!



# 1976: Diffie and Hellman have an idea...



## Abstract

Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

## 12.1

### Introduction

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to

Manuscript received June 3, 1976. This work was partially supported by the National Science Foundation under NSF Grant ENG 10173. Portions of this work were presented at the IEEE Information Theory Workshop, Lenox, MA, June 23-25, 1975 and the IEEE International Symposium on Information Theory in Ronneby, Sweden, June 21-24, 1976.

W. Diffie is with the Department of Electrical Engineering, Stanford University, Stanford, CA, and the Stanford Artificial Intelligence Laboratory, Stanford, CA 94305.

M. E. Hellman is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

Originally published in IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November 1976

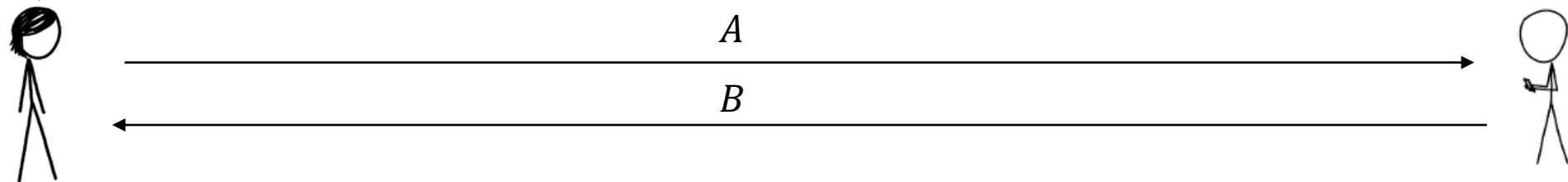
Turing Award in 2015

# Diffie Hellman key agreement

Fix large primes  $p, q$  where  $q|p - 1$

Fix  $g \in Z_p^*$  such that  $g$  has order  $q$

} Public information!



1. Choose random  $a \in Z_q$
2. Compute  $A = g^a \bmod p$
3. Output  $k = B^a \bmod p$

1. Choose random  $b \in Z_q$
2. Compute  $B = g^b \bmod p$
3. Output  $k = A^b \bmod p$

# Diffie Hellman key agreement

## Why it works

$$B^a = (g^b)^a = g^{ab} = (g^a)^b = A^b$$

## Example

$$p = 23, g = 5$$

Alices chooses  $a = 4$ , Bob chooses  $b = 7$

Exchanged messages:  $A = 4, B = 17$

$$17^4 = 4^7 = 8 \bmod 23$$

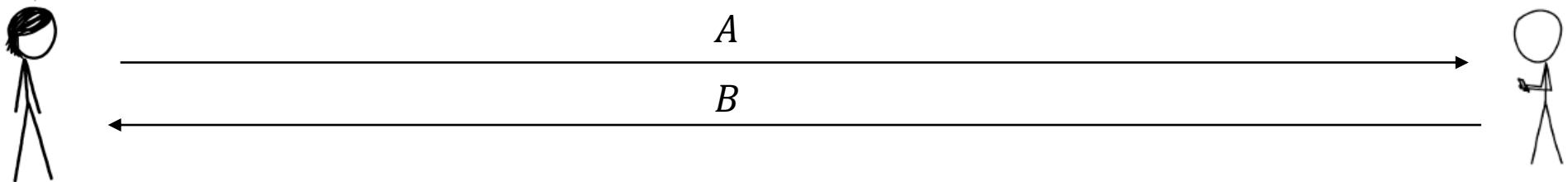
# Security of Diffie Hellman

# Security of Diffie Hellman

Fix large primes  $p, q$  where  $q|p - 1$

Fix  $g \in \mathbb{Z}_p^*$  such that  $g$  has order  $q$

} Public information!



1. Choose
2. Compute
3. Output

Attacker's task:  
Given  $g, A, B$  find  $k$

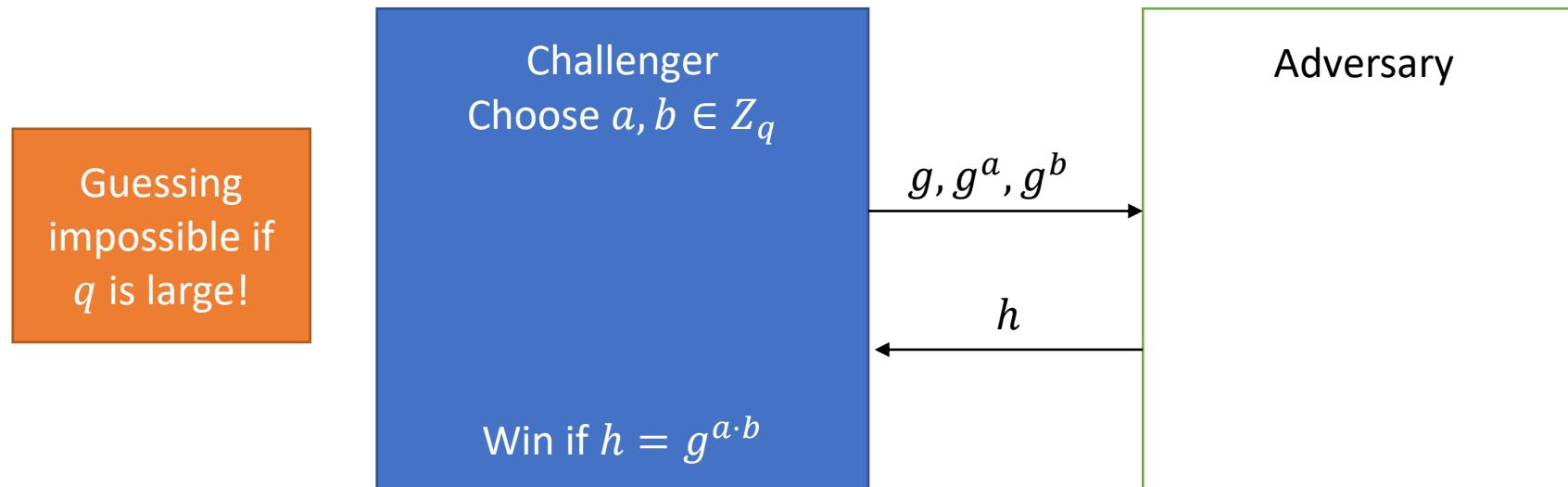
Clearly, DLOG must be difficult!  
But breaking DH is not the same  
as DLOG...

$\in \mathbb{Z}_q$   
 $mod p$   
 $mod p$

# Computational Diffie Hellman Problem (CDH)

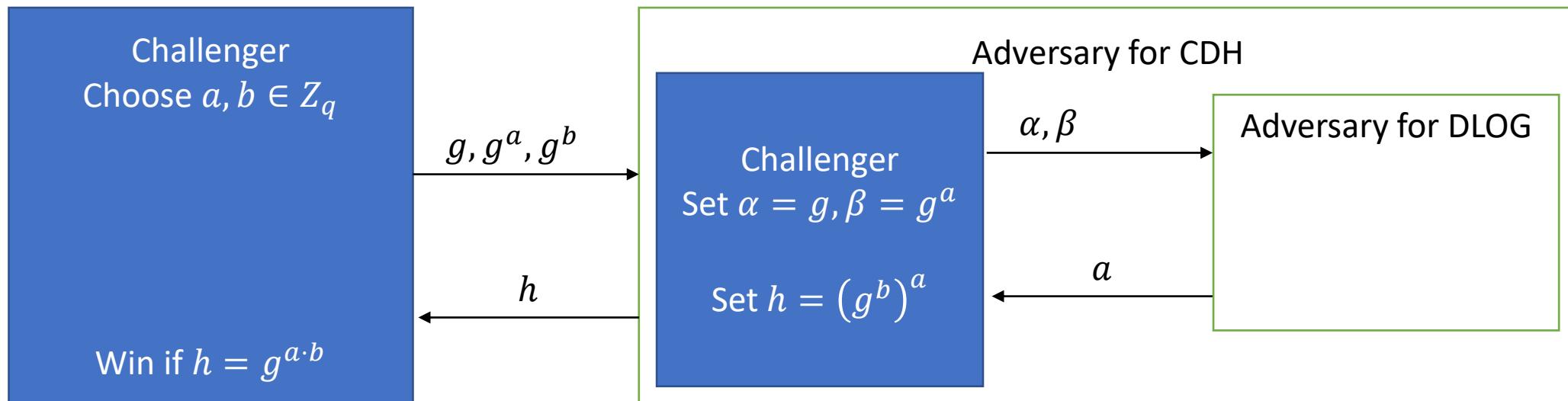
Fix large primes  $p, q$  where  $q \mid p - 1$

Fix  $g \in \mathbb{Z}_p^*$  such that  $g$  has order  $q$



# CDH vs. DLOG

Attack on DLOG  $\Rightarrow$  Attack on CDH



Reverse is not known in general!

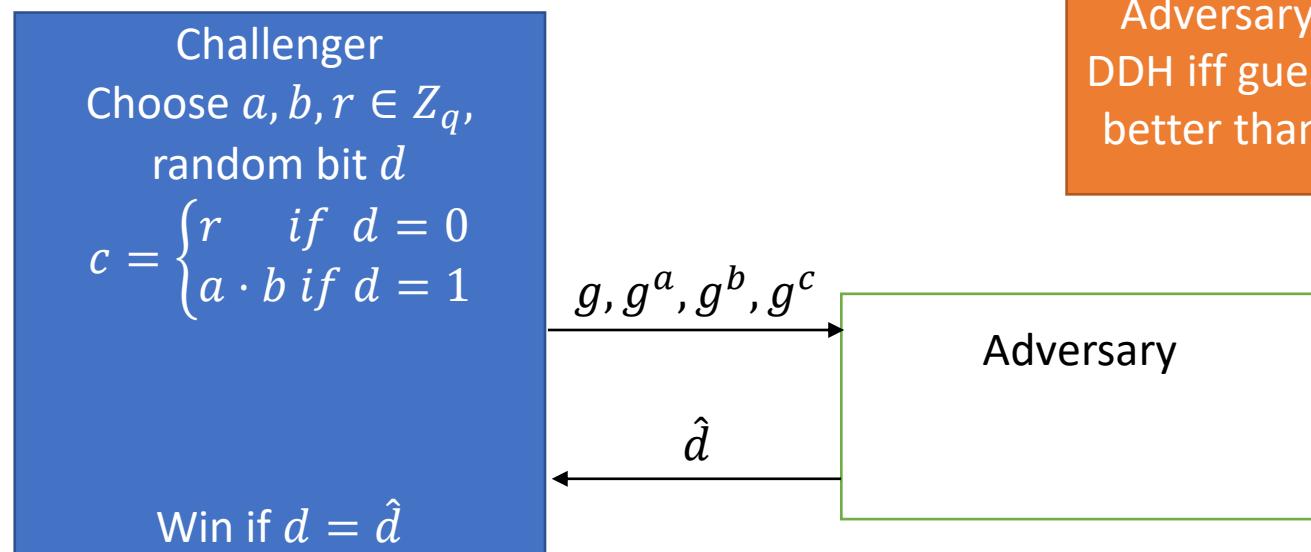
# Decisional Diffie Hellman Problem (DDH)

Fix large primes  $p, q$  where  $q \mid p - 1$

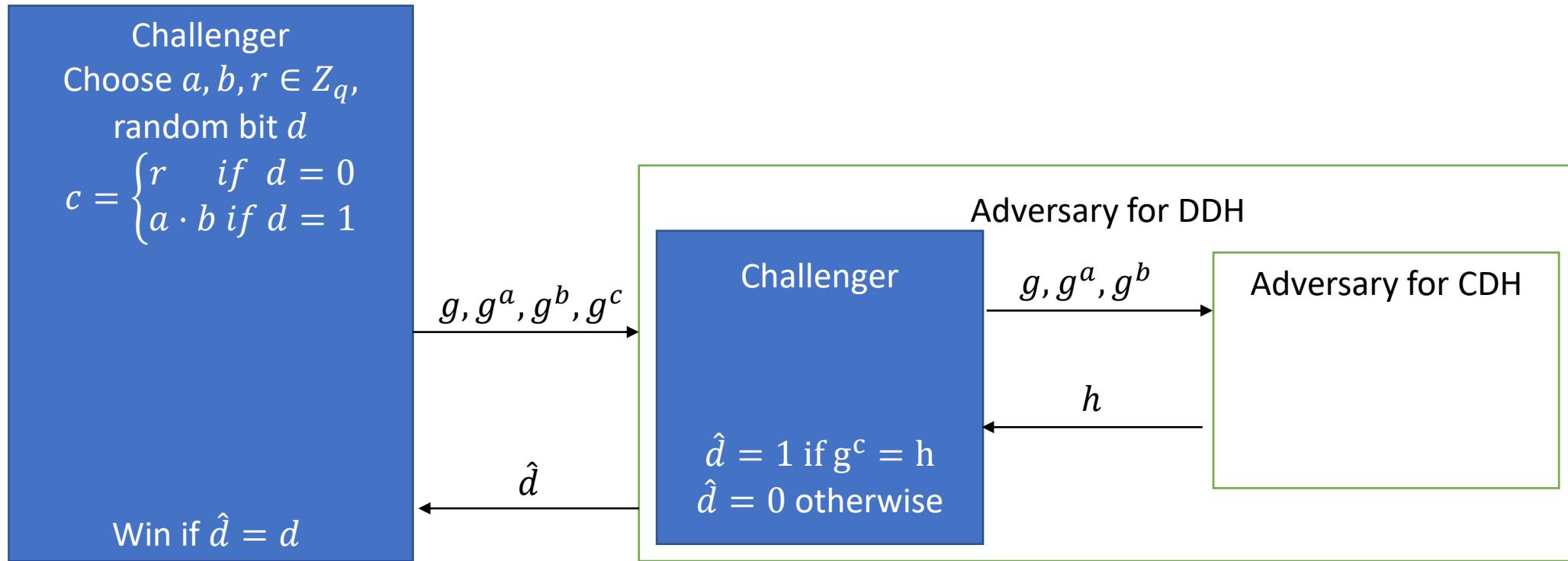
Fix  $g \in \mathbb{Z}_p^*$  such that  $g$  has order  $q$

Guessing is correct with 50% chance!

Adversary wins DDH iff guesses >> better than 50%!



# DDH vs. CDH

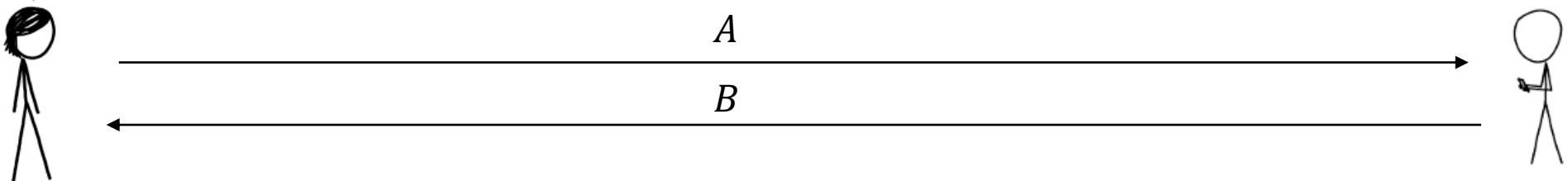


Reverse is not known in general.

# Security of Diffie Hellman

Fix large primes  $p, q$  where  $q|p - 1$

Fix  $g \in Z_p^*$  such that  $g$  has order  $q$



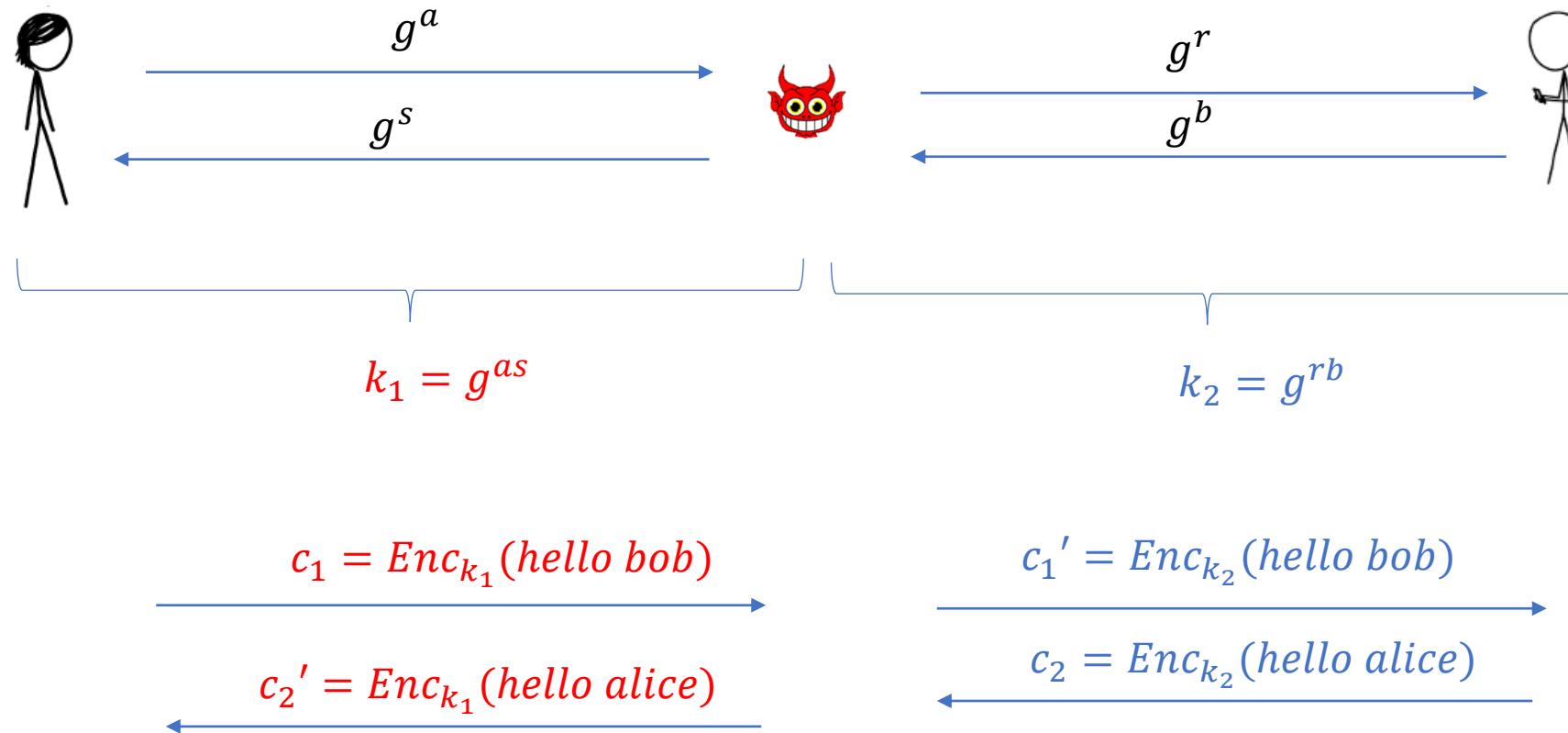
1. Choose random  $a \in Z_q$
2. Compute  $A = g^a \bmod p$
3. Output  $k = B^a \bmod p$

Assuming CDH, no attacker can efficiently compute  $k$

1. Choose random  $b \in Z_q$
2. Compute  $B = g^b \bmod p$
3. Output  $k = A^b \bmod p$

Assuming DDH, no attacker can efficiently distinguish  $k$  from a random group element

