CRYPTOGRAPHY

(lecture 5)

Literature:

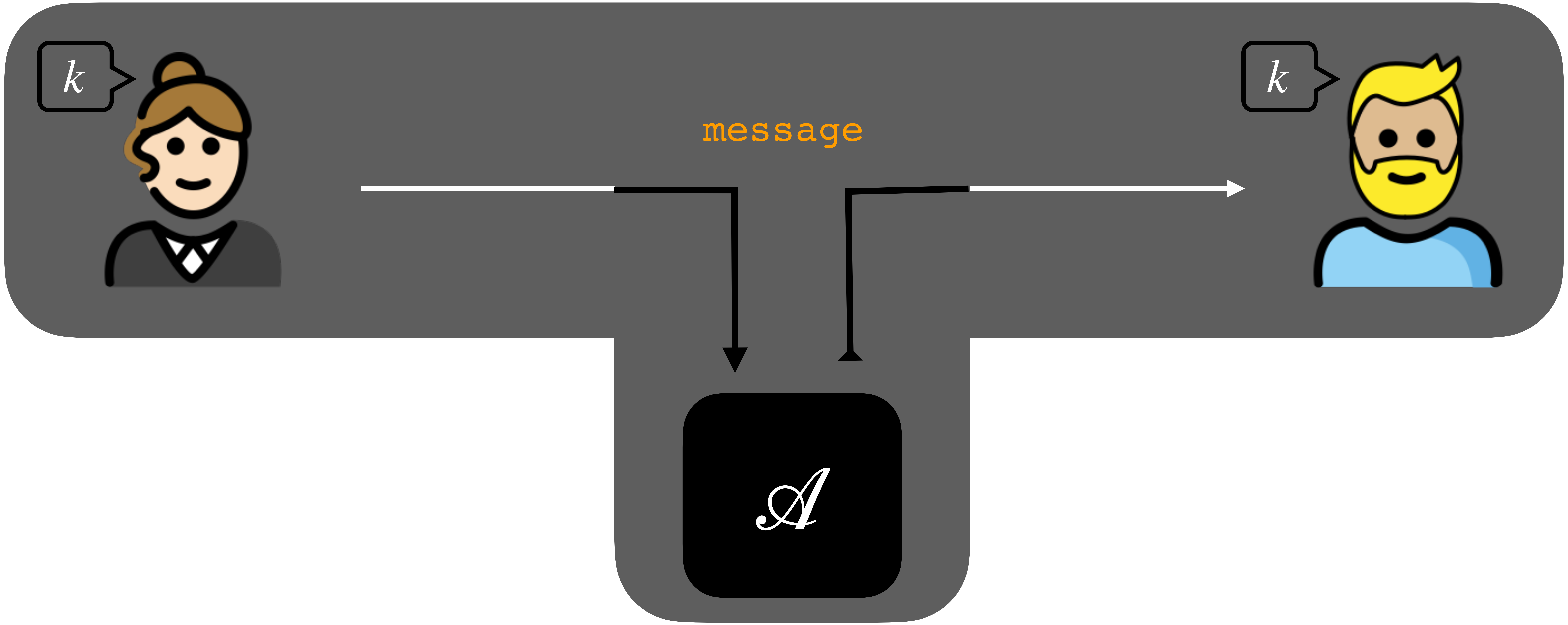
"Handbook of Applied Cryptography" (ch **3.6, 3.7, 3.9.1**)
"Lecture Notes on Cryptography" by Goldwasser and Bellare (ch **11.1, 10.1, 10.3.1**)
"A Graduate Course in Applied Cryptography" (ch **10.2.0, 10.4**, **10.5-10.5.1, 13.1**)
"Lecture Notes on Introduction to Cryptography" by V. Goyal (ch 6)
If you like cryptography, you should ready this paper once in your lifetime: [DH76]
Background on Number Theory (available on Canvas)

Announcements

- Deadline for submitting first draft of HA1 TODAY (end of the day)
- This Friday's lecture will be given by Ivan!
- If you have questions / doubts about HA1 come to me at the end of this lecture

…Back in Module 1…

- Perfectly secure encryption (**OTP**)
- Semantically secure encryption (**PRG**)
- IND-CPA secure encryption (**AES**, **Block Ciphers**)
- Integrity (**MAC**, **AEAD**)




I was here first!

Module 2: Agenda

Introduction to Public Key Cryptography
- The Core Idea
- One-Way Trapdoor Functions

Key-Exchange
- Problem Statement
- A Simple Solution
- Formalisation: **Group Theory**
- Diffie-Hellman Key Exchange (DH)

(Some) Hardness Assumptions
- DLog, CDH, DDH
- Reductions Between Problems

More on DH
- On the Bit Security of DH Keys
- Securing DH Keys
- Choosing Good Parameters
- MiM Attack

Digital Signatures
- Problem Statement
- Syntax
- ECDSA

Public Key Encryption
Much More on Digital Signatures
Secure Instant Messaging
Post Quantum Cryptography

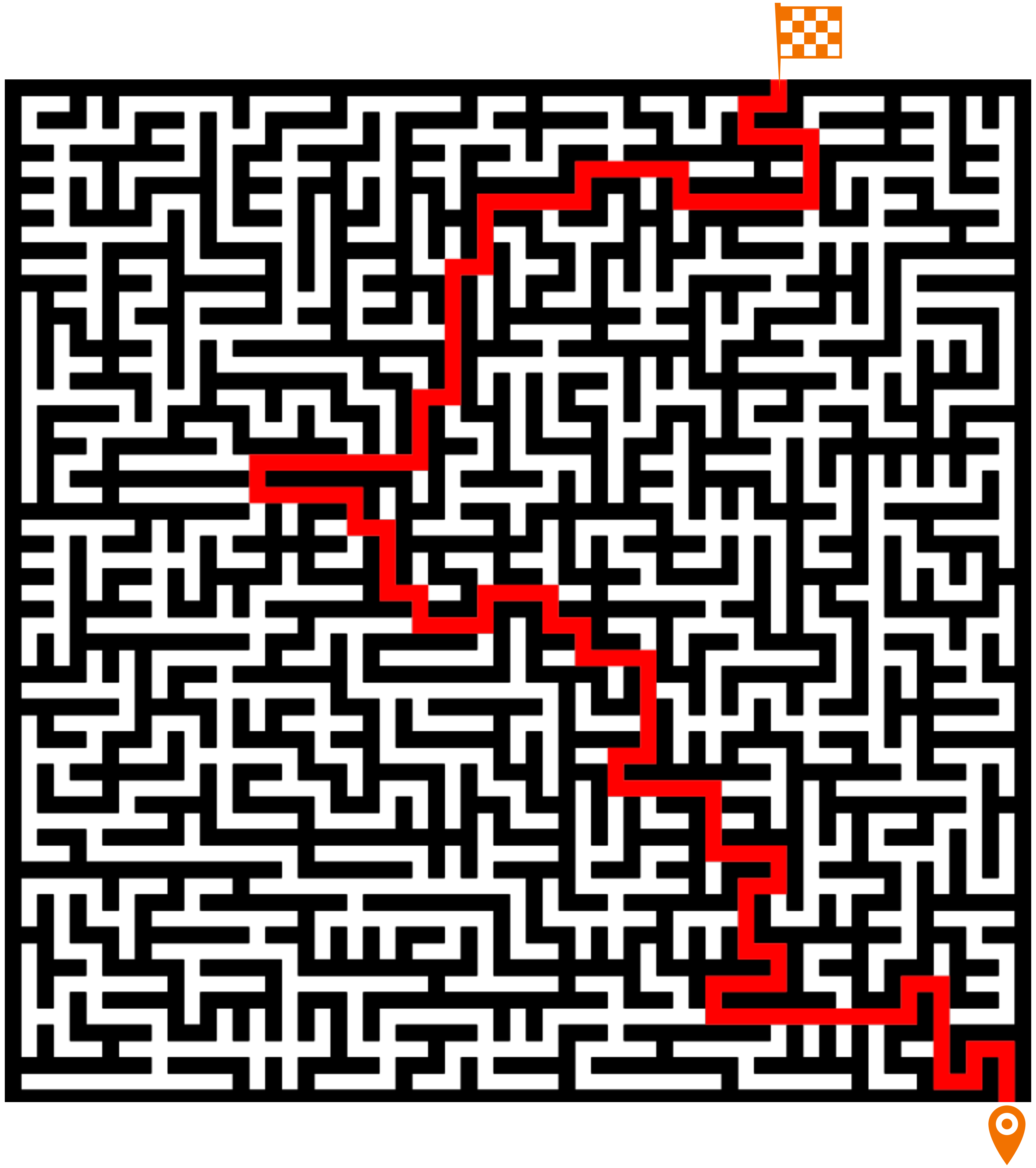
The One Fundamental Concept in Public Key Cryptography (PKC)

This is a **hard** problem

..unless..