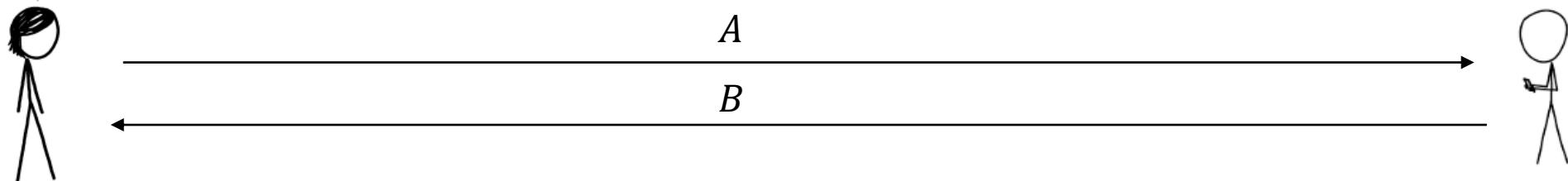
# Diffie Hellman and active attacks



# Diffie Hellman key agreement - generalized

Fix a large group  $G$  of prime order, generator  $g \in G$

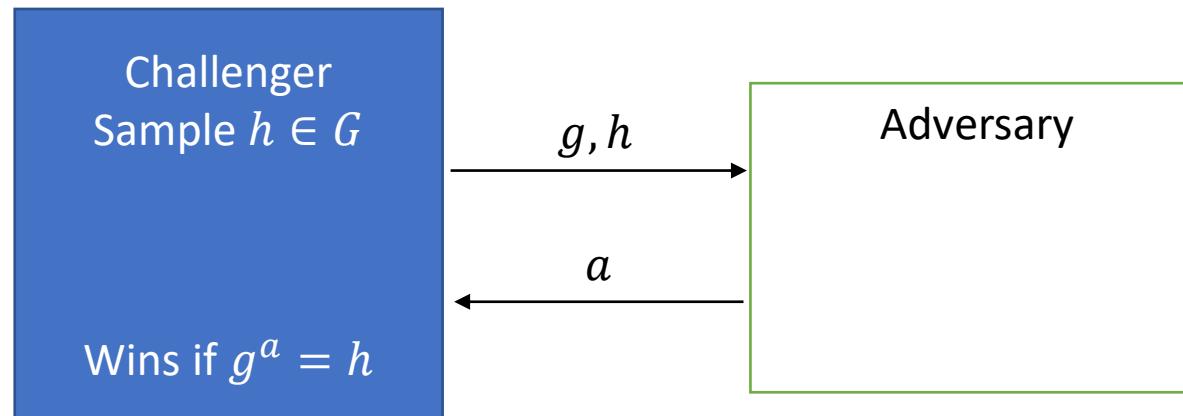
It must be hard to compute  $a$  given  $G, g, g^a$



- |                                  |                                  |
|----------------------------------|----------------------------------|
| 1. Choose random $a \in Z_{ G }$ | 1. Choose random $b \in Z_{ G }$ |
| 2. Compute $A = g^a$             | 2. Compute $B = g^b$             |
| 3. Output $k = B^a$              | 3. Output $k = A^b$              |

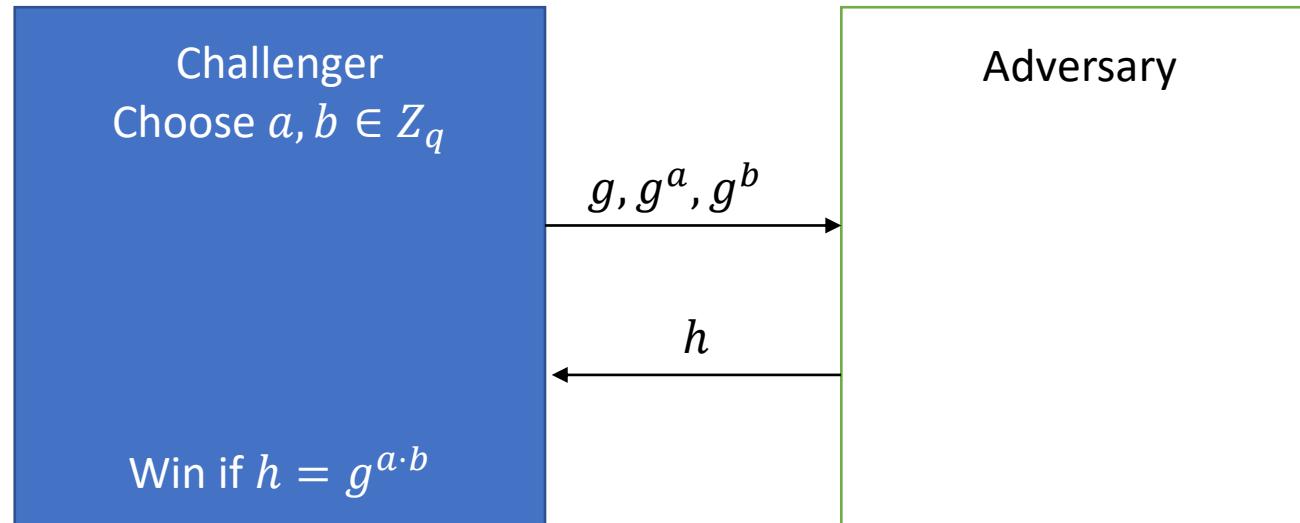
# Generalized DLOG

Fix a large group  $G$  of prime order  $q$ , generator  $g \in G$



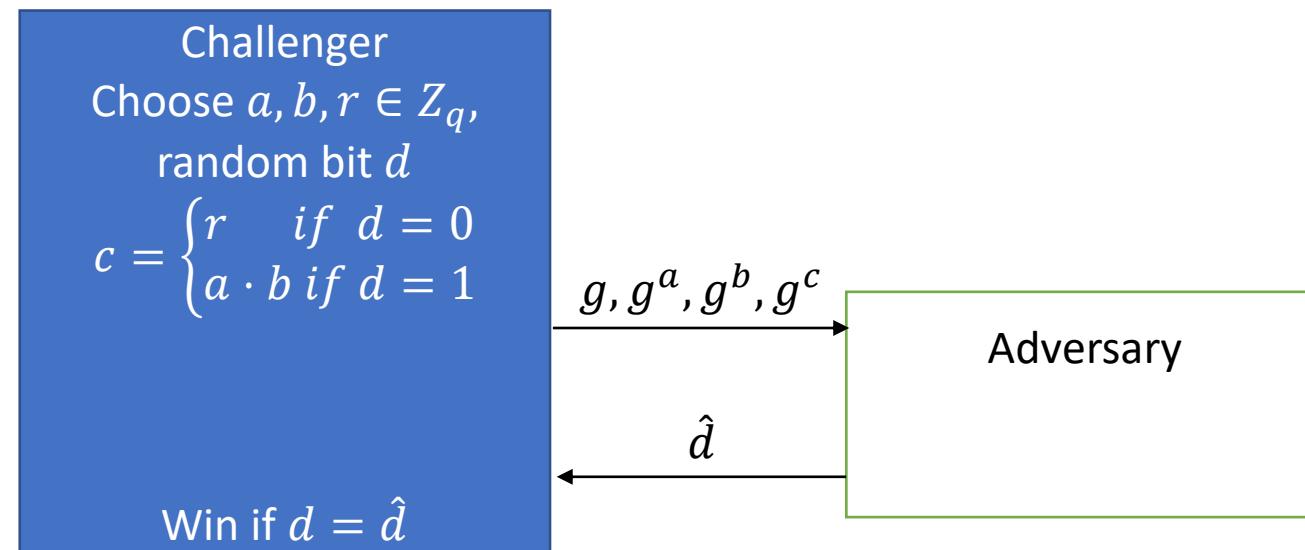
# Generalized CDH

Fix a large group  $G$  of prime order  $q$ , generator  $g \in G$



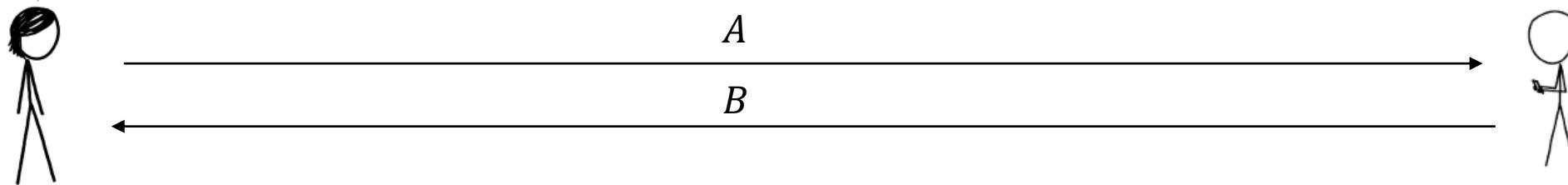
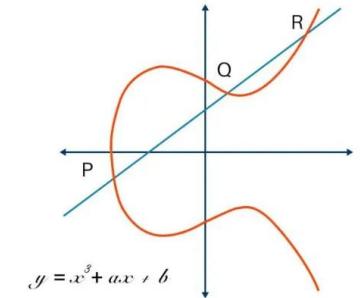
# Generalized DDH

Fix a large group  $G$  of prime order  $q$ , generator  $g \in G$



# Diffie Hellman in practice

**We saw:**  $\log_2(p)$  in 1000s of bits



Used in practice:  
so-called Elliptic-curve groups (NIST curves, EC25519)

They deliver 128 bit security, but  $A, B$  only  $\approx 260$  bits long

# How does ECC work?

Let  $K$  be any field not of characteristic 2,3 and  $a, b \in K$ .

Then the solutions  $(X, Y)$  of  $E: Y^2 = X^3 + aX + b$  form a group.

In particular, if  $K = F_q$  with  $q = p^n, p > 3$  where  $4a^3 \neq 27b^2$  then  $|E(K)|$  has order  $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$

# The group law if $p > 3$

Let  $E: Y^2 = X^3 + aX + b$  with  $a, b \in F_q$ . Assume the existence of a point  $O$ .

Define  $P_1 + O = O + P_1 = O$  and  $P_1 - P_1 = O$

If  $P_1 = (x_1, y_1)$  and  $P_2 = (x_2, y_2)$  then  $-P_1 = (x_1, -y_1)$

If  $x_1 \neq x_2$  then  $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$

If  $x_1 = x_2$  and  $y_1 \neq 0$  then  $\lambda = \frac{3x_1^2 + a}{2y_1}$

Then  $P_3 = P_1 + P_2 \neq O$  is given by

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= (x_1 - x_3)\lambda - y_1 \end{aligned}$$

# EC group has additive group law

Let  $P \in E(F_q)$  then define

$$[m]P = \underbrace{P + \cdots + P}_{m \text{ times}}$$

ECDLP:

Given  $E(F_q), P, [m]P$  it is hard to recover  $m$

# Example

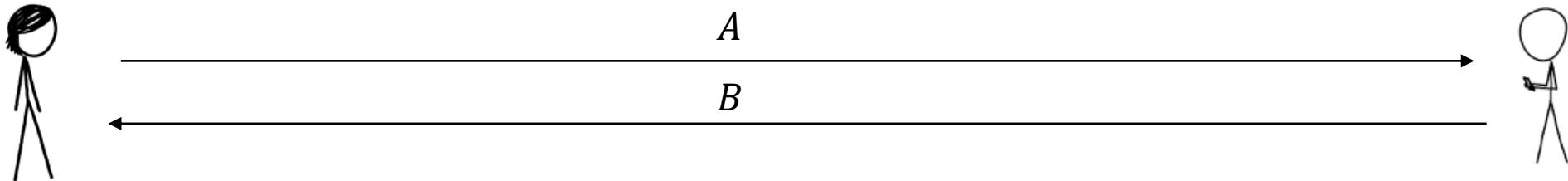
Let  $q = 7, a = 1, b = 3$   
 $E: Y^2 = X^3 + X + 3$

$+$	$O$	$(4, 1)$	$(6, 6)$	$(5, 0)$	$(6, 1)$	$(4, 6)$
$O$	$O$	$(4, 1)$	$(6, 6)$	$(5, 0)$	$(6, 1)$	$(4, 6)$
$(4, 1)$	$(4, 1)$	$(6, 6)$	$(5, 0)$	$(6, 1)$	$(4, 6)$	$O$
$(6, 6)$	$(6, 6)$	$(5, 0)$	$(6, 1)$	$(4, 6)$	$O$	$(4, 1)$
$(5, 0)$	$(5, 0)$	$(6, 1)$	$(4, 6)$	$O$	$(4, 1)$	$(6, 6)$
$(6, 1)$	$(6, 1)$	$(4, 6)$	$O$	$(4, 1)$	$(6, 6)$	$(5, 0)$
$(4, 6)$	$(4, 6)$	$O$	$(4, 1)$	$(6, 6)$	$(5, 0)$	$(6, 1)$

If  $P = (4,1)$  then

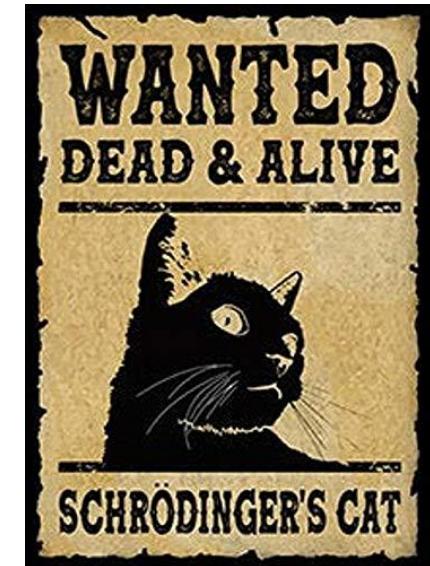
$$\begin{aligned}[2]P &= (6,6) \\ [3]P &= (5,0) \\ [4]P &= (6,1) \\ [5]P &= (4,6) \\ [6]P &= O\end{aligned}$$

# Diffie Hellman



## Caveat:

Any Diffie-Hellman ( $\text{mod } p$ , Elliptic curve)  
not secure against quantum computers



# Summary

1. The Discrete Logarithm Problem
2. Diffie Hellman Key Exchange
3. The CDH and DDH problems

# The quantum computing threat to cryptography

Course 01410, Crypto I

Christian Majenz

Assistant Professor, Cybersecurity Engineering Section, DTU Compute

# After today, you will...

- ▶ ...be able to put quantum computing as a disruptive technology into perspective,
- ▶ ...have identified cryptographic infrastructure that is susceptible to quantum computing attacks,
- ▶ ...have a basic understanding of how quantum computers can be used to attack cryptographic schemes, and
- ▶ ...have a rough idea when cryptanalytically relevant quantum computers can be expected to become reality.

# Intro

# Quantum computers

# Quantum computers

- ▶ Accelerating effort to build a quantum computer

# Quantum computers

- ▶ Accelerating effort to build a quantum computer
- ▶ Major investments:



Microsoft



# Quantum computers

- ▶ Accelerating effort to build a quantum computer
- ▶ Major investments:



Microsoft



**We need to prepare cryptography for the arrival  
of quantum computers!**

# Quantum computers

- ▶ Accelerating effort to build a quantum computer
- ▶ Major investments:



Microsoft



**We need to prepare cryptography for the arrival of quantum computers!**

- ▶ Security against quantum attackers

# Quantum computers

- ▶ Accelerating effort to build a quantum computer
- ▶ Major investments:



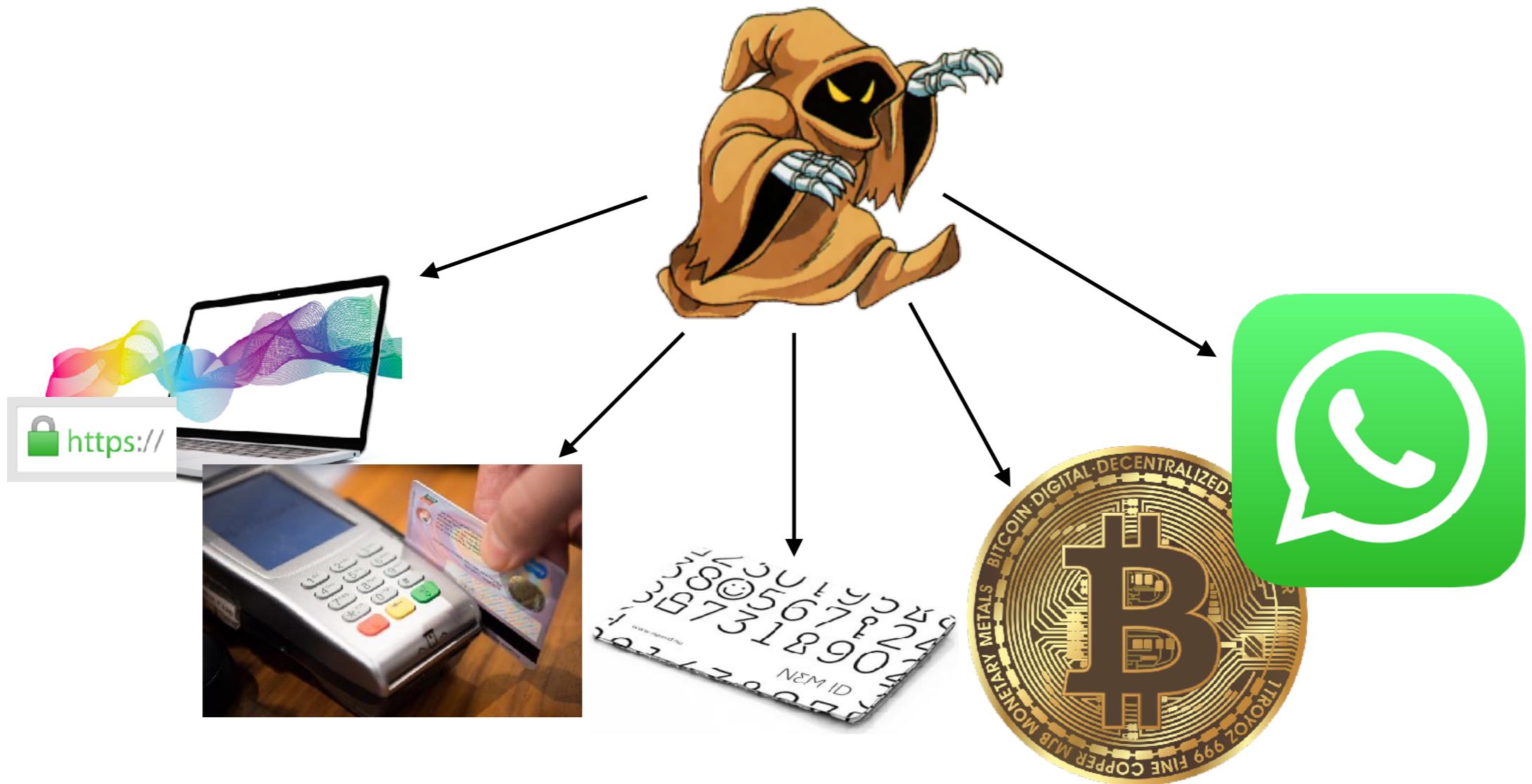
Microsoft



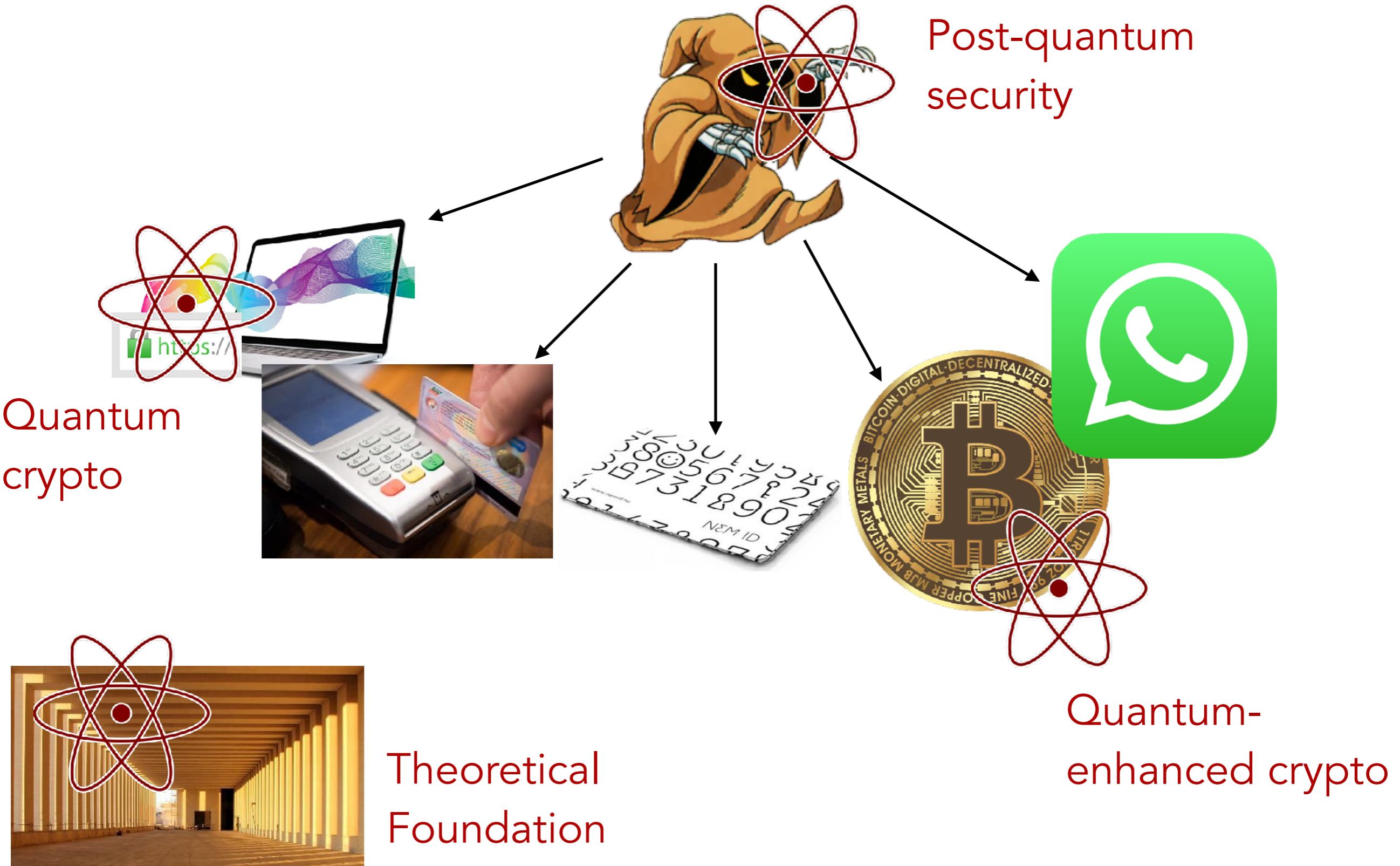
**We need to prepare cryptography for the arrival of quantum computers!**