🧐 *What does "hard" mean in cryptography?*

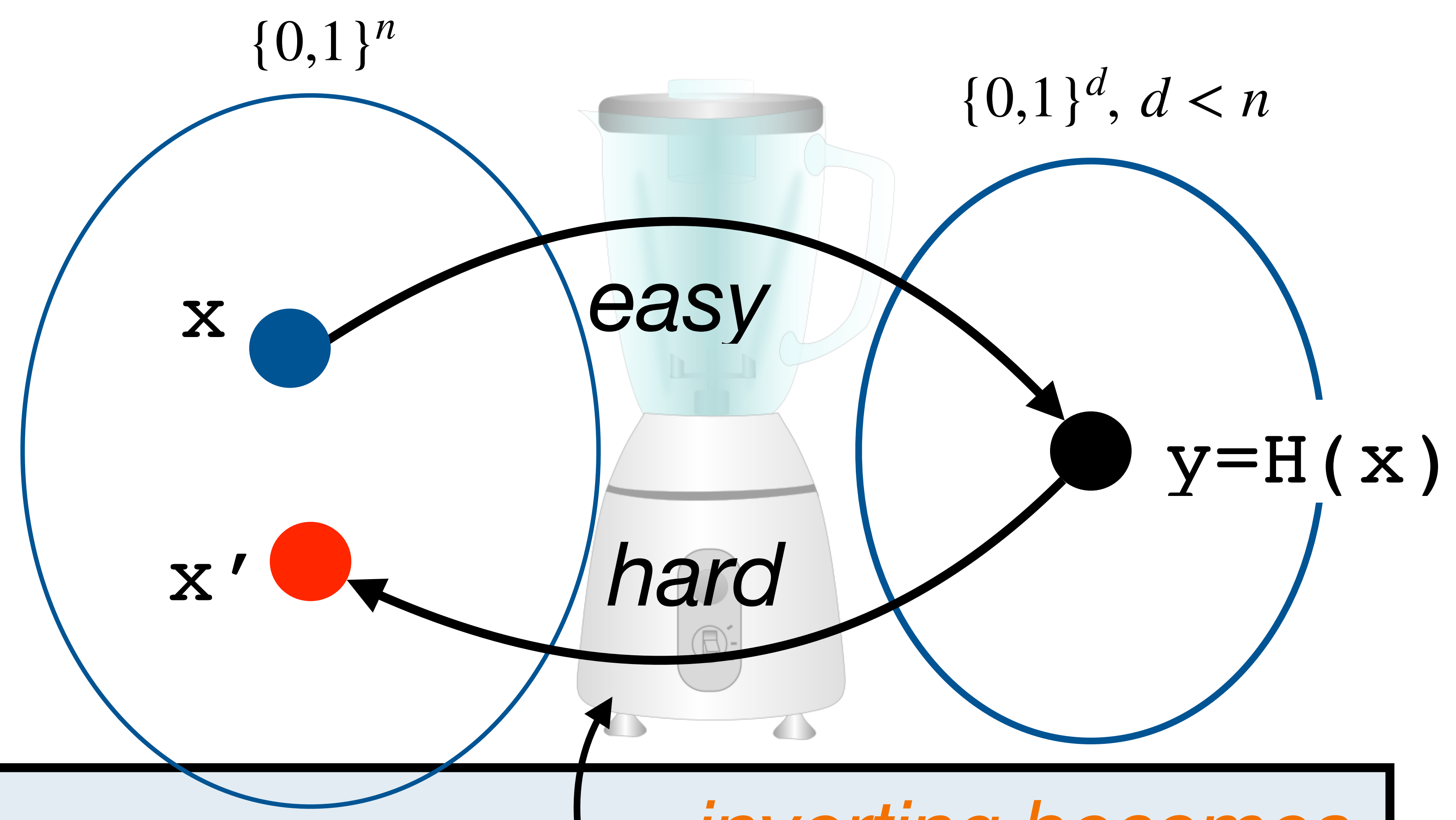
[the problem is solvable, but solving it requires time proportional to the age of our universe]



… you know some additional information that makes solving the problem easy! (**trapdoor**)

PKC is all about this 'efficiency gap' in solving a mathematical problem + tons of randomness

One-Way Trapdoor Function



inverting becomes easy for whoever knows the ***trapdoor***

<u>Definition: ONE-WAY TRAPDOOR FUNCTION (SCHEME)</u>

A trapdoor function scheme defined over two finite sets X, Y is a triple of PPT algorithms $(KeyGen, F, I)$ defined as follows:

- $KeyGen(n) \rightarrow (\mathsf{pk}, \mathsf{sk})$ is a probabilistic key generation algorithm
- For every **pk** output by *KeyGen*, $F(\mathsf{pk}, \cdot\,) : X \rightarrow Y$ is a deterministic algorithm $(F(\mathsf{pk}, x) = y)$
- For every **sk** output by *KeyGen*, $I(\mathsf{sk}, \cdot\,) : Y \rightarrow X$ is a deterministic algorithm $(I(\mathsf{sk}, y') = x')$

AND it holds that: $I(\mathsf{sk}, F(\mathsf{pk}, x)) = x$ for all keys generated by *KeyGen* and for all input $x \in X$.

sk is the **trapdoor**

One-Way Trapdoor Function - Security

$\mathscr{C}$

$KeyGen(n) \rightarrow (\mathsf{pk}, \mathsf{sk})$

$x \leftarrow \$X$

$F(\mathsf{pk}, x) = y$

$\longleftarrow (KeyGen, F, I) \longrightarrow$

$\longrightarrow (\mathsf{pk}, y) \longrightarrow$

$\longleftarrow x^* \longleftarrow$

$\mathscr{A}$

win or lose

$\mathscr{A}$ wins the game if $x^* = x$. If $x^* \neq x$, $\mathscr{A}$ loses.

A scheme $(KeyGen, F, I)$ is **one-way trapdoor** if for any PPT adversary $\mathscr{A}$ it holds that: $Adv(\mathscr{A}) = Pr[\mathscr{A}\ wins] < negl(n)$

This security game models the "one-way" property

The condition $I(\mathsf{sk}, F(\mathsf{pk}, x)) = x$ (from the previous slide) **models the "trapdoor" property**

An Example: RSA as a One-Way Trapdoor Function

- $KeyGen(n) \to (\mathrm{pk}, \mathrm{sk})$: Pick two large primes p,q (think 1024-bit long).
  Pick a random e $\leftarrow \$\mathbb{Z}_N^*$, compute its inverse d mod ϕ(N).
  Set pk=(N,e) and sk=(p,q,d)
- $F(\mathrm{pk}, \cdot\,) : \mathbb{Z}_N \to \mathbb{Z}_N$: given x $\in \mathbb{Z}_N$, return **y = x^{e} mod N**
- $I(\mathrm{sk}, \cdot\,) : \mathbb{Z}_N \to \mathbb{Z}_N$: given y $\in \mathbb{Z}_N$, return **x = y^{d} mod N**

Module 2: Agenda

Introduction to Public Key Cryptography

- The Core Idea
- One-Way Trapdoor Functions

Key-Exchange

- Problem Statement
- A Simple Solution
- Formalisation: **Group Theory**
- Diffie-Hellman Key Exchange (DH)

(Some) Hardness Assumptions

- DLog, CDH, DDH
- Reductions Between Problems

More on DH

- On the Bit Security of DH Keys
- Securing DH Keys
- Choosing Good Parameters
- MiM Attack

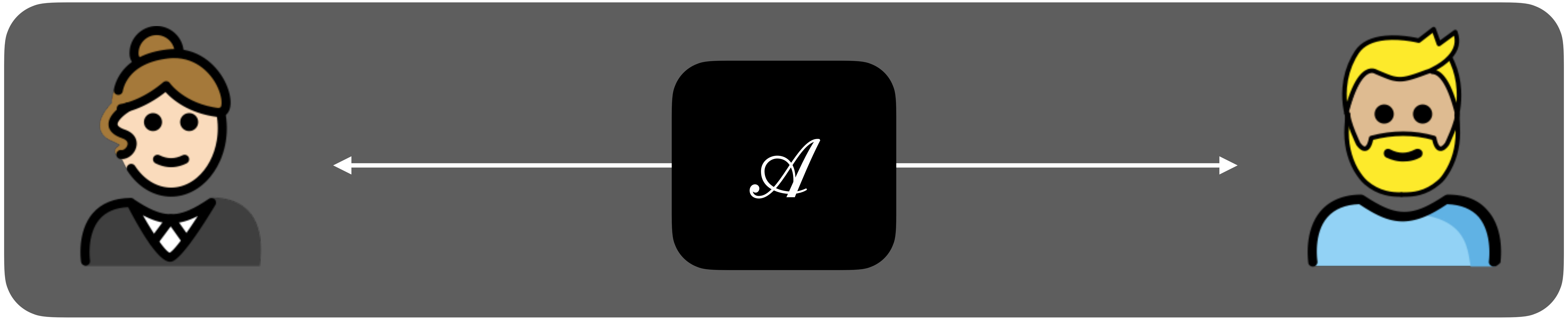
Digital Signatures

- Problem Statement
- Syntax
- ECDSA

Public Key Encryption
Much More on Digital Signatures
Secure Instant Messaging
Post Quantum Cryptography

Problem Statement

Alice and Bob want to find a way to share a secret key k without relying on a previously shared secret **AND** they want to do so, using a public channel, that is monitored* by the Adversary



Tool: Exponentiation g^x

**For the sake of this lecture, we only consider passive $\mathscr{A}$ (eavesdropper)*

A Simple Solution

Pick a number g

Pick another number a

Compute the exponentiation

$A = g^a$

Compute the shared secret

$K = B^a$

$\xrightarrow{(g,\ A)}$

$\xleftarrow{B}$

Pick a number b

Compute $B = g^b$

Compute the shared secret

$K = A^b$

Why is K the same for both?

What prevents $\mathscr{A}$ from learning K given (g,A,B)?

In theory (=unconditionally) nothing!

In practice, this challenge is (computationally) hard to solve, **if** you work in the correct *domain*

Let’s get formal!