- ▶ Security against quantum attackers
- ▶ Quantum cryptography

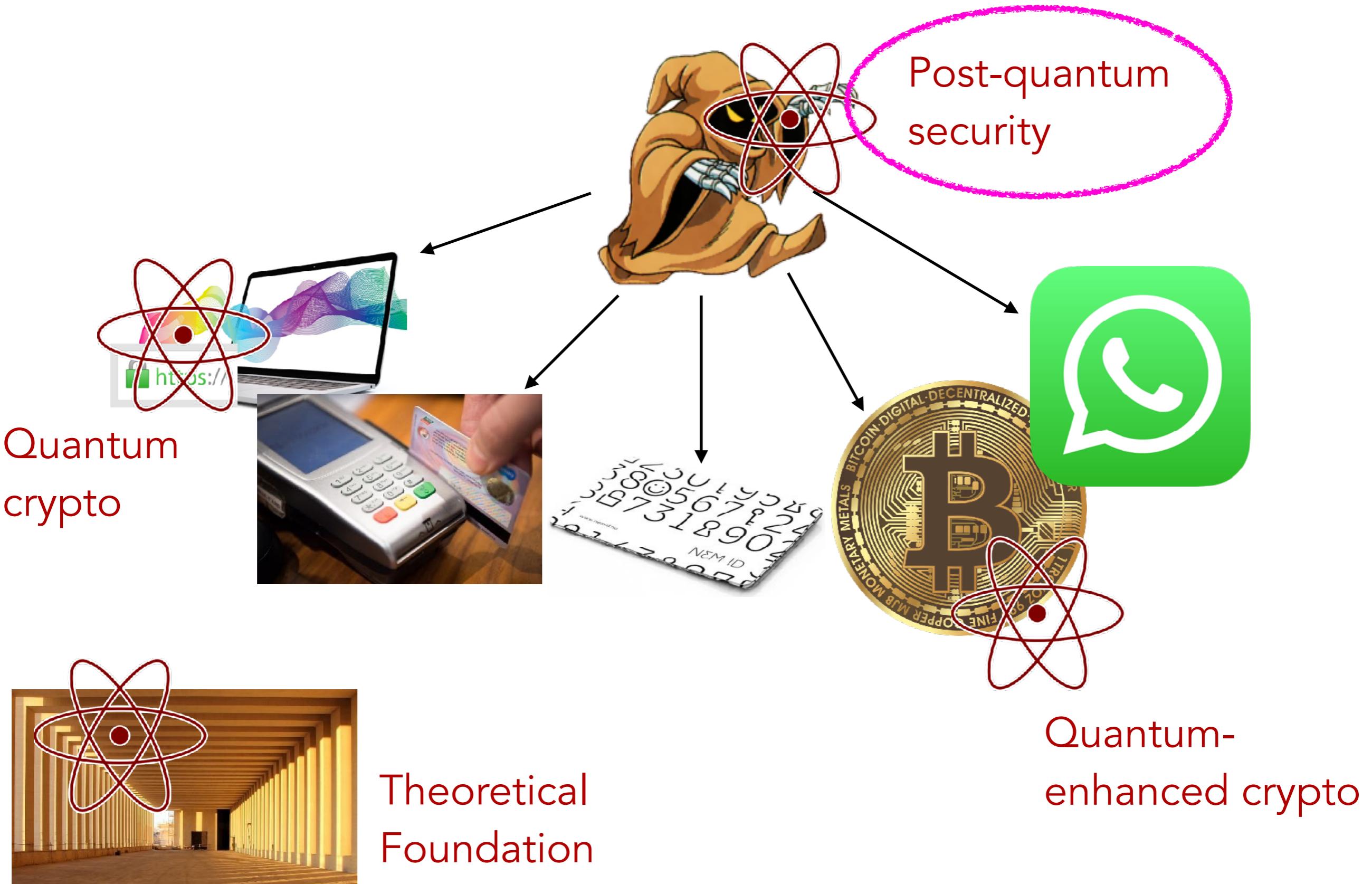
# Cryptography is everywhere



# Quantum computing is changing cryptography



# Quantum computing is changing cryptography



What is a quantum computer?

# Quantum physics is everywhere

# Quantum physics is everywhere

In Information technology:

- ▶ Semiconductors

# Quantum physics is everywhere

In Information technology:

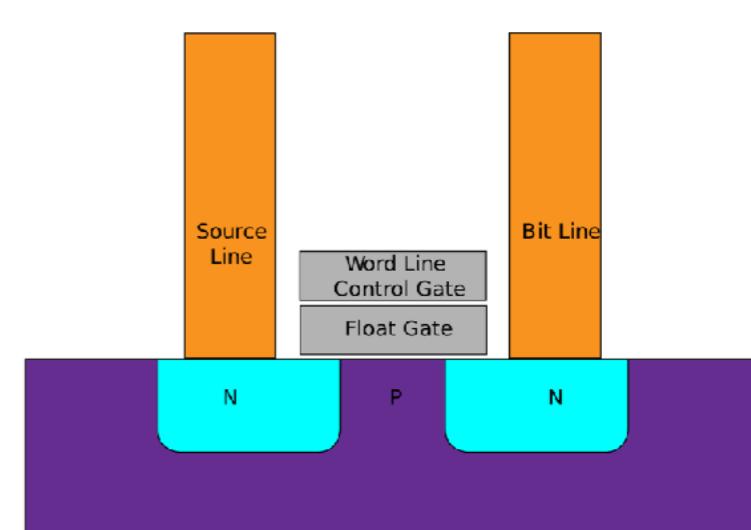
- ▶ Semiconductors
  - Transistors



# Quantum physics is everywhere

In Information technology:

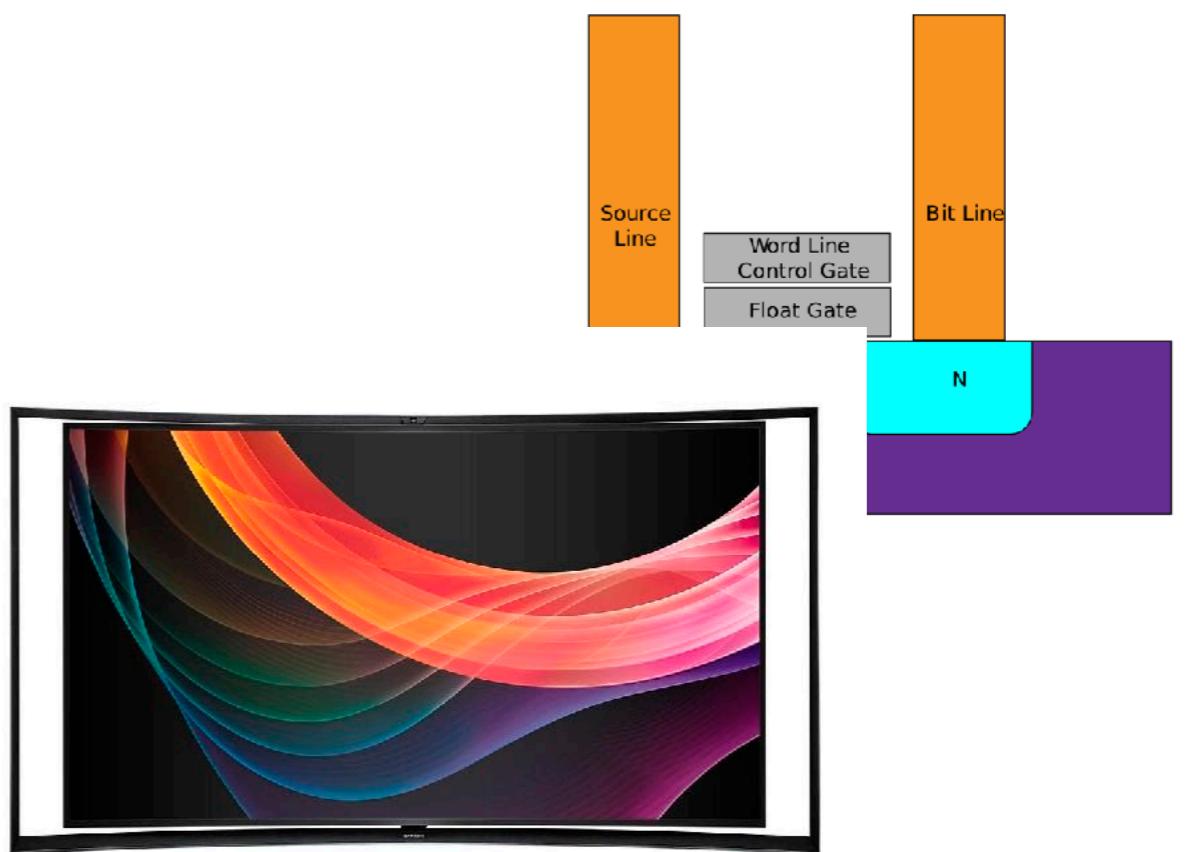
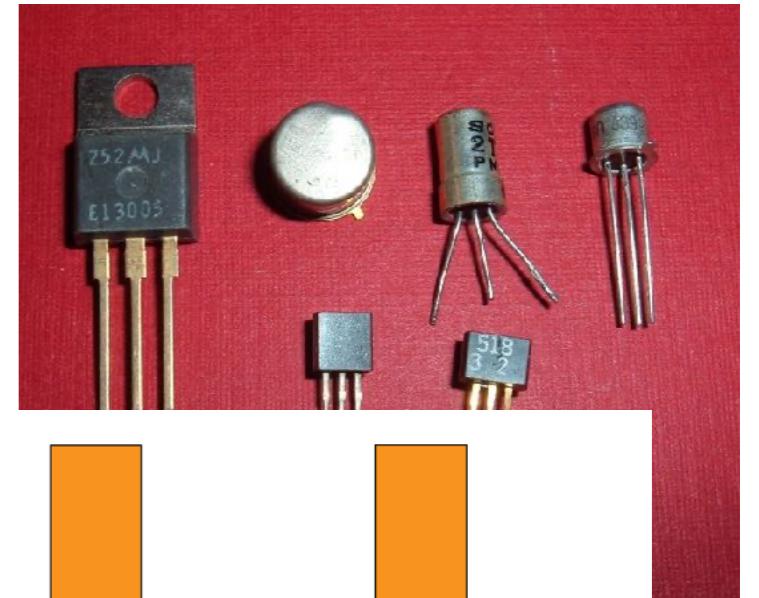
- ▶ Semiconductors
  - Transistors
  - Flash memory



# Quantum physics is everywhere

In Information technology:

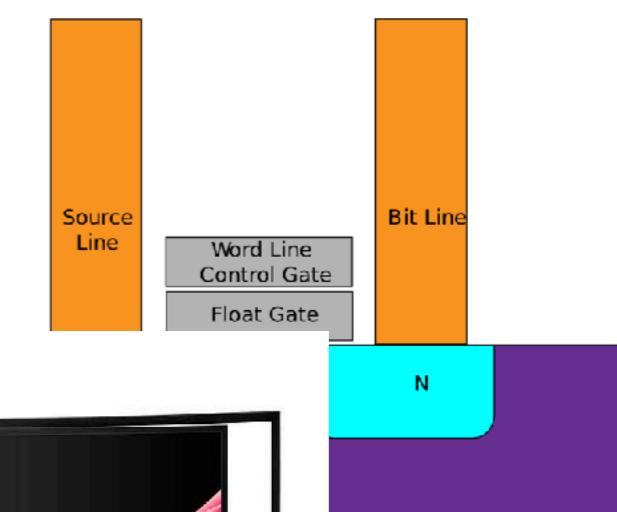
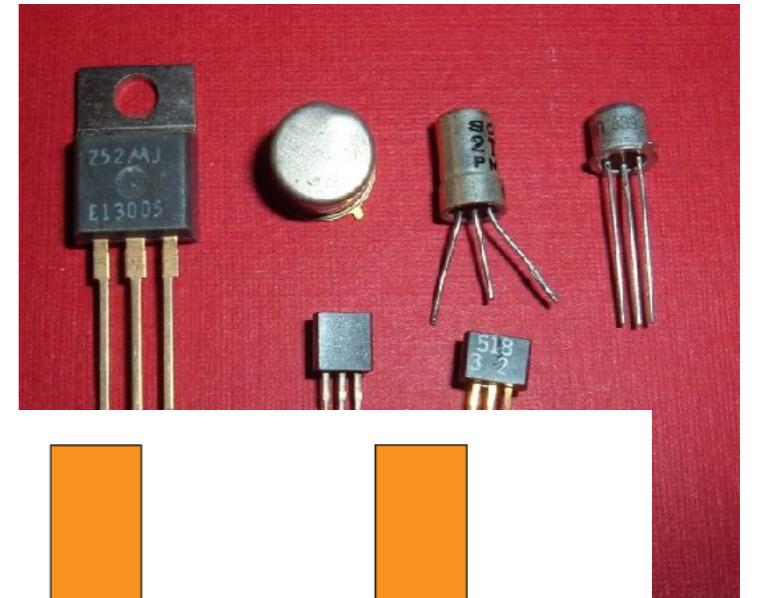
- ▶ Semiconductors
  - Transistors
  - Flash memory
  - OLED
  - ...



# Quantum physics is everywhere

In Information technology:

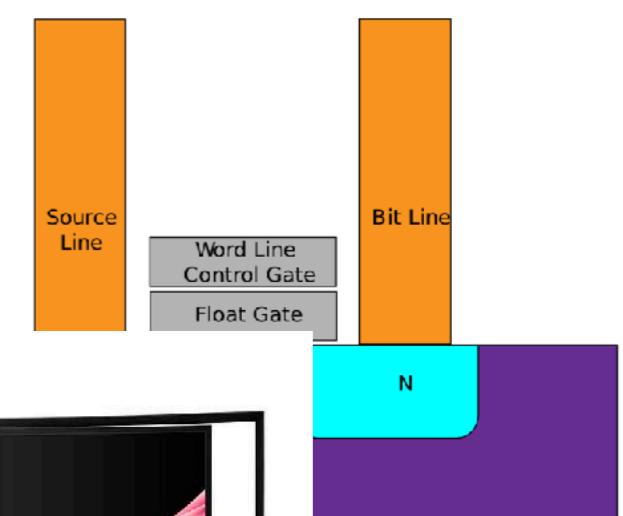
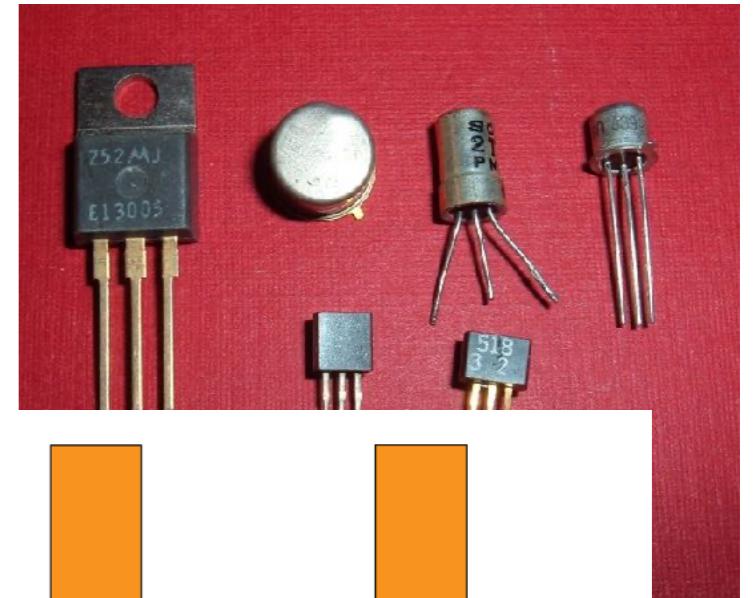
- ▶ Semiconductors
  - Transistors
  - Flash memory
  - OLED
  - ...
- ▶ Optical communication



# Quantum physics is everywhere

In Information technology:

- ▶ Semiconductors
  - Transistors
  - Flash memory
  - OLED
  - ...
- ▶ Optical communication
- ▶ 3D movie theatre
- ▶ ...