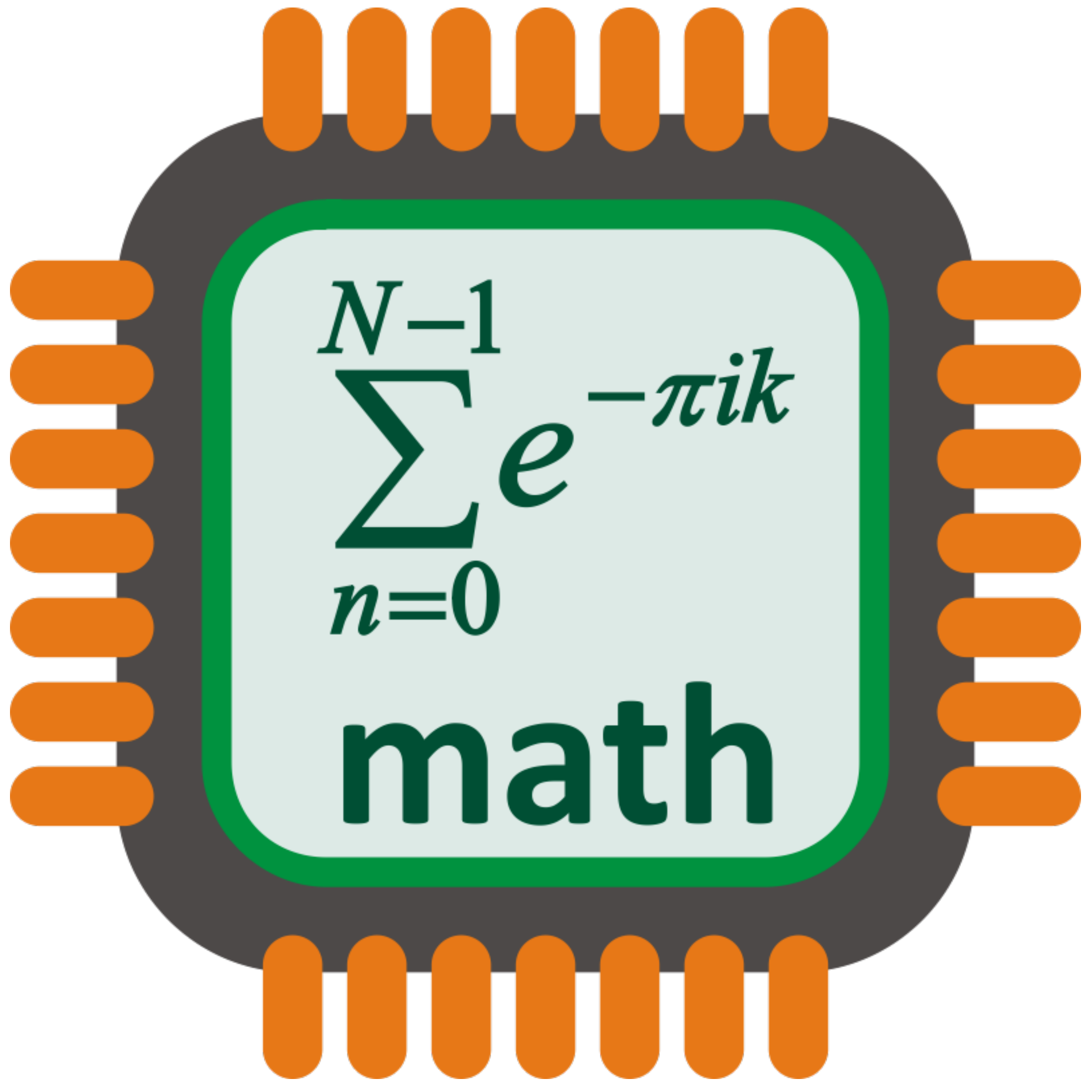
A First Attempt

g is a prime number and the exponents, a,b, are large positive integers

$30226801971775055948247051683954096612865741943 = 7^{?}$

This approach could work, but there is no upper limit on **how large** A,B,K may become.

Moreover, if we want to use K for a symmetric encryption scheme, K needs to be encoded into an n-bit string, for some **fixed** value n.



WANTED: a mathematical object that allows us to do arbitrary exponentiations while guaranteeing the values we get stay within a certain range.

(Cyclic) Group

$\mathbb{Z}_{12}$

$\text{mod } n$

$3 + 5 = ? \mod 7$

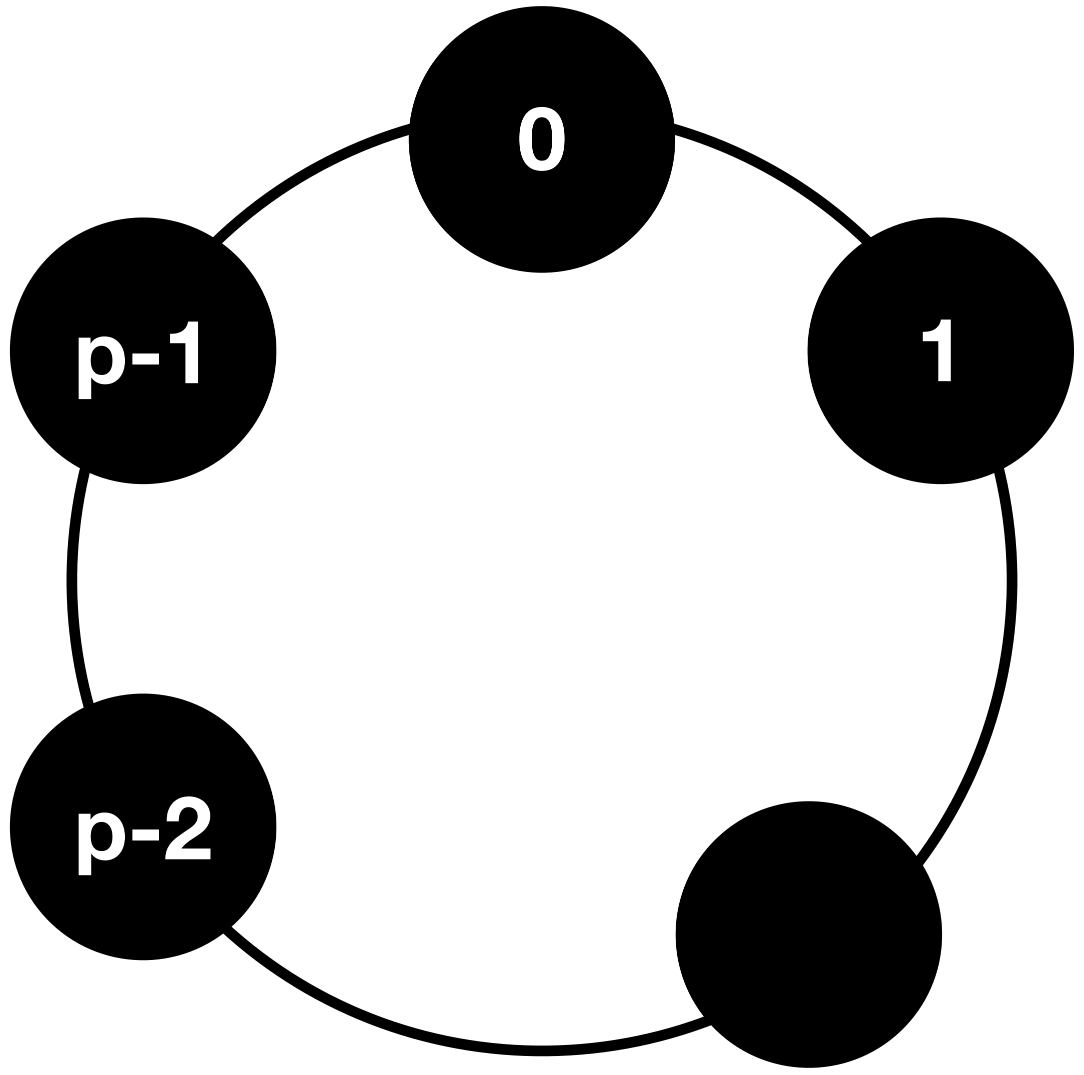
$3^5 = ? \mod 7$

Cyclic Group

Think $\mathbb{G} = (\mathbb{Z}_p, +)$

$\mathbb{Z}_p = \{0,1,\ldots,p-1\}$

$g \star h = g + h \mod p$



Definition: Cyclic Group

A group $\mathbb{G}$ is a finite set of elements (usually also denoted at $\mathbb{G}$) together with an operation $\star$, that is, a function $\star$: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$ with the following properties:

1. **Closure**: for all $g, h \in \mathbb{G}$ it holds that $g \star h \in \mathbb{G}$
2. **Associativity**: for all $g, h, k \in \mathbb{G}$ it holds that
$$(g \star h) \star k = g \star (h \star k)$$
3. **Identity**: There exists an element $e \in \mathbb{G}$ such that
$$e \star g = g \star e = g \text{ for all } g \in \mathbb{G}$$
4. **Inverse**: for every $g \in \mathbb{G}$ there exists a (unique!) element $\bar{g} \in \mathbb{G}$ such that $g \star \bar{g} = e$.

A **cyclic group**, is a group for which there exists at least one element $g \in \mathbb{G}$ that **generates** the whole group: $\langle g \rangle = \{g, g \star g, g \star g \star g, \ldots\} = \mathbb{G}$, g is called **generator.**

A Closer Look at $\mathbb{Z}_p$ = ($\mathbb{Z}_p$, +), With p a Large Prime Number

$\mathbb{Z}_p = \{0,1,\ldots,p-1\}$

Has cardinality ***p-1*** *(which is for sure divisible by 2 and at least one more prime number)*

$\mathbb{Z}_p^* = \{1,\ldots,p-1\}$ all elements in $\mathbb{Z}_p$ that have a multiplicative inverse

Consider the group $\mathbb{Z}_p^* = (\mathbb{Z}_p^*, \cdot\,)$ equipped with multiplication. This group has a funky structure that you will study in **Home Assignment 2**

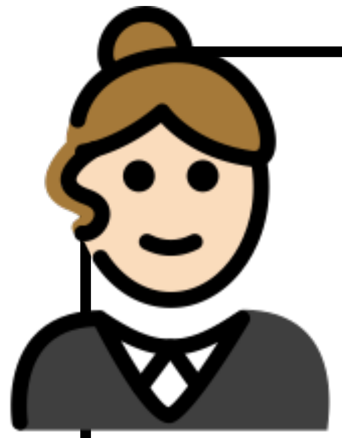
Let $p - 1 = \prod q_i^{\alpha_i}$ (decomposition in prime factors)

Then $\mathbb{Z}_p^*$ contains cyclic sub-groups of **order** $q_1, q_1^2, \ldots, q_1^{\alpha_1}, q_2, \ldots,$

the smallest positive integer n such that: $g^n = 1$ in $\mathbb{G}$.