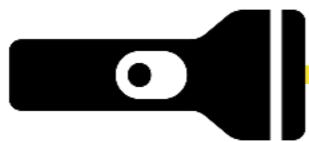


# Example: polarization of light



# Example: polarization of light

Classical\* property of Light: Color



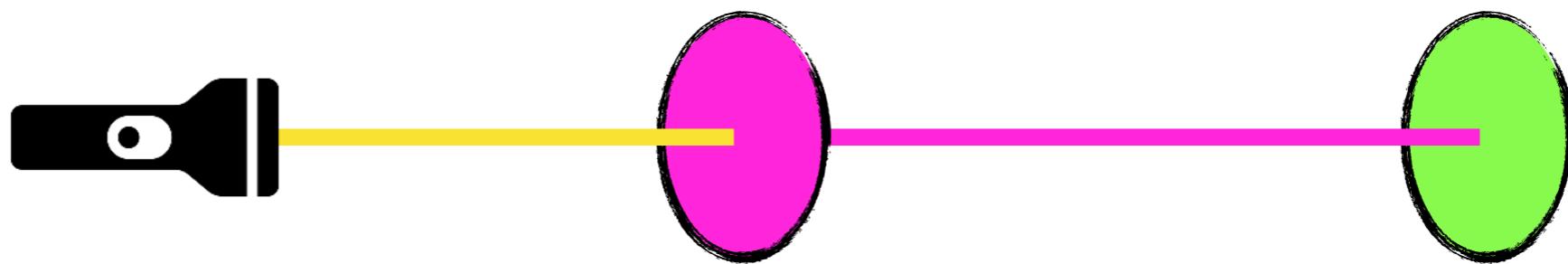
# Example: polarization of light

Classical\* property of Light: Color



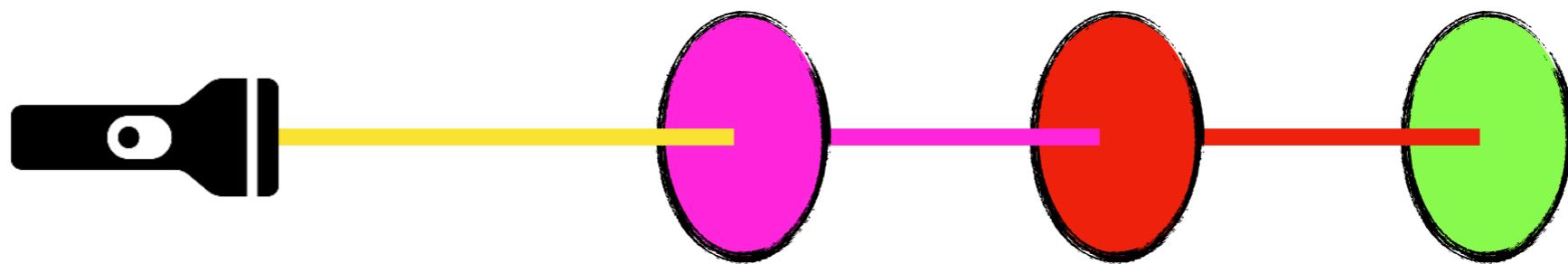
# Example: polarization of light

Classical\* property of Light: Color



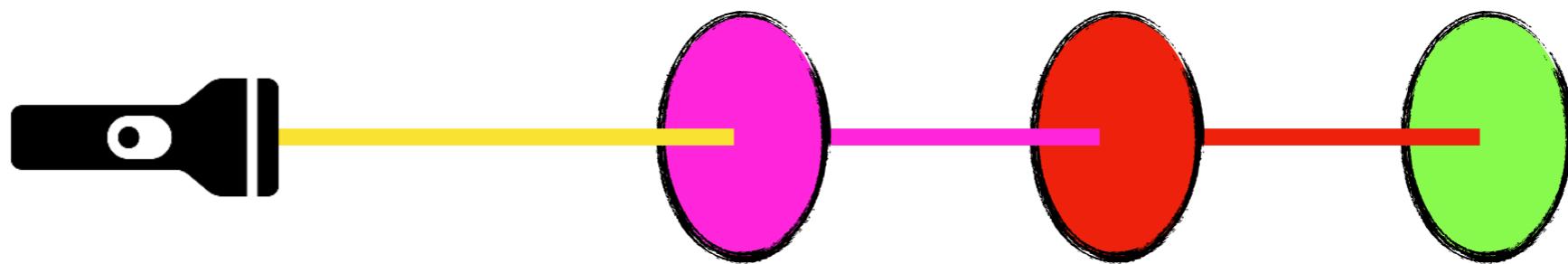
# Example: polarization of light

Classical\* property of Light: Color



# Example: polarization of light

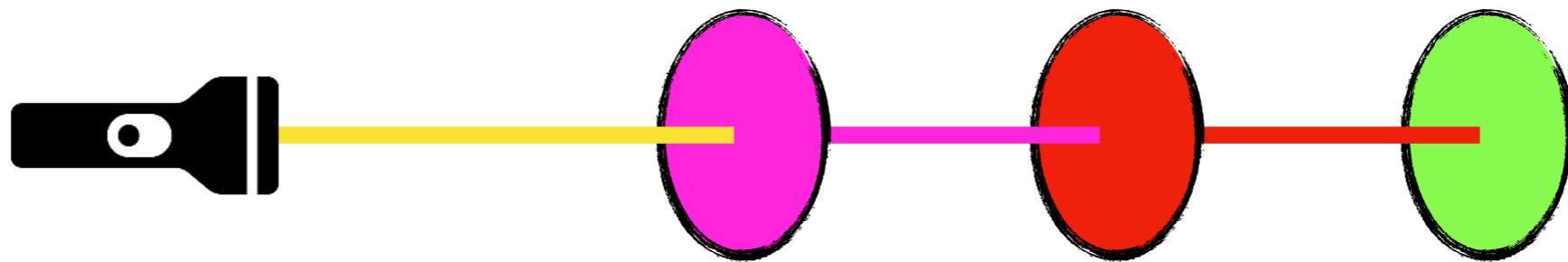
Classical\* property of Light: Color



Quantum property of light: Polarization

# Example: polarization of light

Classical\* property of Light: Color

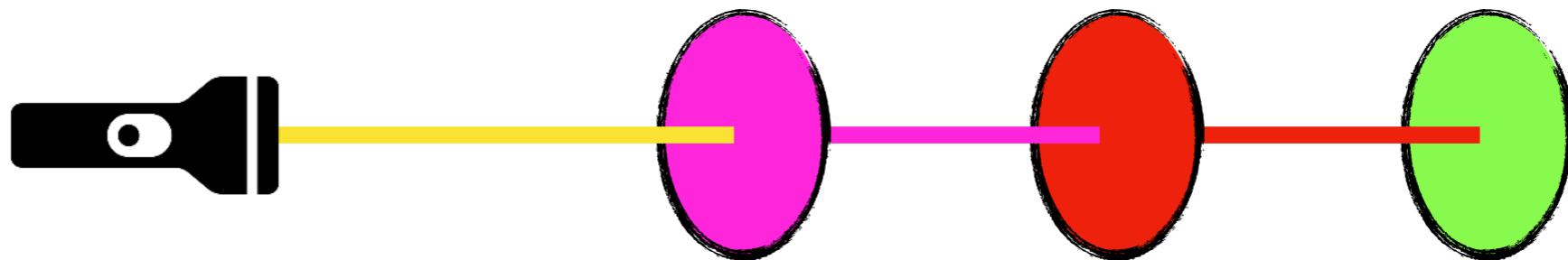


Quantum property of light: Polarization



# Example: polarization of light

Classical\* property of Light: Color

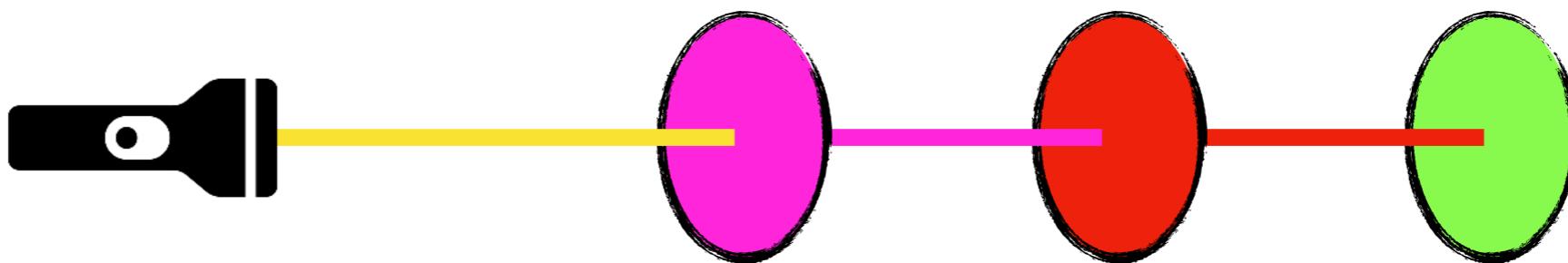


Quantum property of light: Polarization

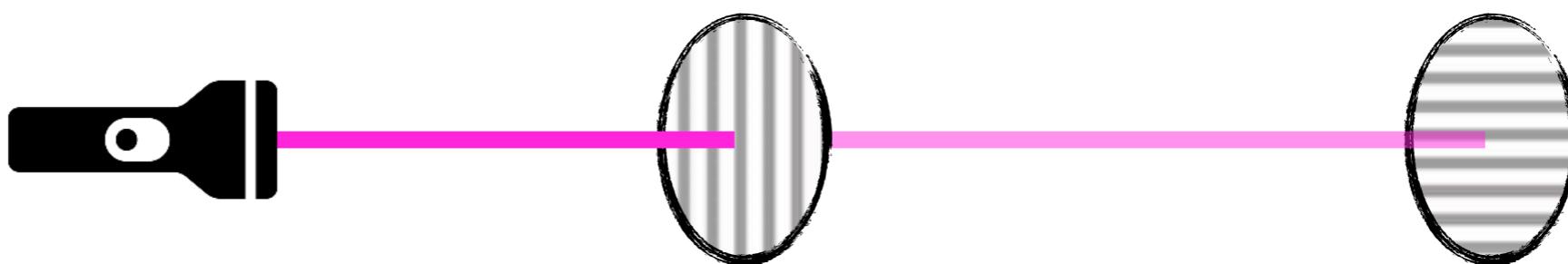


# Example: polarization of light

Classical\* property of Light: Color

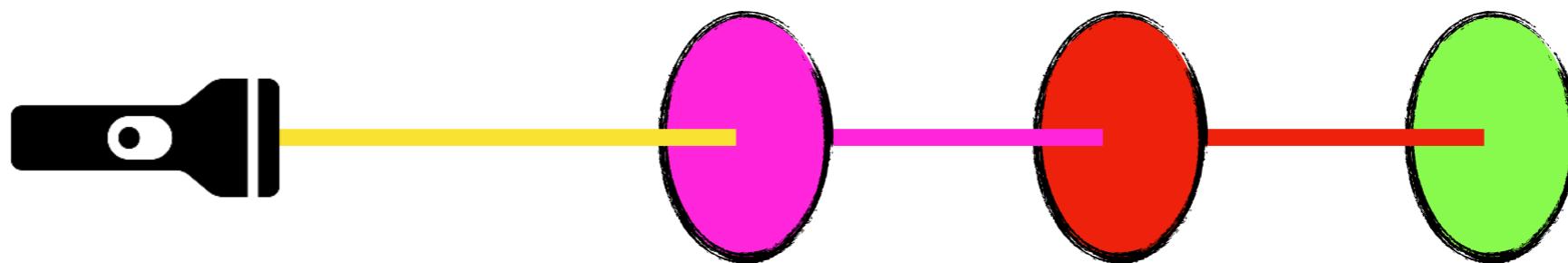


Quantum property of light: Polarization

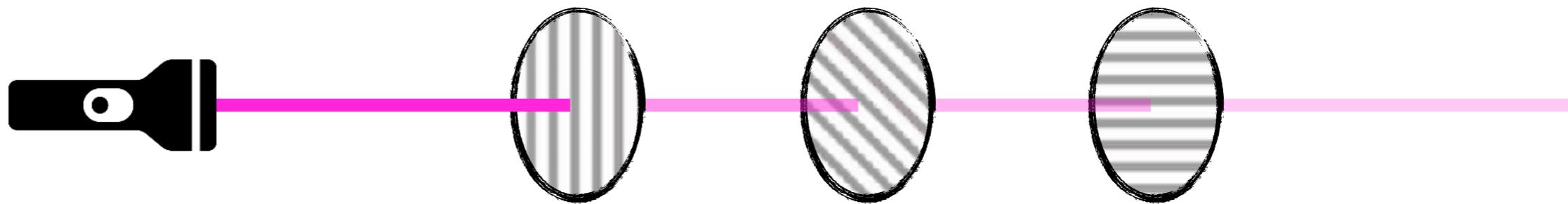


# Example: polarization of light

Classical\* property of Light: Color



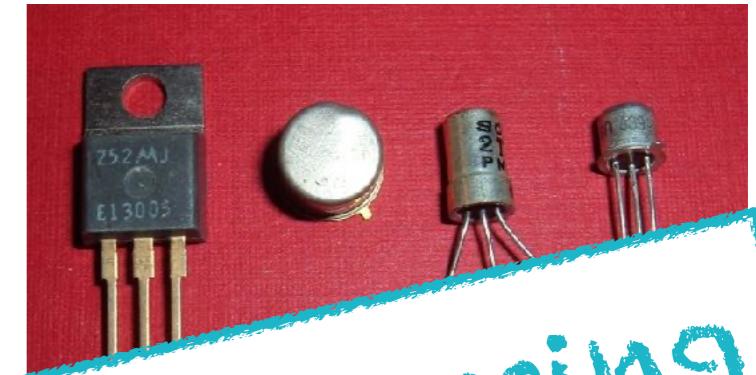
Quantum property of light: Polarization



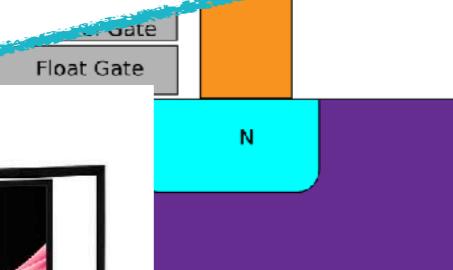
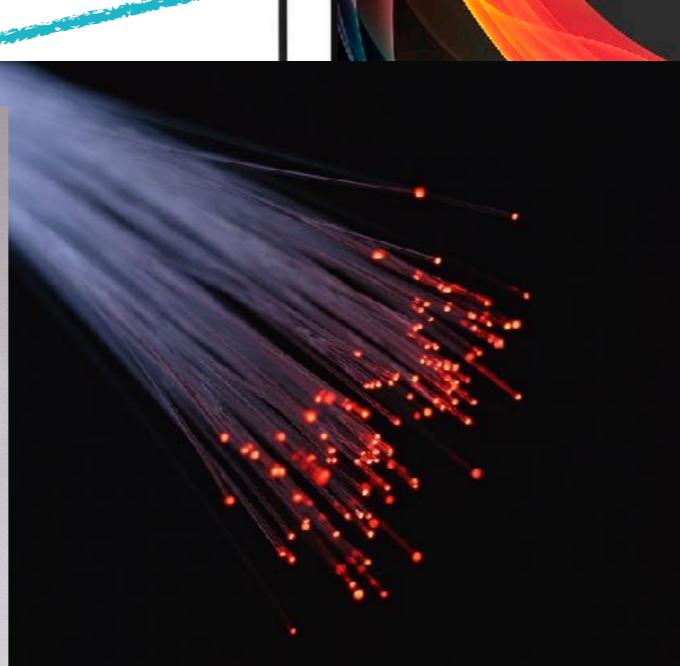
# Quantum physics is everywhere

In Information technology:

- ▶ Semiconductors
  - Transistors
  - Flash memory
  - OLED
  - ...
- ▶ Optical co...



quantum physics is used for processing  
"non-quantum" information.



# Quantum computing

“Information is Physical.”

Rolf Landauer

# Quantum computing

“Information is Physical.”

Rolf Landauer

⇒ If the physics is quantum, information is as well.

# What is a quantum computer?

Classical information: "Does the flashlight emit green light?"

Specified by a bit  $b \in \{0,1\}$

More generally: bit strings  $x \in \{0,1\}^n$

Basic operations: Logical gates

# What is a quantum computer?

Classical information: "Does the flashlight emit green light?"

Specified by a bit  $b \in \{0,1\}$

More generally: bit strings  $x \in \{0,1\}^n$

Basic operations: Logical gates

Quantum information: "What is the polarization of the light emitted from the flashlight?"

Specified by a unit vector  $|\psi\rangle \in \mathbb{R}^2$

# What is a quantum computer?

Classical information: "Does the flashlight emit green light?"

Specified by a bit  $b \in \{0,1\}$

More generally: bit strings  $x \in \{0,1\}^n$

Basic operations: Logical gates

Quantum information: "What is the polarization of the light emitted from the flashlight?"

Specified by a unit vector  $|\psi\rangle \in \mathbb{R}^2$

More generally: unit vectors  $|\psi\rangle \in \mathbb{C}^{2^n}$  (bit strings: basis)

# What is a quantum computer?

Classical information: “Does the flashlight emit green light?”

Specified by a bit  $b \in \{0,1\}$

More generally: bit strings  $x \in \{0,1\}^n$

Basic operations: Logical gates

Quantum information: “What is the polarization of the light emitted from the flashlight?”

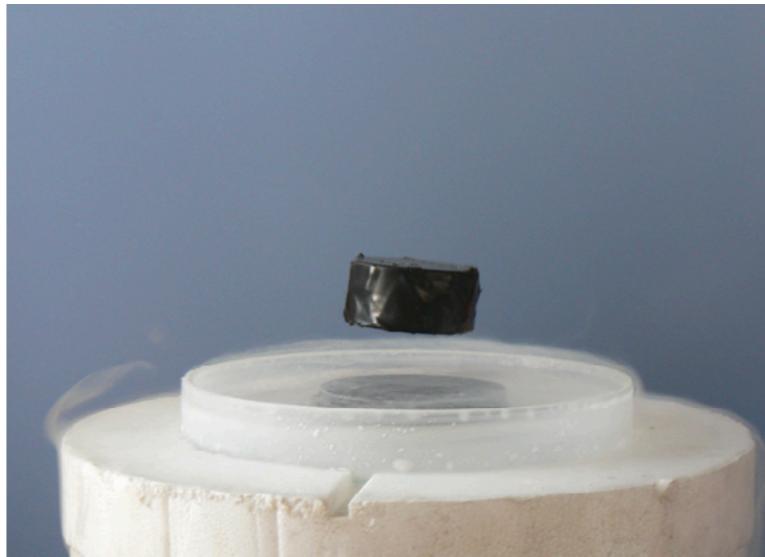
Specified by a unit vector  $|\psi\rangle \in \mathbb{R}^2$

More generally: unit vectors  $|\psi\rangle \in \mathbb{C}^{2^n}$  (bit strings: basis)

Basic operations: “Quantum gates”  $\Leftrightarrow$  Unitary matrices

# Physical realization

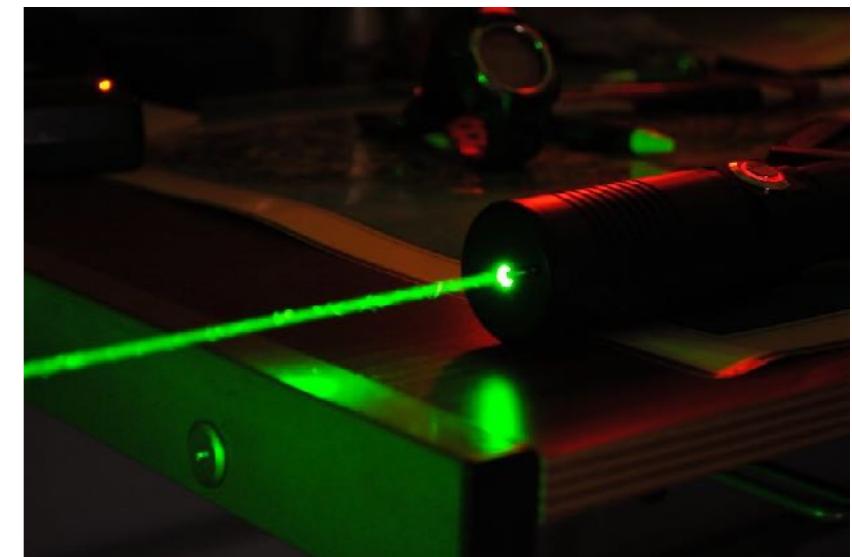
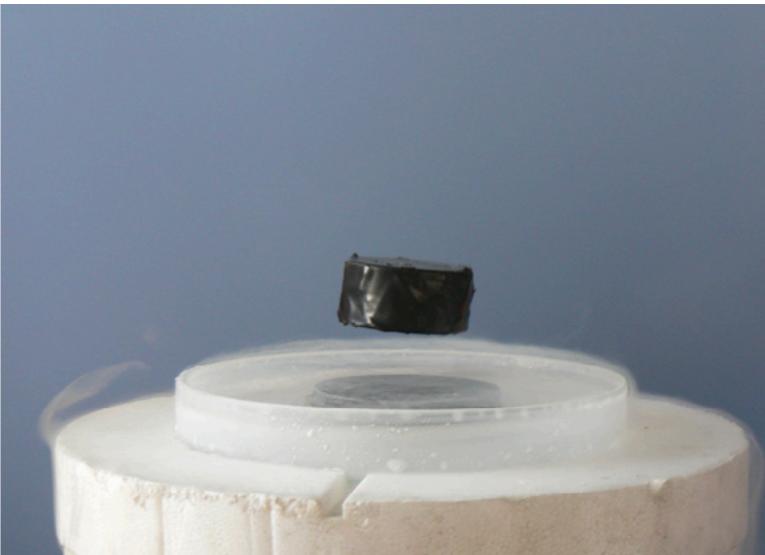
# Physical realization