The Diffie-Hellman Key Exchange Protocol

Setting Let p be a large prime (2048-bits long). Find a generator g of a subgroup of prime order q in $\mathbb{Z}_p^*$. Let p, q, g be all public information.



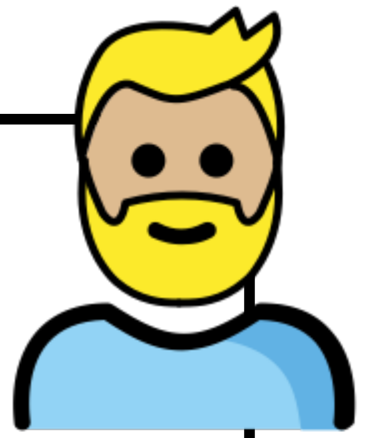
$a \leftarrow \$\mathbb{Z}_q \setminus \{0\}$

Compute $A = g^a \mod p$

Compute the shared secret $K = B^a$

$\xrightarrow{\quad A \quad}$

$\xleftarrow{\quad B \quad}$



$b \leftarrow \$\mathbb{Z}_q \setminus \{0\}$

Compute $B = g^b \mod p$

Compute the shared secret $K = A^b$

Security

For this protocol to be secure it is necessary that the values a, b are not obtainable from A, B

Module 2: Agenda

Introduction to Public Key Cryptography

- The Core Idea
- One-Way Trapdoor Functions

Key-Exchange

- Problem Statement
- A Simple Solution
- Formalisation: **Group Theory**
- Diffie-Hellman Key Exchange (DH)

(Some) Hardness Assumptions

- DLog, CDH, DDH
- Reductions Between Problems

More on DH

- On the Bit Security of DH Keys
- Securing DH Keys
- Choosing Good Parameters
- MiM Attack

Digital Signatures

- Problem Statement
- Syntax
- ECDSA

Public Key Encryption
Much More on Digital Signatures
Secure Instant Messaging
Post Quantum Cryptography

The Discrete Logarithm Assumption (DL, DLog or dLog)

Let $\mathbb{G}$ be cyclic group of order q (where q is a n-bit long prime) and g be a generator of $\mathbb{G}$.
The **discrete logarithm assumption** states that it is computationally infeasible for any efficient attacker to find the exponent x such that $g^x = h$ for a random $h \in \mathbb{G}$.

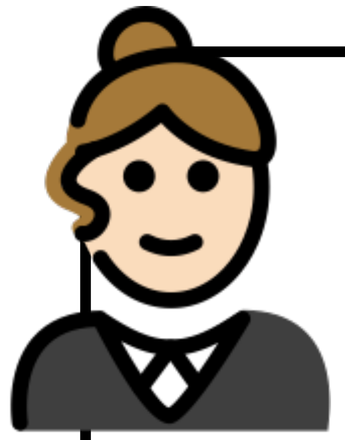
Formally: $Pr[x^* = x \,|\, x \leftarrow \$\mathbb{Z}_q, x^* \leftarrow \mathscr{A}(q, g, g^x)] < negl(n)$

This is an **assumption**: it **cannot be proven**!

Decades of cryptanalysis and scrutiny by the cryptographic community world-wide
has make us gain confidence that this assumption is true, for *large enough* primes

Is This Enough?

Formally: $Pr[x^* = x \,|\, x \leftarrow \$\mathbb{Z}_q, x^* \leftarrow \mathscr{A}(q, g, g^x)] < negl(n)$



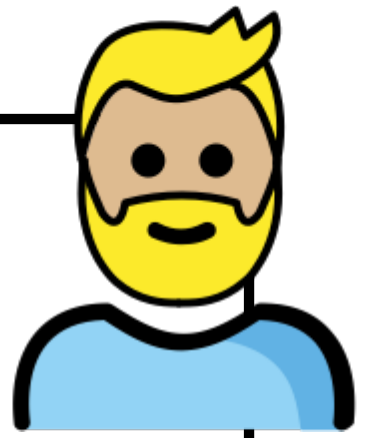
$a \leftarrow \$\mathbb{Z}_q$

Compute $A = g^a \mod p$

Compute the shared secret $K = B^a$

$\xrightarrow{\quad A \quad}$

$\xleftarrow{\quad B \quad}$



$b \leftarrow \$\mathbb{Z}_q$

Compute $B = g^b \mod p$

Compute the shared secret $K = A^b$

..it may still be possible for $\mathscr{A}$ to compute K combining A, B and without learning a, b..

The Computational Diffie-Hellman Assumption (CDH)

Let $\mathbb{G}$ be cyclic group of order q (where q is a n-bit long prime) and g be a generator of $\mathbb{G}$. The **computational Diffie-Hellman assumption** states that it is computationally infeasible for any efficient attacker to find g^{ab} given g, g^a, g^b.