- ▶ Superconducting

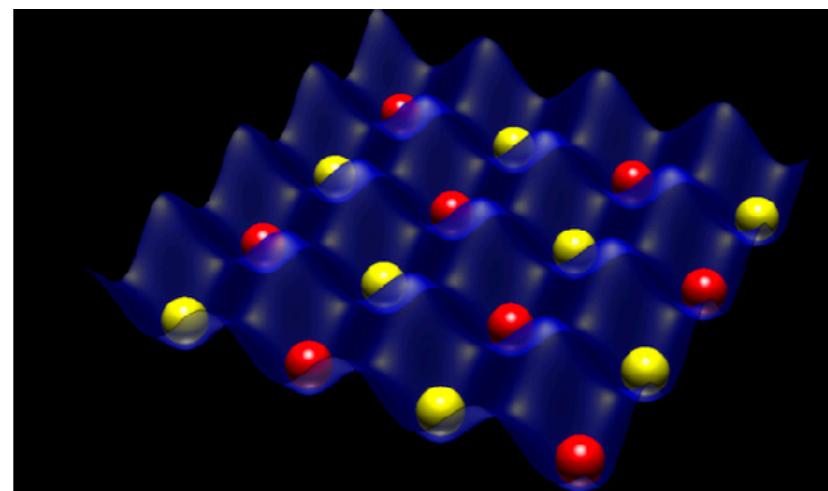
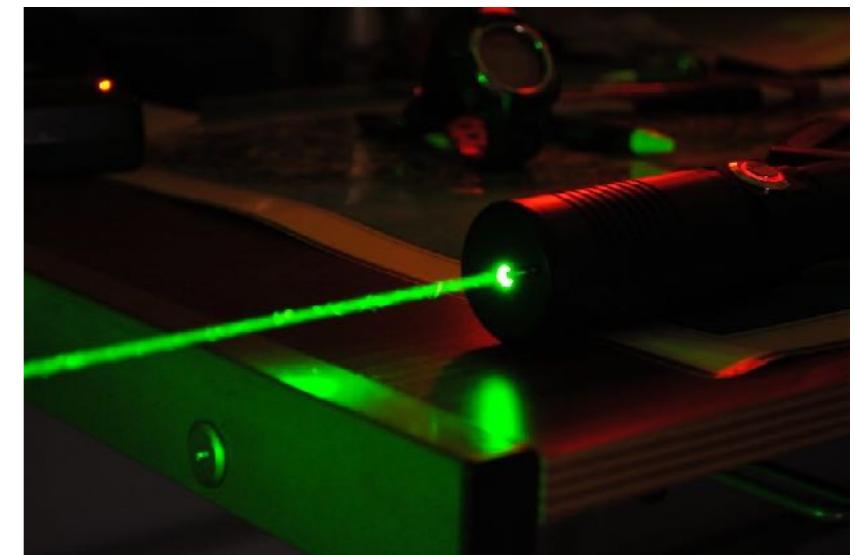
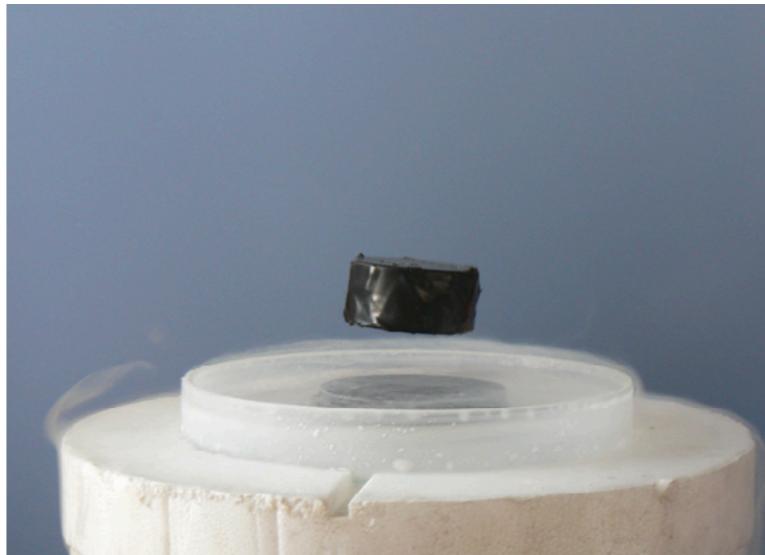
# Physical realization

- ▶ Superconducting
- ▶ Photonic



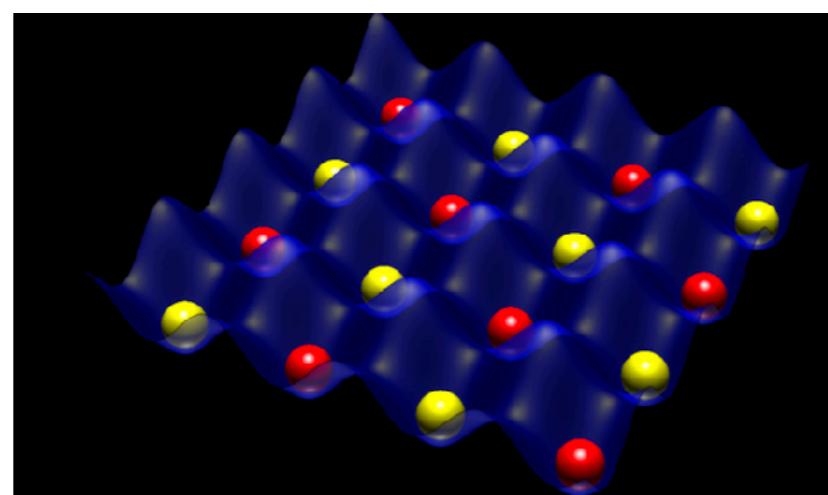
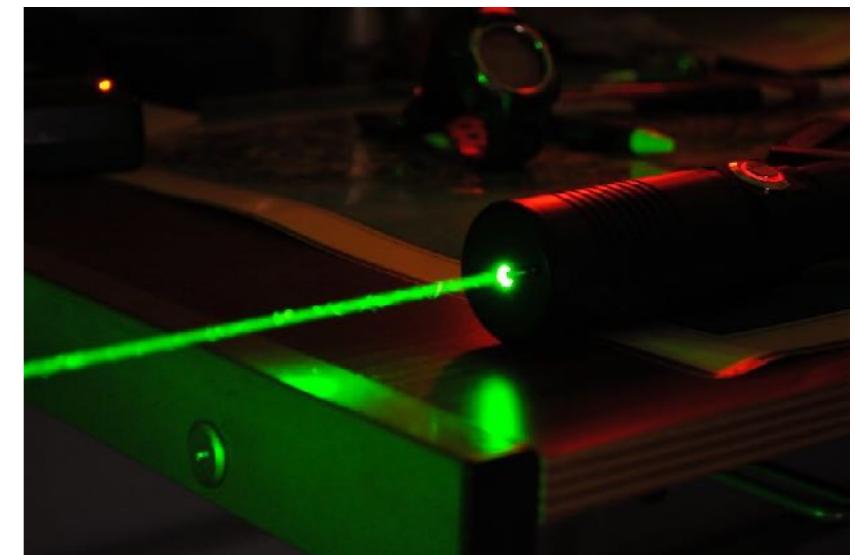
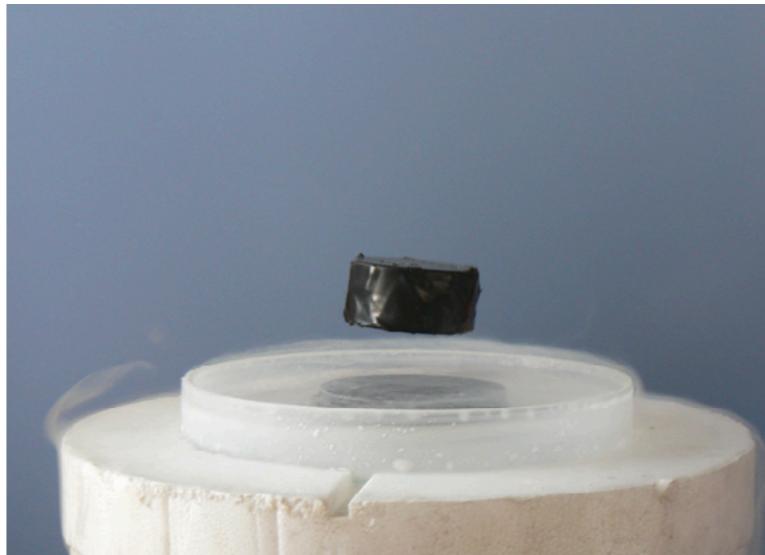
# Physical realization

- ▶ Superconducting
- ▶ Photonic
- ▶ Cold atoms



# Physical realization

- ▶ Superconducting
- ▶ Photonic
- ▶ Cold atoms
- ▶ Trapped ions



But what is it good for...

# Potential applications of quantum computing

# Potential applications of quantum computing

- ▶ Quantum chemistry simulation (For material science, pharmacology, battery tech...)

# Potential applications of quantum computing

- ▶ Quantum chemistry simulation (For material science, pharmacology, battery tech...)
- ▶ Breaking cryptographic schemes

# Potential applications of quantum computing

- ▶ Quantum chemistry simulation (For material science, pharmacology, battery tech...)
- ▶ Breaking cryptographic schemes
- ▶ Optimization?

# Potential applications of quantum computing

- ▶ Quantum chemistry simulation (For material science, pharmacology, battery tech...)
- ▶ **Breaking cryptographic schemes!**
- ▶ Optimization?

Which cryptographic schemes  
are quantum-vulnerable?

# Which schemes are quantum-vulnerable?

# Which schemes are quantum-vulnerable?

- ▶ RSA

# Which schemes are quantum-vulnerable?

- ▶ RSA
- ▶ Diffie-Hellman (ordinary and elliptic curve)

# Which schemes are quantum-vulnerable?

- ▶ RSA
  - ▶ Diffie-Hellman (ordinary and elliptic curve)
- =>All\* currently used public-key crypto!

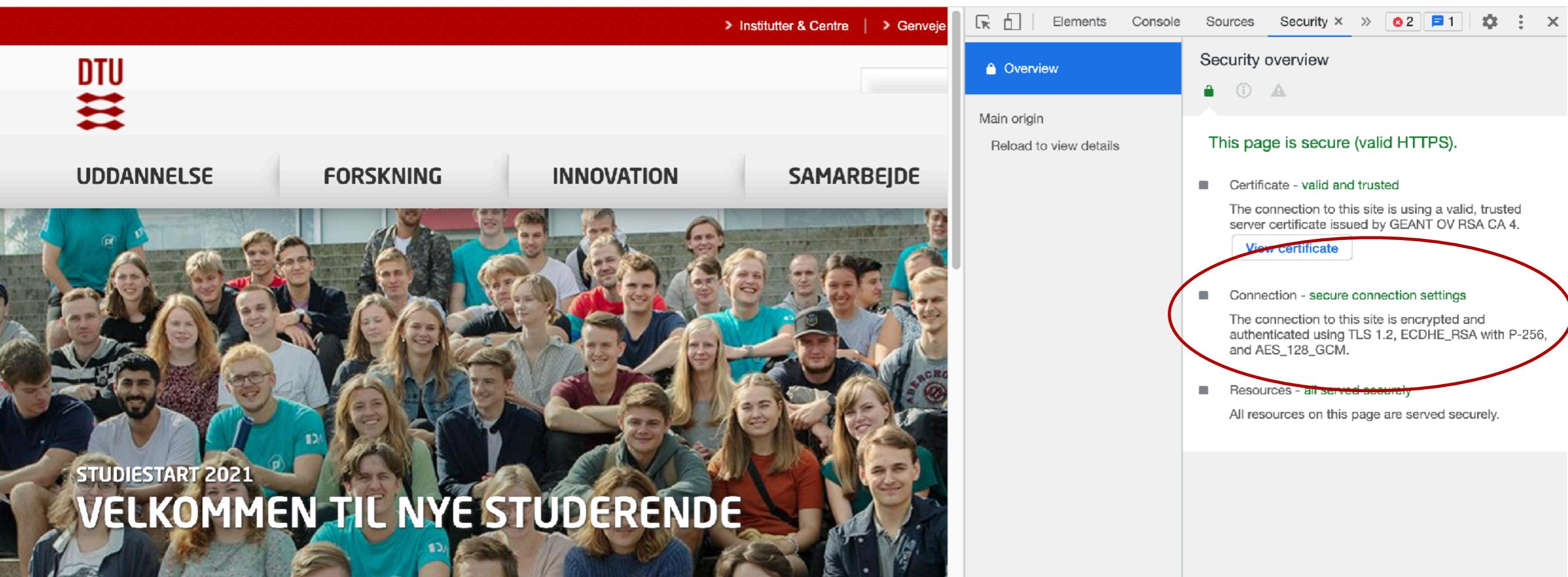


# Where are the broken schemes?

Discuss in pairs and collect examples: what technology you have used today that depends on quantum-broken crypto?

# Where is the broken stuff?

## Example: TLS



The screenshot shows a web browser displaying the DTU website. The left side of the screen shows the homepage with a large group photo of students and the text "STUDIESTART 2021" and "VÆLKOMMEN TIL NYE STUDERENDE". The right side shows the browser's security overview panel, which is titled "This page is secure (valid HTTPS)". The panel details the secure connection settings, including a valid and trusted certificate issued by GEANT OV RSA CA 4, and secure connection settings using TLS 1.2, ECDHE\_RSA with P-256, and AES\_128\_GCM. A red oval highlights the "Connection - secure connection settings" section.

DTU

UDDANNELSE FORSKNING INNOVATION SAMARBEJDE

STUDIESTART 2021  
VÆLKOMMEN TIL NYE STUDERENDE

Elements Console Sources Security x 2 1

Overview

Main origin Reload to view details

Security overview

This page is secure (valid HTTPS).

- Certificate - valid and trusted  
The connection to this site is using a valid, trusted server certificate issued by GEANT OV RSA CA 4.  
[View certificate](#)
- Connection - secure connection settings  
The connection to this site is encrypted and authenticated using TLS 1.2, ECDHE\_RSA with P-256, and AES\_128\_GCM.
- Resources - all served securely  
All resources on this page are served securely.

# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106

TLS (TLS 1.2, ECDHE\_RSA  
with P-256, and AES\_128\_GCM)  <https://>

# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106

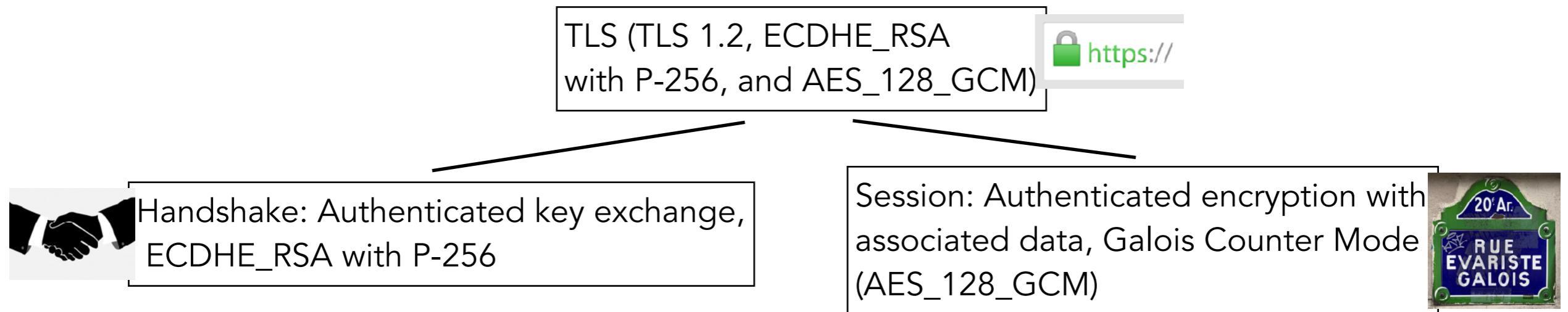
TLS (TLS 1.2, ECDHE\_RSA  
with P-256, and AES\_128\_GCM)



Handshake: Authenticated key exchange,  
ECDHE\_RSA with P-256

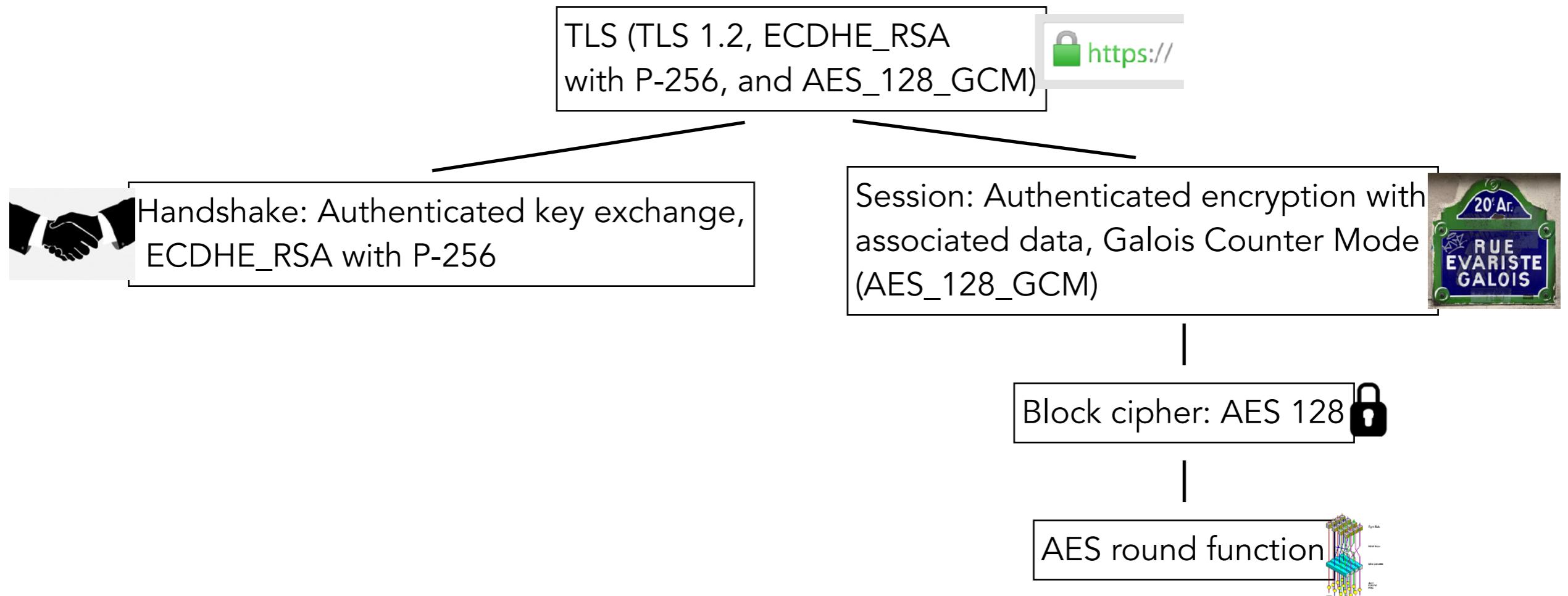
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



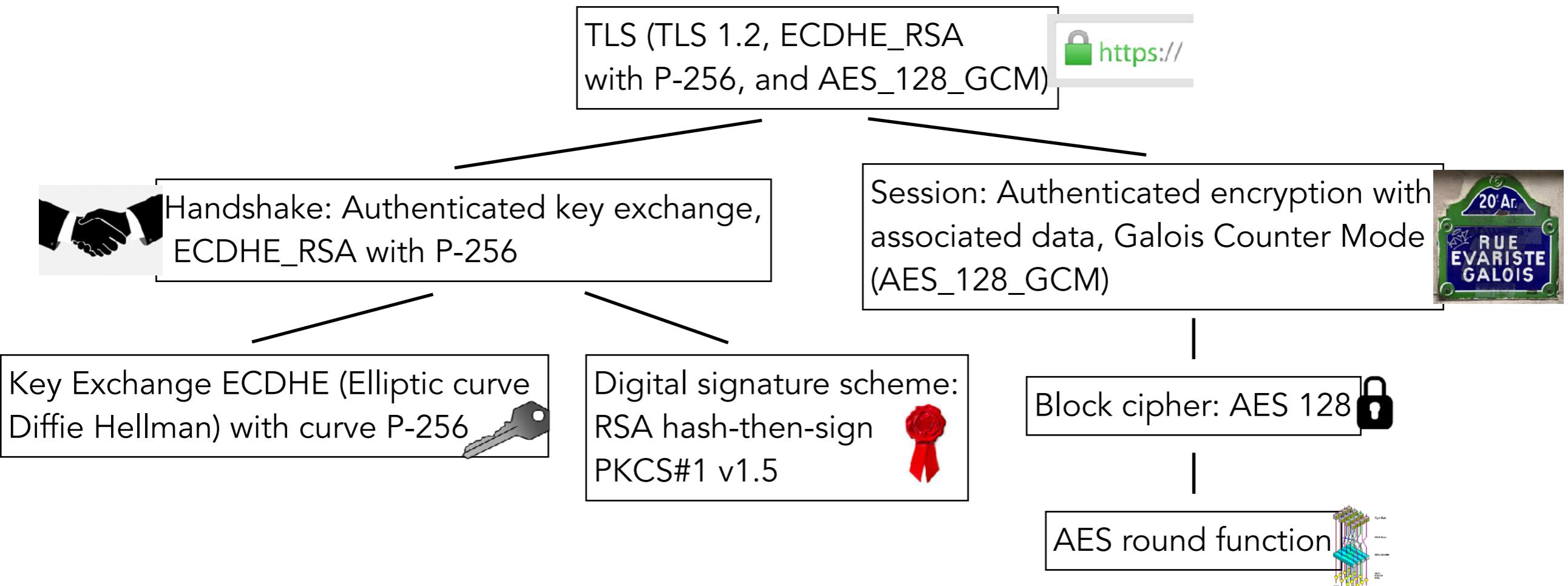
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



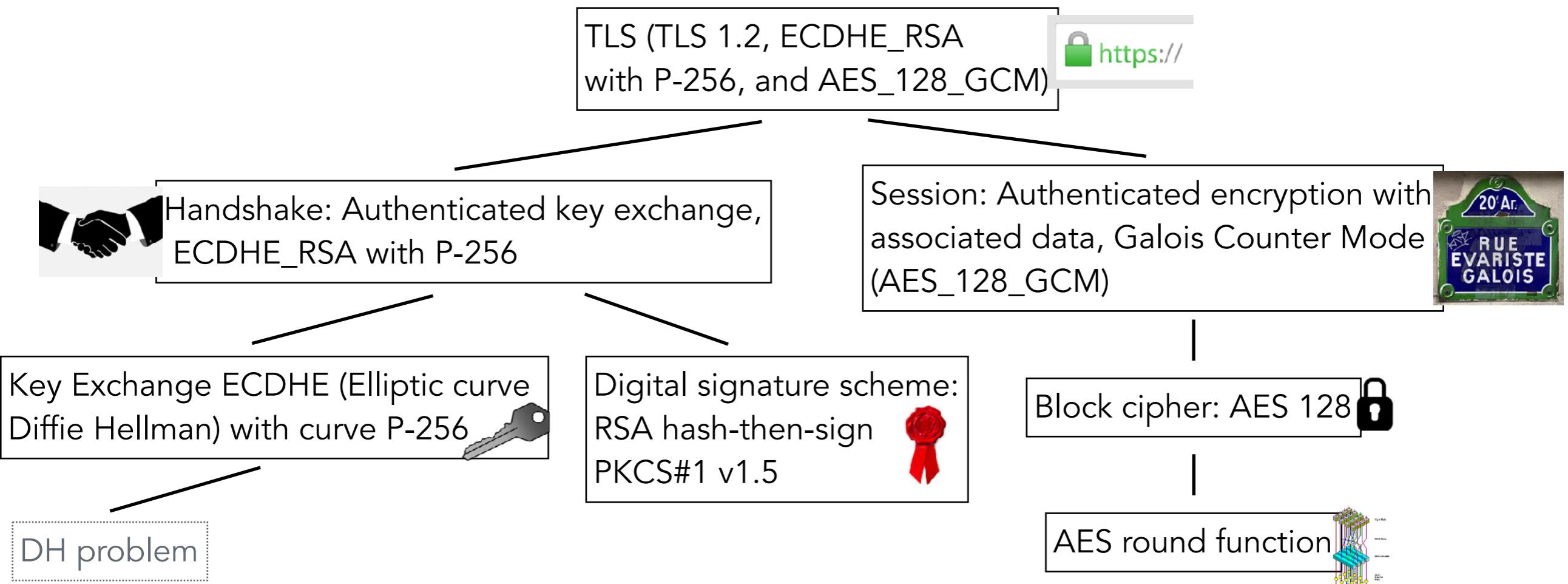
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



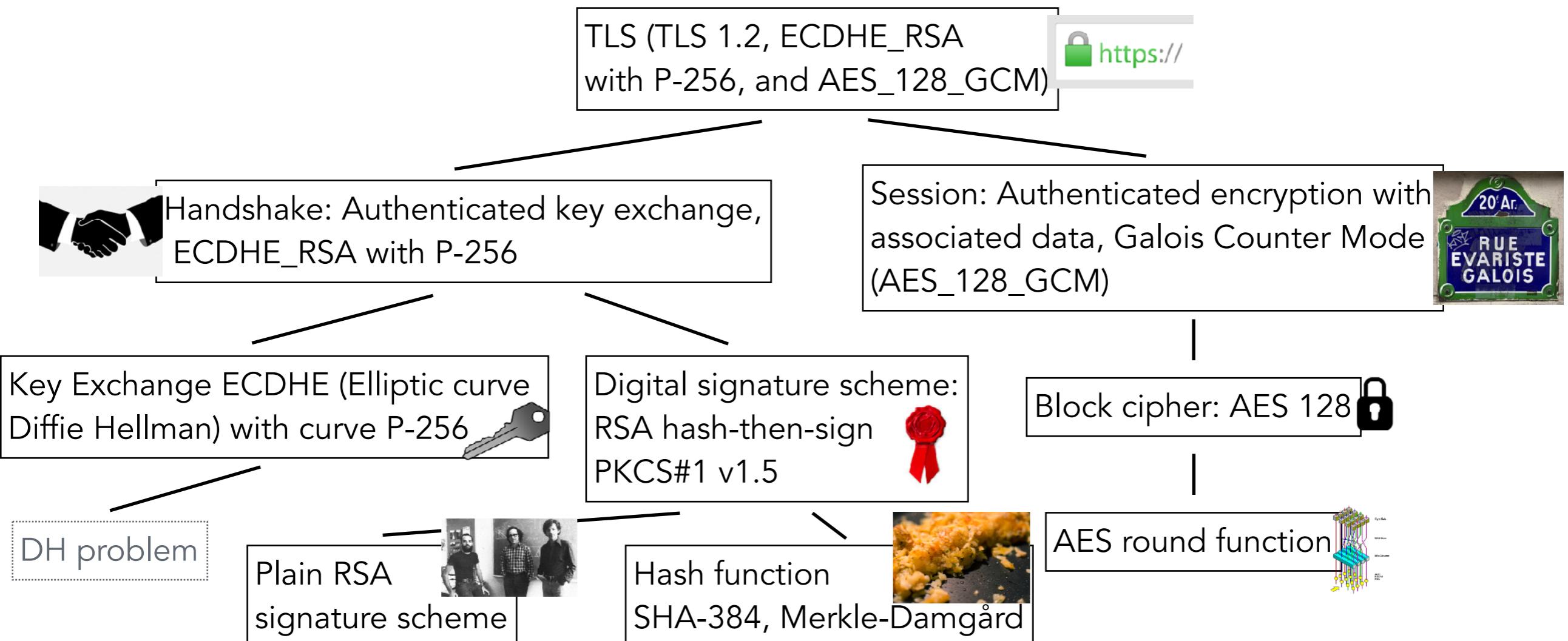
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



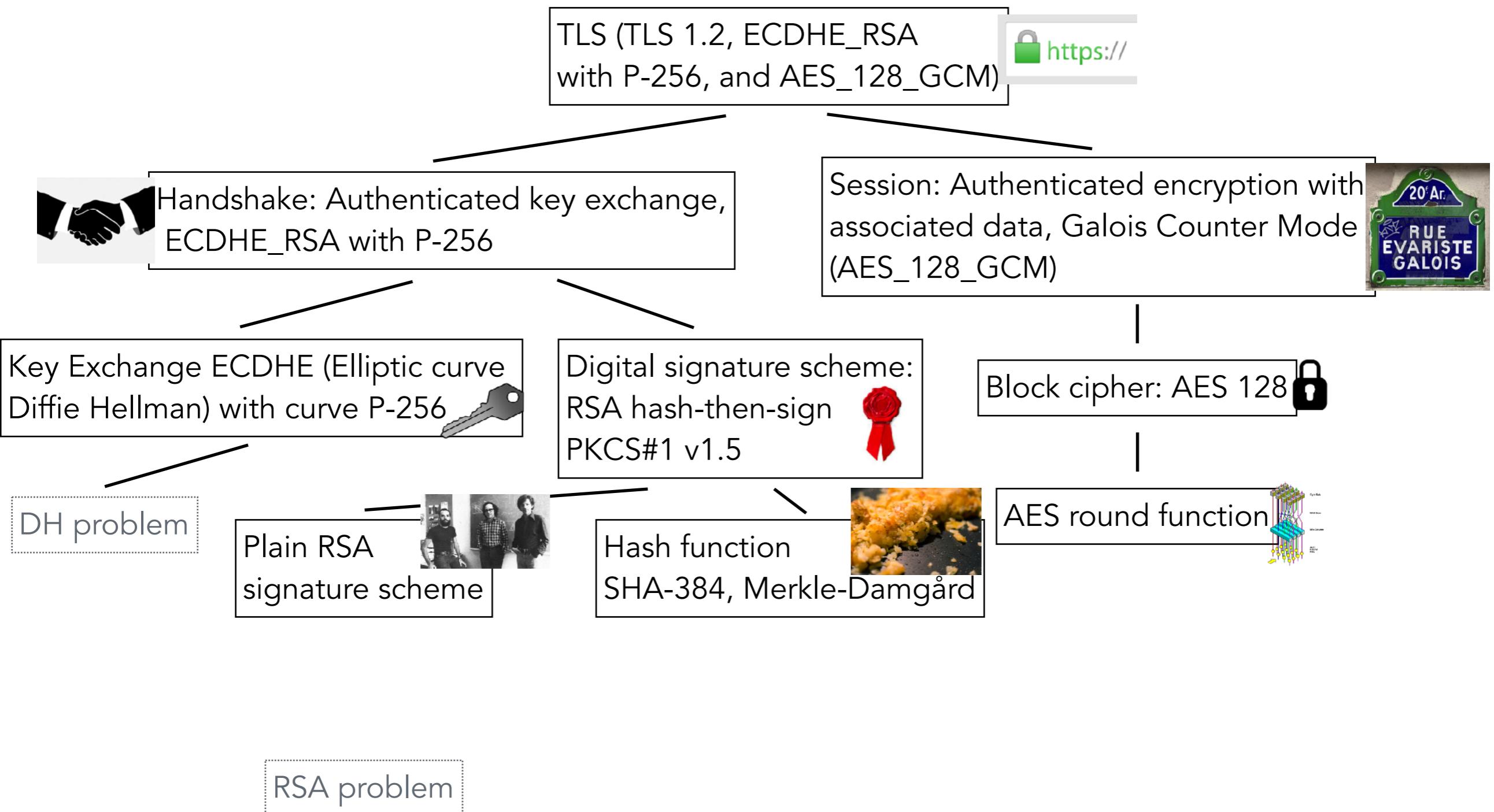
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



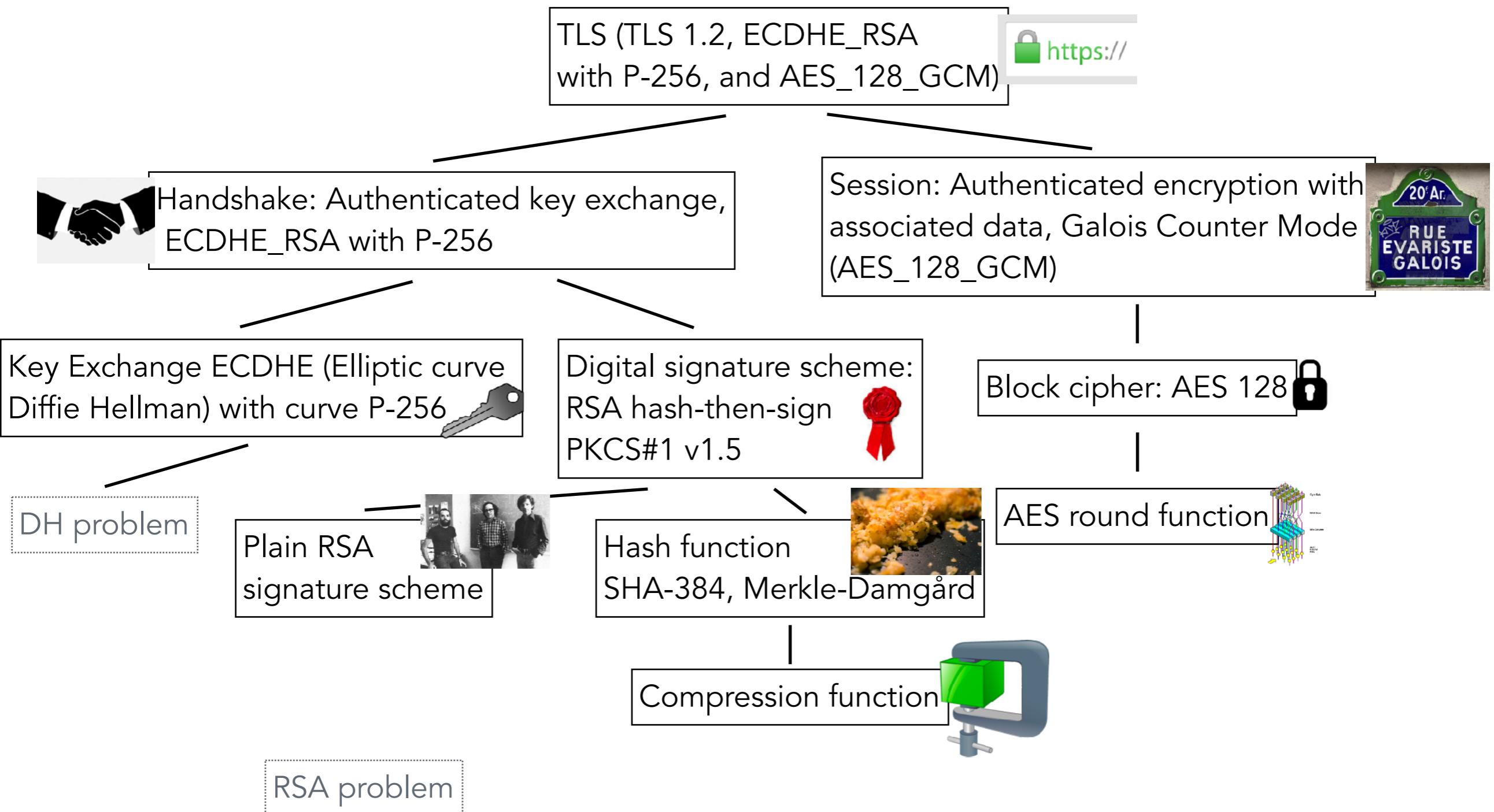
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



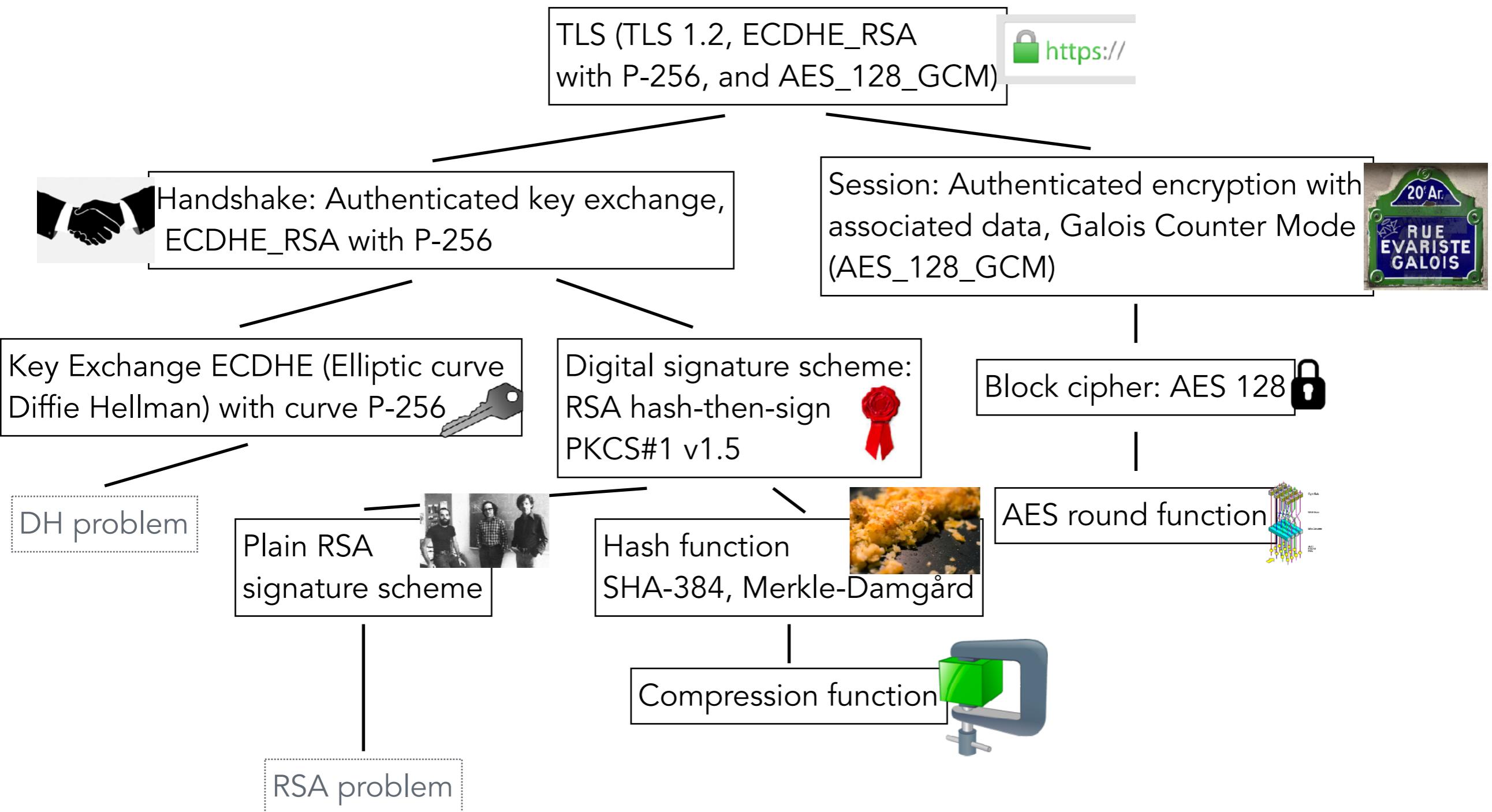
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



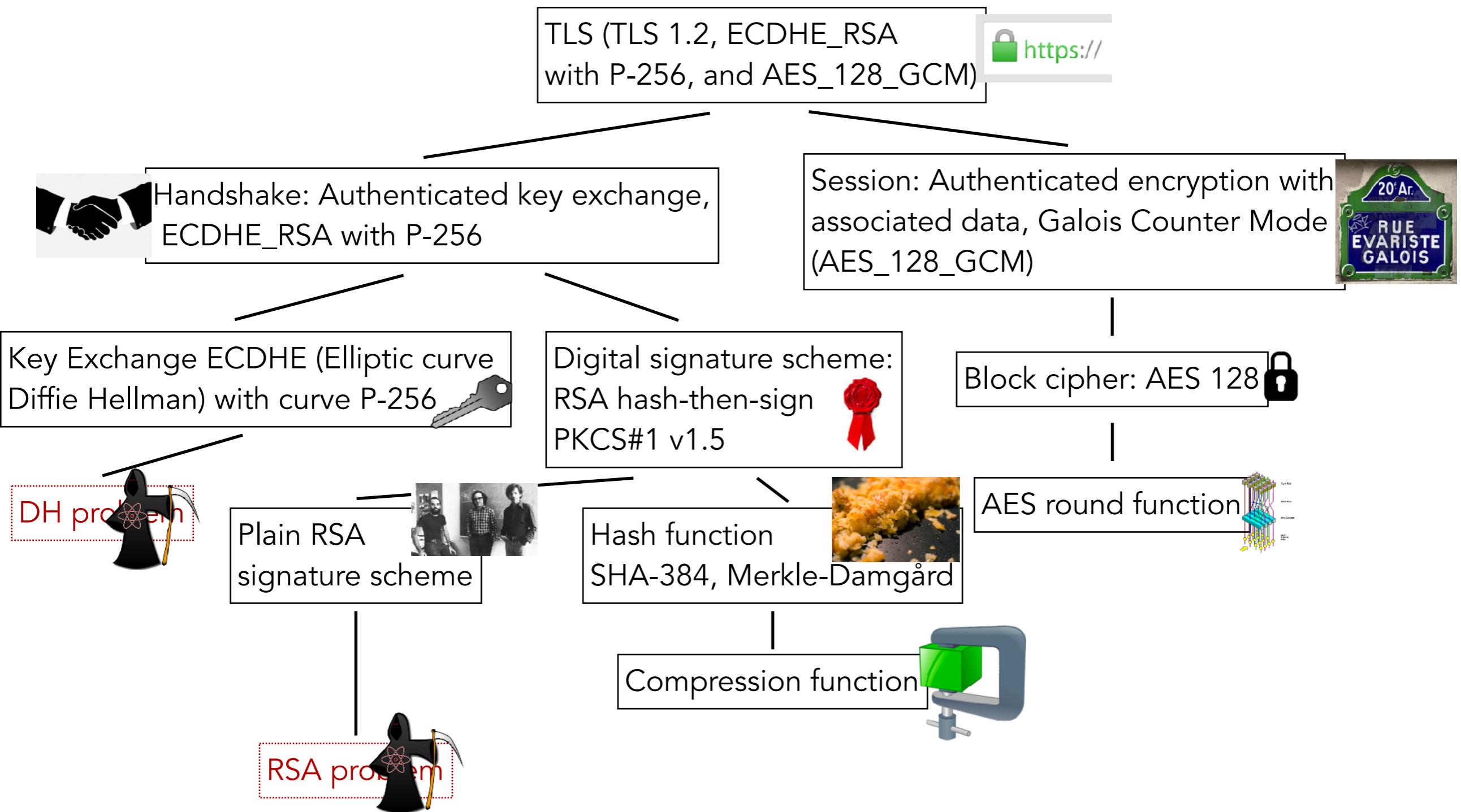
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



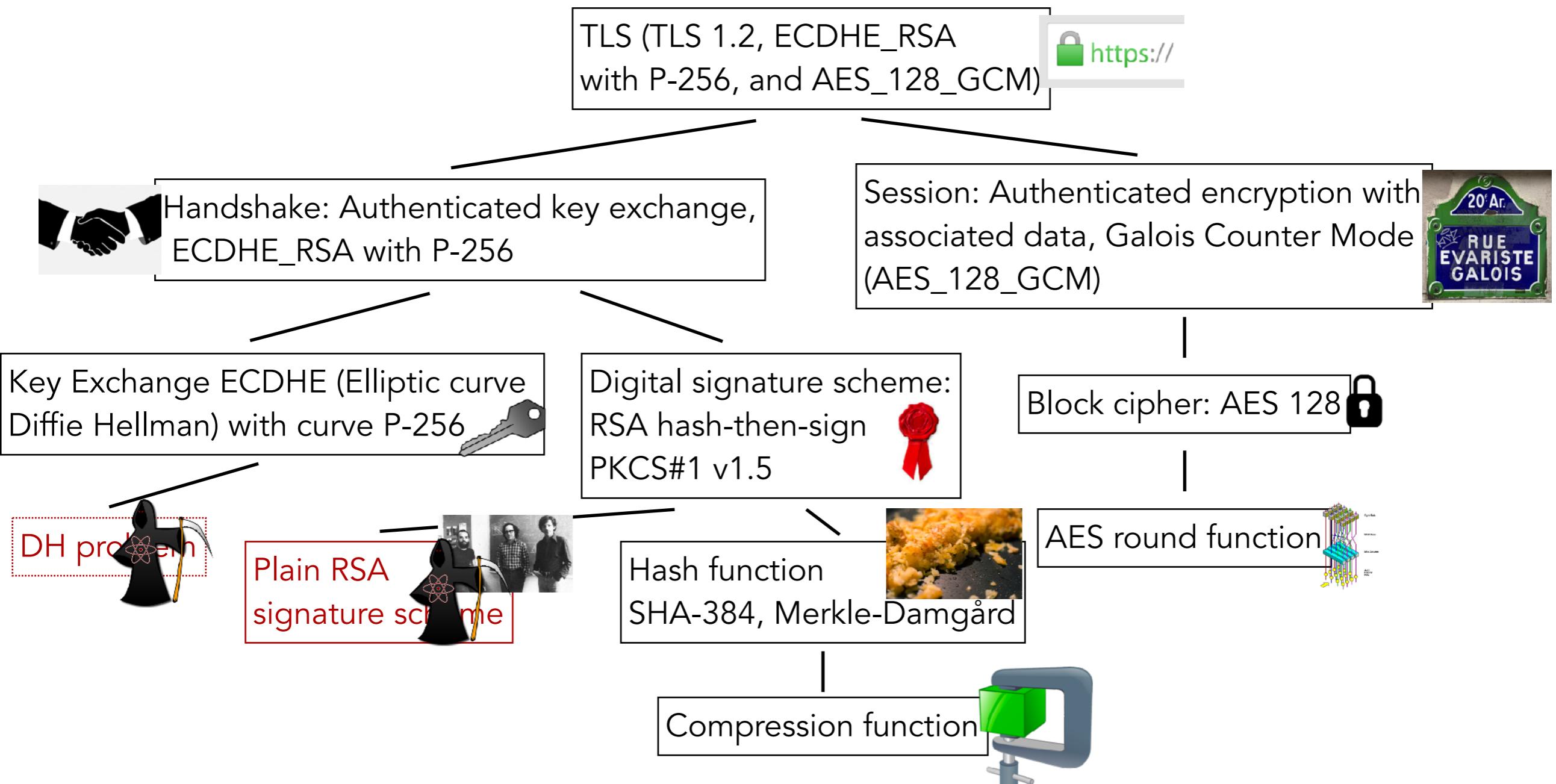
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



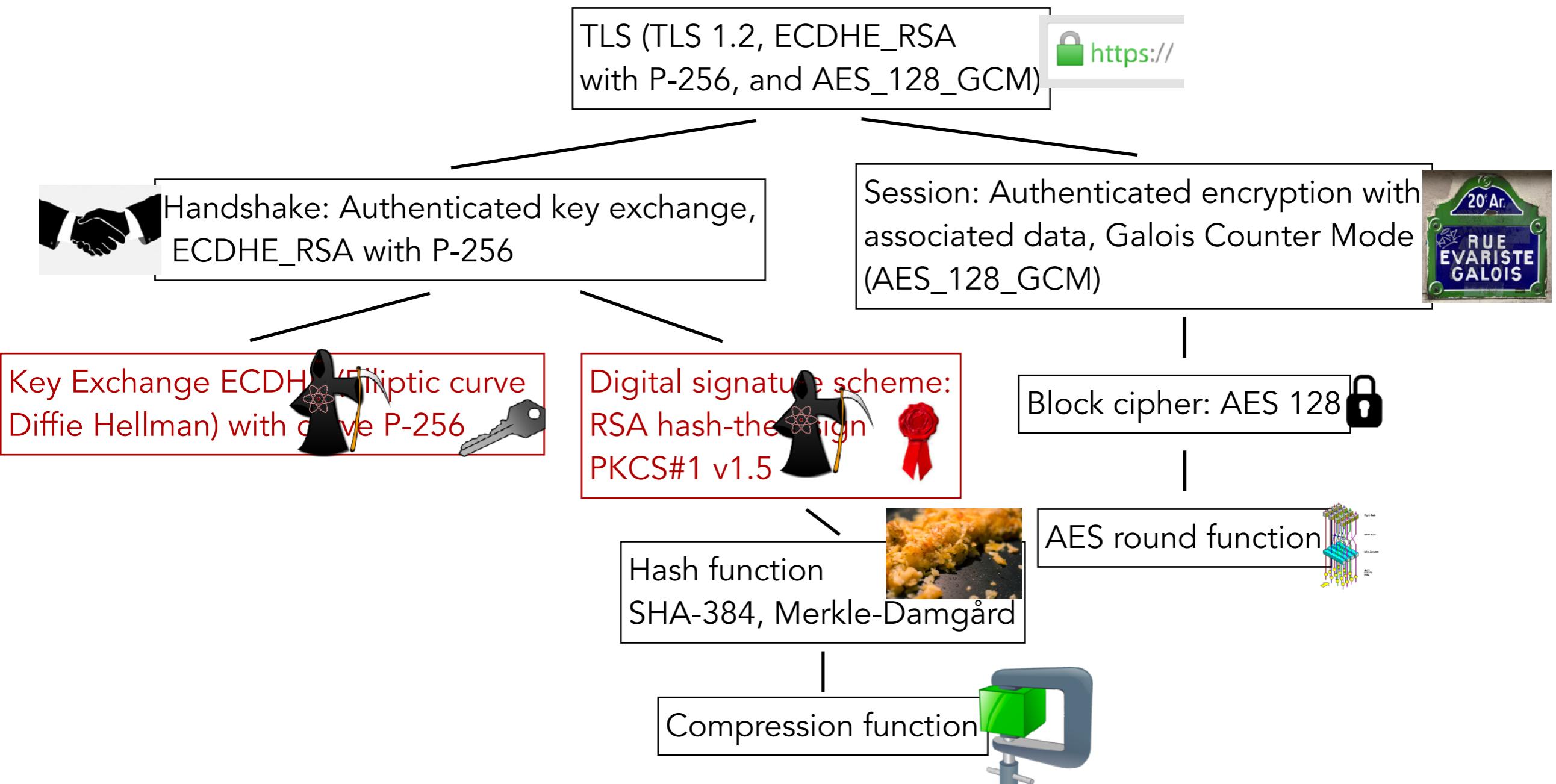
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



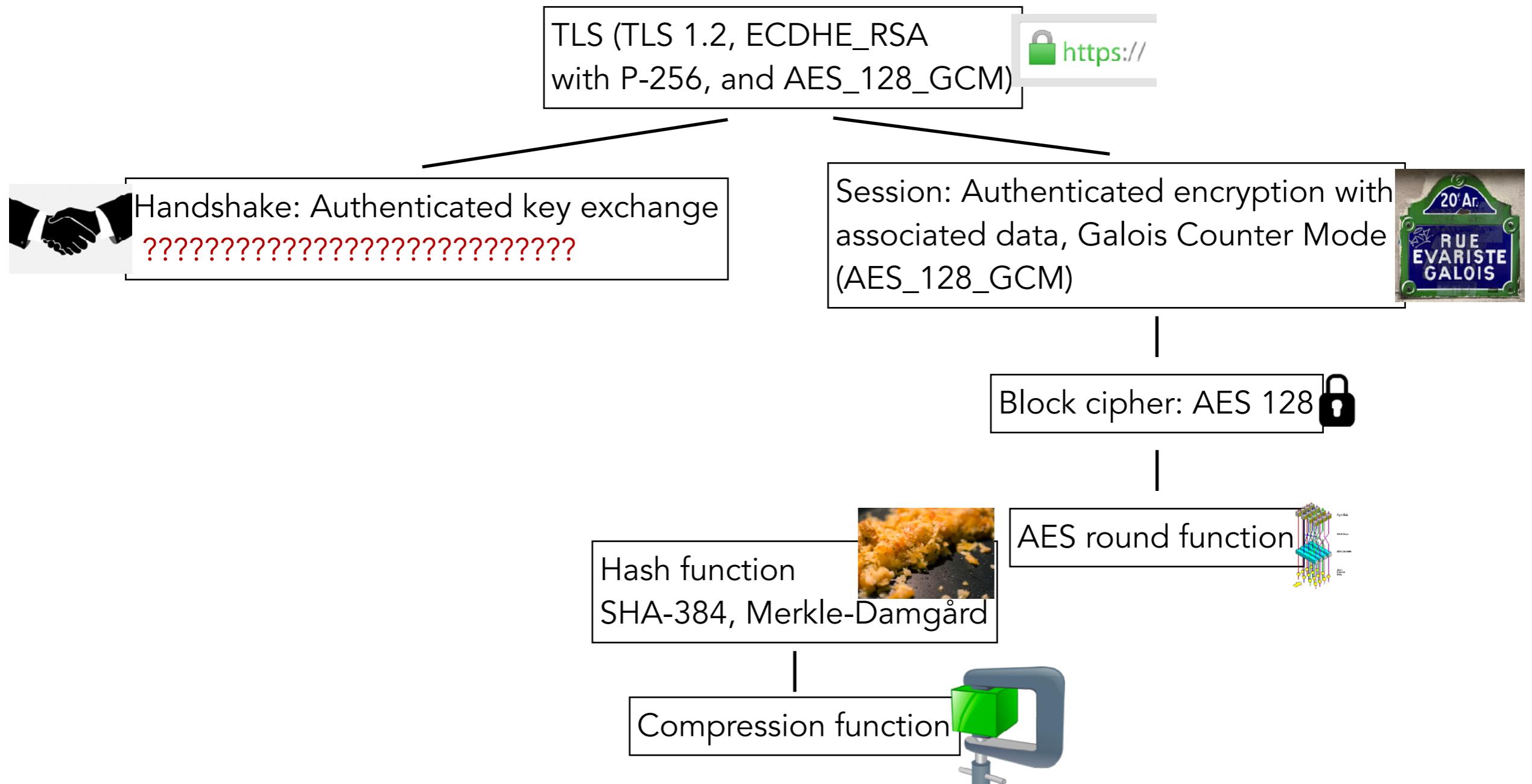
# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



# Quantum vulnerabilities in TLS

## Breakdown of TLS used by dtu.dk and Brave 1.28.106



How does a quantum  
computer break crypto?

# Example: RSA

- ▶ Public-key cryptosystem: encryption and digital signature
- ▶ Based on number theory
- ▶ Used e.g. in HTTPS, Certificates...
- ▶ Can be broken via *factoring*

# Factoring

# Factoring

- ▶ Multiplying is easy.

# Factoring

- ▶ Multiplying is easy.

$$7 \cdot 11$$

# Factoring

- ▶ Multiplying is easy.

$$7 \cdot 11$$

$$47 \cdot 53$$

# Factoring

- ▶ Multiplying is easy.

$$7 \cdot 11$$

$$47 \cdot 53$$

5201819300953669437777897868400987444737 · 27505893044478565097389357278392740789

# Factoring

- ▶ Multiplying is easy.
- ▶ Factoring is hard!

$$7 \cdot 11$$

$$47 \cdot 53$$

5201819300953669437777897868400987444737 · 27505893044478565097389357278392740789

# Factoring

- ▶ Multiplying is easy.
- ▶ Factoring is hard!

$$7 \cdot 11$$

$$47 \cdot 53$$

5201819300953669437777897868400987444737 · 27505893044478565097389357278392740789

$$55 = ? \cdot ?$$

# Factoring

- ▶ Multiplying is easy.
- ▶ Factoring is hard!

$$7 \cdot 11$$

$$47 \cdot 53$$

5201819300953669437777897868400987444737 · 27505893044478565097389357278392740789

$$55 = ? \cdot ?$$

$$143 = ? \cdot ?$$

# Factoring

- ▶ Multiplying is easy.
- ▶ Factoring is hard!

$$7 \cdot 11$$

$$47 \cdot 53$$

5201819300953669437777897868400987444737 · 27505893044478565097389357278392740789

$$55 = ? \cdot ?$$

$$143 = ? \cdot ?$$

143080685328735887915213203008099413269667159770855033244081195723311103277493 = ? \cdot ?

# Factoring

- ▶ Multiplying is easy.
- ▶ Factoring is hard?

$$7 \cdot 11$$

$$47 \cdot 53$$

5201819300953669437777897868400987444737 · 27505893044478565097389357278392740789

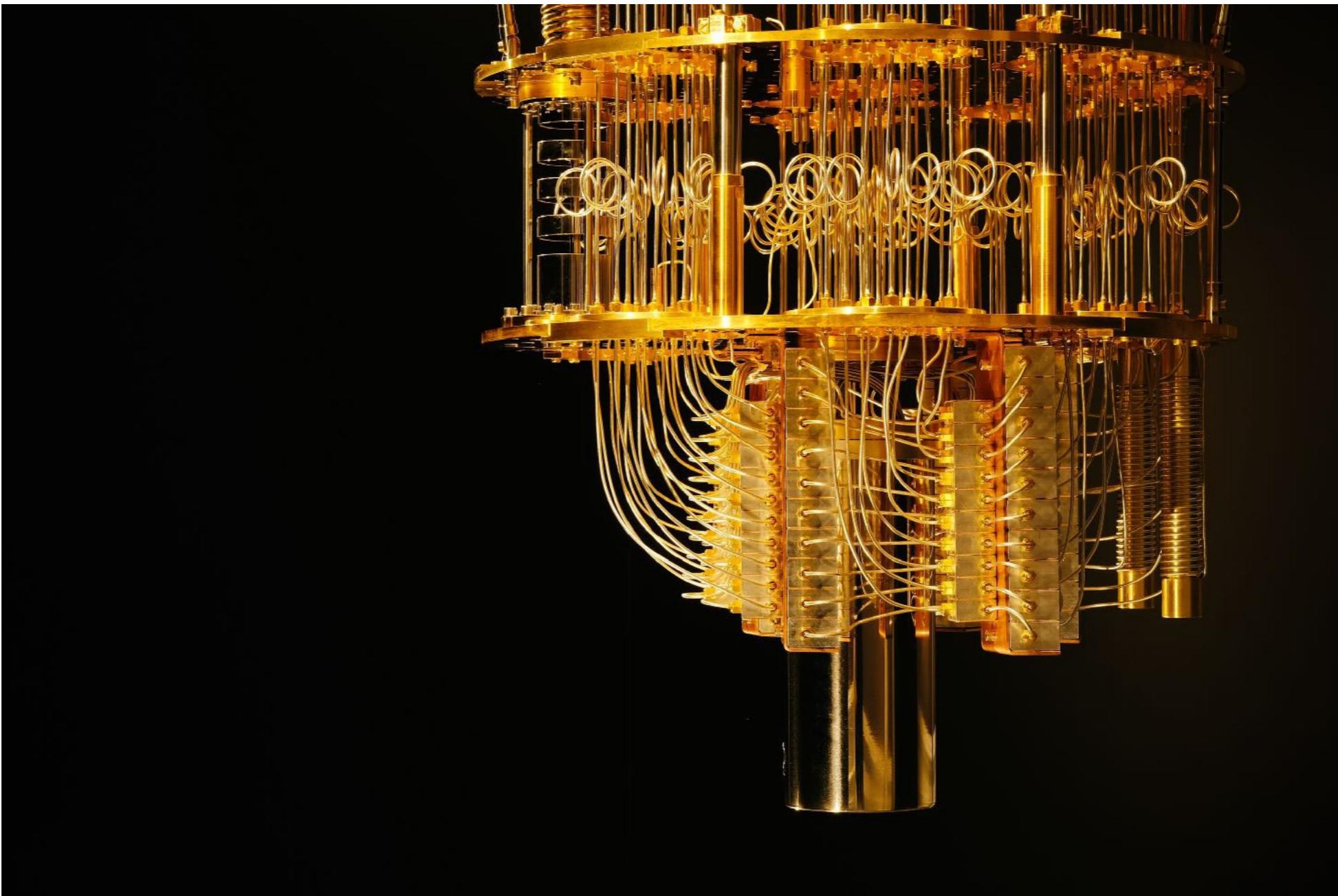
$$55 = ? \cdot ?$$

$$143 = ? \cdot ?$$

143080685328735887915213203008099413269667159770855033244081195723311103277493 = ? \cdot ?

# Turns out that...

...it is not hard for quantum computers.



# Factoring

How a quantum computer factors integers:

Problem: given  $N$  not prime, find  $q \notin \{1, N\}$  such that  $q | N$

Algorithm:

- ▶ Pick a random  $a$  with  $1 < a < N$
- ▶ Compute  $\gcd(a, N)$ . If it's  $\neq 1$  we found a factor (Yay!)
- ▶ Otherwise, let  $f(x) = a^x \pmod{N}$ , find  $r < N$  with  $f(x + r) = f(x)$
- ▶ hope that  $r$  is even and  $-1 \neq a^{r/2} \pmod{N}$
- ▶ Output  $\gcd(a^{r/2} - 1, N)$

# Factoring

How a quantum computer factors

Problem: given  $N$  not prime,

Algorithm:

- ▶ Pick a random  $a$  with  $1 < a < N$
- ▶ Compute  $\gcd(a, N)$ . If it's  $\neq 1$  we found a factor (Yay!)
- ▶ Otherwise, let  $f(x) = a^x \pmod{N}$ , **find  $r < N$  with  $f(x + r) = f(x)$ ???**
- ▶ hope that  $r$  is even and  $-1 \neq a^{r/2} \pmod{N}$
- ▶ Output  $\gcd(a^{r/2} - 1, N)$



The period-finding problem

# Period-finding via Fourier transform

- ▶ Otherwise, let  $f(x) = a^x \pmod{N}$  and find  $r$  with  $f(x + r) = f(x)????$

# Quantum Fourier transform

Cartoon slide (sorry!)

# Quantum Fourier transform

Cartoon slide (sorry!)

Classical Computer



Quantum Computer



# Quantum Fourier transform

Cartoon slide (sorry!)

Classical Computer

Bits!

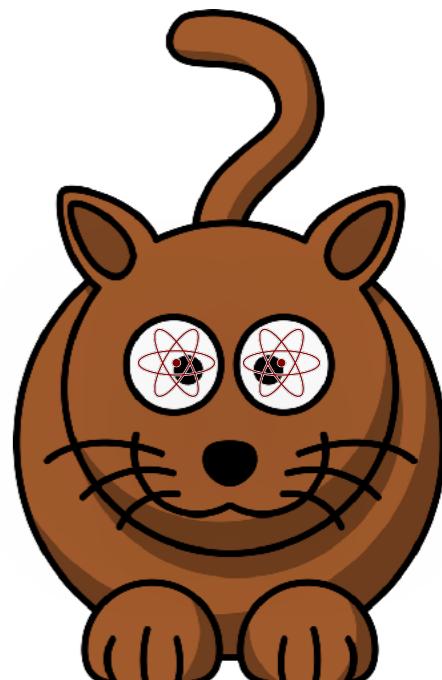
$$x \in \{0,1\}^n$$



Quantum Computer

Qubits!

$$|\psi\rangle \in \mathbb{C}^{2^n}$$



# Quantum Fourier transform

Cartoon slide (sorry!)

Classical Computer

Bits!

$$x \in \{0,1\}^n$$

Fourier sampling in  $\mathcal{O}(\ell)$ .



Quantum Computer

Qubits!

$$|\psi\rangle \in \mathbb{C}^{2^n}$$

Fourier sampling in  $\mathcal{O}(\log \ell)$ .



# Assessing quantum attack risk

# By when do we need to fix quantum-vulnerabilities?

Depends on:

How long do you need to keep your secrets secure?  
(*x* years)

How much time will it take to re-tool the existing infrastructure? (*y* years)

How long will it take for a large-scale quantum computer to be built? (*z* years)

Theorem (Mosca): If  $x + y > z$ , then worry.



# By when do we need to fix quantum-vulnerabilities?



- ▶ How big is y?
- ▶ How big is x?
- ▶ How big is z? Hard to say... ⇒ Exercise sheet!

# Summary: quantum computing threat

- ▶ Quantum computers are fundamentally different from regular computers
- ▶ Quantum computers are currently not available
- ▶ They have a computing power *incomparable* to regular computers
- ▶ They are good at discovering *structure* in big mathematical objects
- ▶ Once they are built, they can be used to launch attacks against RSA and Diffie-Hellman

# Post-quantum-secure public-key encryption from the Learning With Errors problem

Course 01410, Crypto I

Christian Majenz

Associate Professor, Cybersecurity Engineering Section, DTU Compute

# Plan for today

# NIST

Part I



$$\textcolor{red}{\square} \cdot \textcolor{green}{\square} = \textcolor{green}{\square} + \textcolor{yellow}{\square}$$

Part II

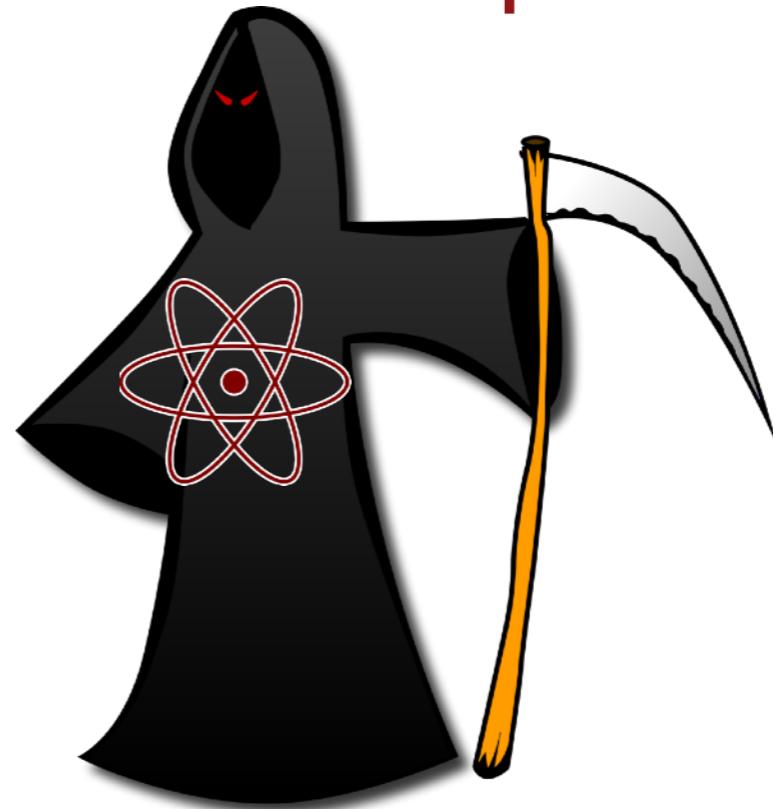
Part III

# Post-quantum public-key cryptography and the NIST competition

# Plenty of alternative hard problems

RSA

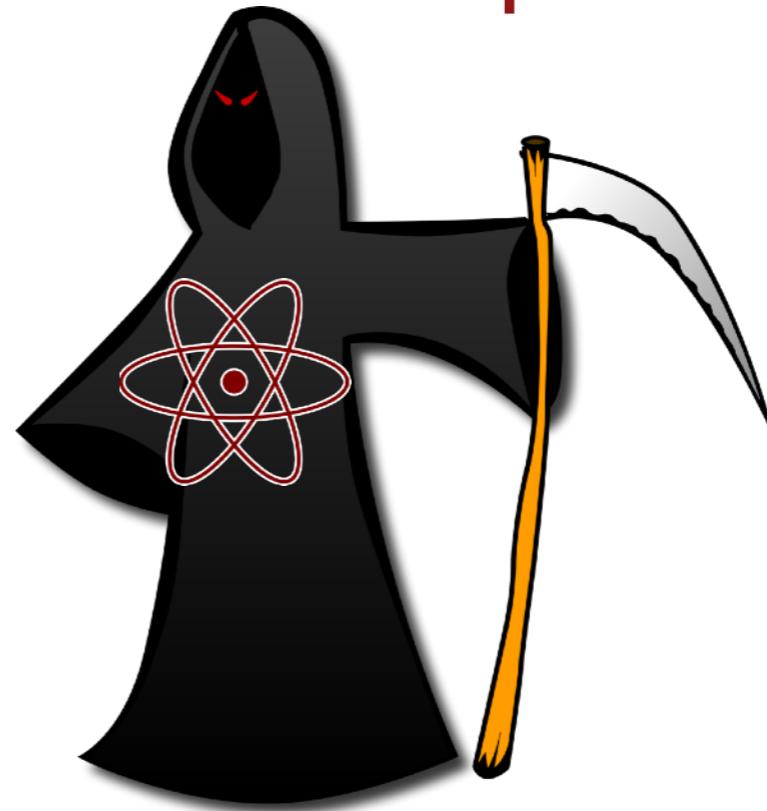
DH



# Plenty of alternative hard problems

RSA

DH

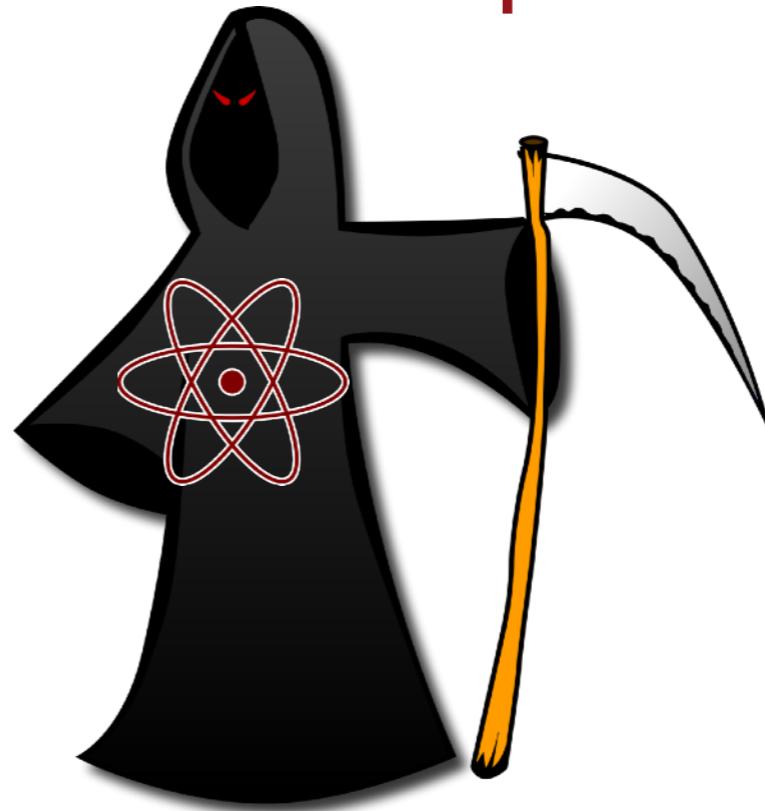


Luckily, public-key encryption and key exchange can also be constructed based on the hardness of

# Plenty of alternative hard problems

RSA

DH



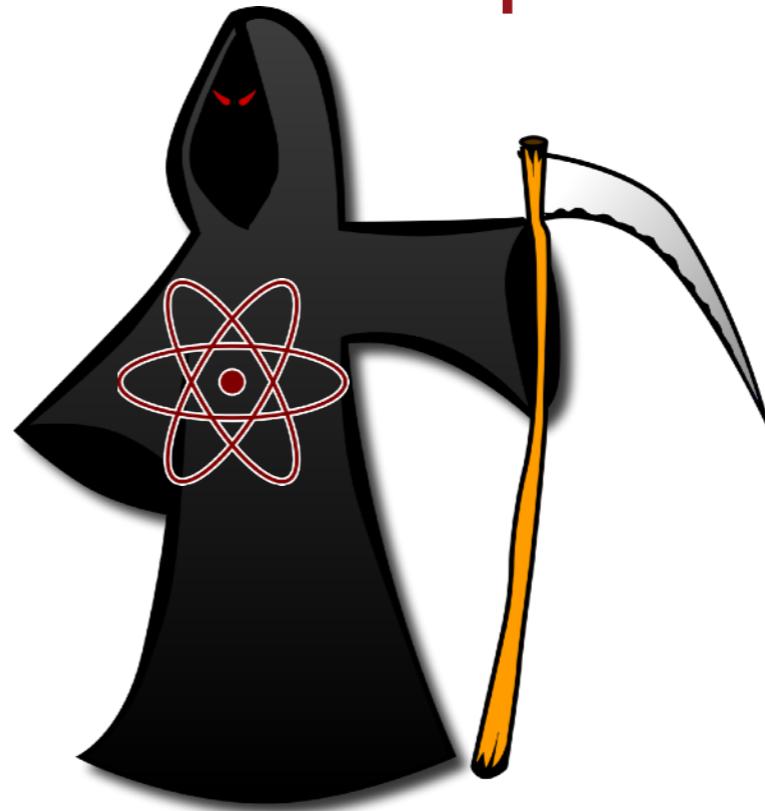
Luckily, public-key encryption and key exchange can also be constructed based on the hardness of

- ▶ Lattice problems

# Plenty of alternative hard problems

RSA

DH



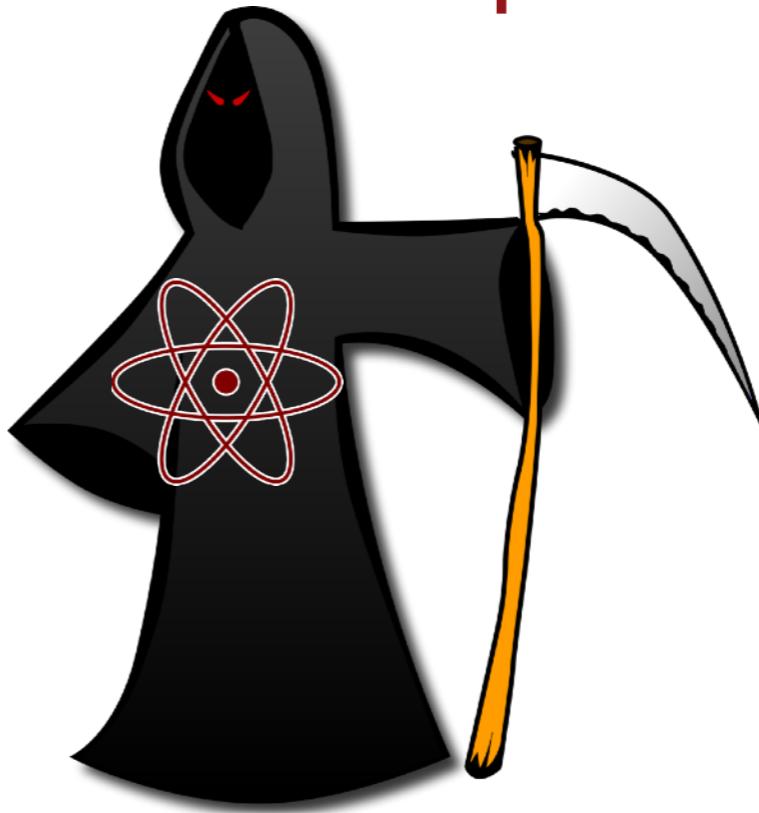
Luckily, public-key encryption and key exchange can also be constructed based on the hardness of

- ▶ Lattice problems
- ▶ Solving systems of multivariate polynomial equations

# Plenty of alternative hard problems

RSA

DH



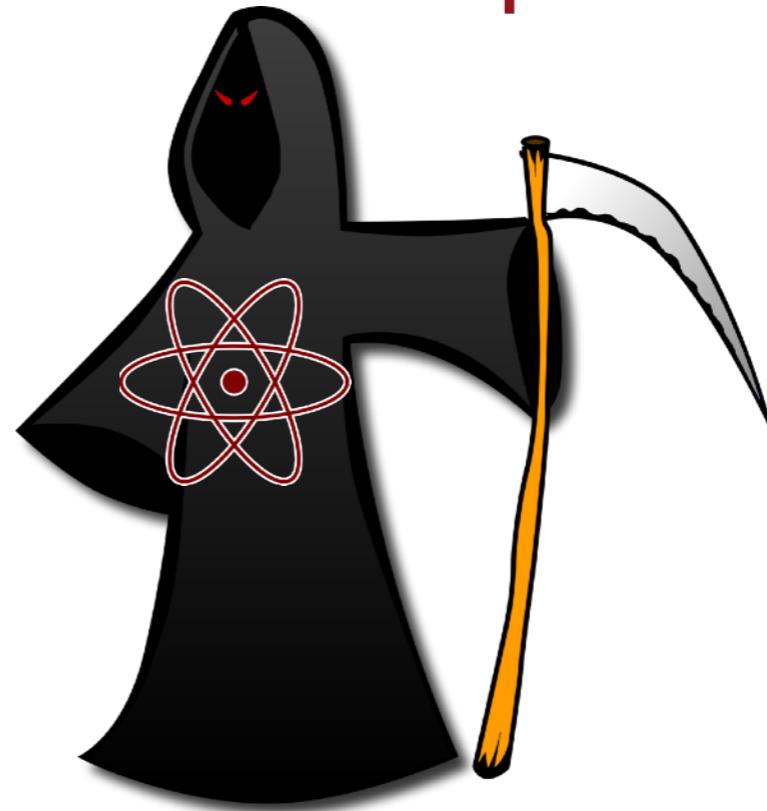
Luckily, public-key encryption and key exchange can also be constructed based on the hardness of

- ▶ Lattice problems
- ▶ Solving systems of multivariate polynomial equations
- ▶ Decoding “obfuscated” error correction codes

# Plenty of alternative hard problems

RSA

DH



Luckily, public-key encryption and key exchange can also be constructed based on the hardness of

- ▶ Lattice problems
- ▶ Solving systems of multivariate polynomial equations
- ▶ Decoding “obfuscated” error correction codes
- ▶ Finding an isogeny between supersingular elliptic curves

# Post-quantum cryptography

⇒ We have

- ▶ Lattice-based
- ▶ Multivariate-polynomial-based
- ▶ Code-based
- ▶ Isogeny-based

Public-key encryption/key exchange

Digital signatures are “easier”! Additionally

- ▶ Hash-based signatures
- ▶ MPC-in-the-head signatures

# NIST post-quantum crypto standardization

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:
  - ▶ Key Encapsulation Mechanism (KEM, ~public-key encryption)

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:
  - ▶ Key Encapsulation Mechanism (KEM, ~public-key encryption)
  - ▶ Digital Signature scheme (DSS)

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:
  - ▶ Key Encapsulation Mechanism (KEM, ~public-key encryption)
  - ▶ Digital Signature scheme (DSS)
- ▶ Initially: All classes represented

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:
  - ▶ Key Encapsulation Mechanism (KEM, ~public-key encryption)
  - ▶ Digital Signature scheme (DSS)
- ▶ Initially: All classes represented
- ▶ To be standardized:

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:
  - ▶ Key Encapsulation Mechanism (KEM, ~public-key encryption)
  - ▶ Digital Signature scheme (DSS)
- ▶ Initially: All classes represented
- ▶ To be standardized:
  - ▶ 1 Lattice KEM: Crystals-Kyber

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:
  - ▶ Key Encapsulation Mechanism (KEM, ~public-key encryption)
  - ▶ Digital Signature scheme (DSS)
- ▶ Initially: All classes represented
- ▶ To be standardized:
  - ▶ 1 Lattice KEM: Crystals-Kyber
  - ▶ 3 DSS: 2 based on lattices: Crystals-Dilithium&Falcon, 1 hash-based: SPHICS+

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:
  - ▶ Key Encapsulation Mechanism (KEM, ~public-key encryption)
  - ▶ Digital Signature scheme (DSS)
- ▶ Initially: All classes represented
- ▶ To be standardized:
  - ▶ 1 Lattice KEM: Crystals-Kyber
  - ▶ 3 DSS: 2 based on lattices: Crystals-Dilithium&Falcon, 1 hash-based: SPHICS+
- ▶ 4th round: New call for signature proposals, 3 code-based KEMs under consideration

# NIST post-quantum crypto standardization

- ▶ Goal: Standardize at least one PQ-secure scheme of each:
  - ▶ Key Encapsulation Mechanism (KEM, ~public-key encryption)
  - ▶ Digital Signature scheme (DSS)
- ▶ Initially: All classes represented
- ▶ To be standardized:
  - ▶ **1 Lattice KEM: Crystals-Kyber**
  - ▶ 3 DSS: 2 based on lattices: Crystals-Dilithium&Falcon, 1 hash-based: SPHICS+
- ▶ 4th round: New call for signature proposals, 3 code-based KEMs under consideration

# Remainder of today's lecture:

- ▶ The Learning With Errors (LWE) problem
- ▶ The Regev encryption scheme
- ▶ Short outlook: What's missing to get to Kyber?

# Part II: The Learning With Errors problem (LWE)

# Warm-up: Gaussian elimination

- ▶ Exercise: Solve the following linear system over  $\mathbb{Z}_{23}$  using Gaussian elimination:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} 13 \\ 20 \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \\ 1 \end{pmatrix} \pmod{23}$$

Hint:  $21^{-1} = 11 \pmod{23}$

# A slightly harder problem

- ▶ Exercise: Solve the following “noisy” linear system over  $\mathbb{Z}_{23}$  using Gaussian elimination:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} 9 \\ 10 \end{pmatrix} = \begin{pmatrix} 6 \\ 19 \\ 13 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix} \pmod{23}$$

Promise:  $\epsilon_i \in \{-1, 0, 1\}$  for  $i = 1, 2, 3$

Hint:  $21^{-1} = 11 \pmod{23}$

# A slightly harder problem

- Exercise: Solve the following “noisy” linear system over  $\mathbb{Z}_{23}$  using elimination:

$$\begin{pmatrix} 1 & 3 & 4 \\ 3 & 5 & 6 \\ 5 & 6 & 10 \end{pmatrix} \begin{pmatrix} 6 \\ \epsilon_1 \\ \epsilon_2 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix} \pmod{23}$$

Essentially the LWE problem!

Promise:  $\epsilon_i \in \{-1, 0, 1\}$  for  $i = 1, 2$

Hint:  $21^{-1} = 11 \pmod{23}$

# Part III: Regev encryption