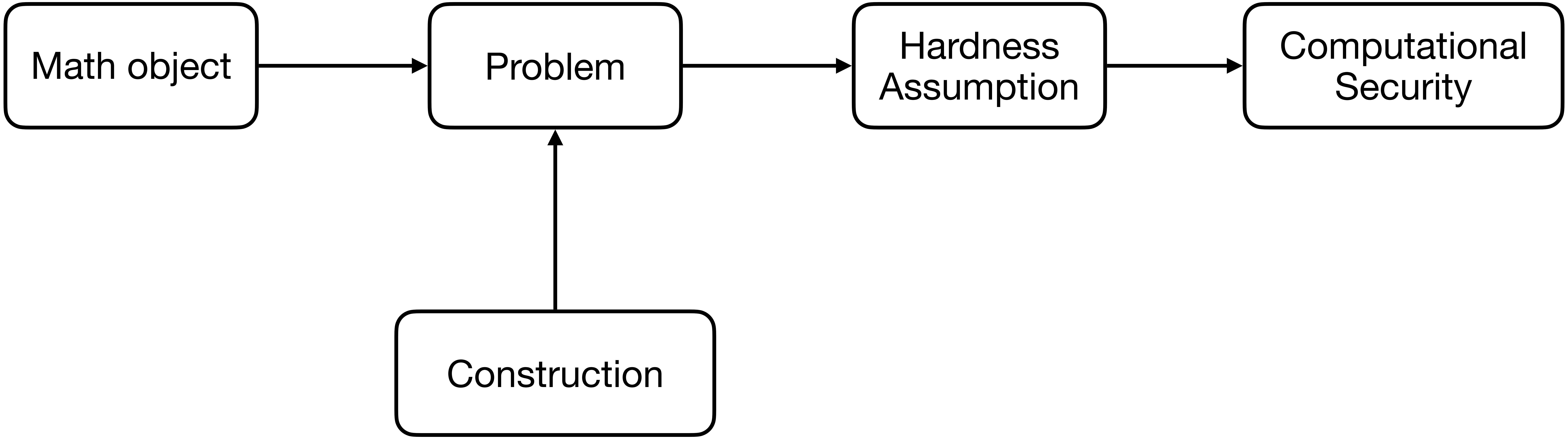
Formally: $Pr[k^* = g^{ab} \,|\, a, b \leftarrow \$\mathbb{Z}_q, k^* \leftarrow \mathscr{A}(g, g^a, g^b)] < negl(n)$

Note: if DLog is easy, then DH is easy.

But: if DH is easy, is it true that DLog is also easy? This is an open question in cryptography.


Why Do We Need
Assumptions in
Cryptography?

The Flow Chart of How Cryptographic Scheme Are Born



Which Problem Is Harder/Easier?

Let A and B be two computational problems.
A is said to **efficiently** (in polynomial time) **reduce** to B, written $A \leq B$ if:

- There is an algorithm which solves A using an algorithm which solves B.
- This algorithm runs in polynomial time if the algorithm for B does.

Proof structure: build a **reduction** (sequence of steps, program)

- Assume we have an oracle (or efficient algorithm) to solve problem B.
- We then use this oracle to give an efficient algorithm for problem A.

Three Problems … And Their Relations

Discrete Logarithm Problem (DLP):
Given $h \in \mathbb{G}$ find x such that $h = g^x$.

$\geq$

Computational DH Problem (CDH):
Given $a = g^x$ and $b = g^y$ find $c = g^{xy}$

$\geq$

Decisional DH Problem (CDH):
Given $a = g^x$, $b = g^y$ and $c = g^z$, determine if $g^{xy} = g^z$.

Given $(g, a, b) \in \mathbb{G}^3$
Use the DLog oracle to compute $y = dLog_g(b)$
Compute the CDH solution: $(a)^y = g^{xy}$
⇒ **CDH is no harder than DLP, i.e. CDH ≤ DLP**

Given $(g, a, b, c) \in \mathbb{G}^4$
Use the CDH oracle to compute $g^{xy} = DH(a, b)$
Check whether $c = g^{xy}$
⇒ **DDH is no harder than CDH, i.e. DDH ≤ CDH**

For more info, check out this blog

Module 2: Agenda

Introduction to Public Key Cryptography
- The Core Idea
- One-Way Trapdoor Functions

Key-Exchange
- Problem Statement
- A Simple Solution
- Formalisation: **Group Theory**
- Diffie-Hellman Key Exchange (DH)

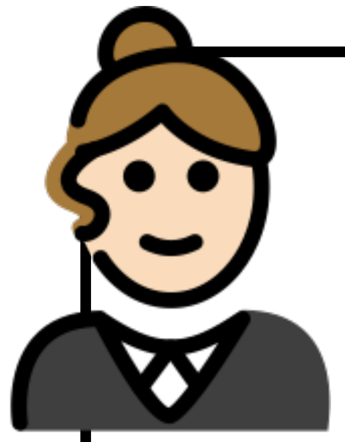
(Some) Hardness Assumptions
- DLog, CDH, DDH
- Reductions Between Problems

More on DH
- On the Bit Security of DH Keys
- Securing DH Keys
- Choosing Good Parameters
- MiM Attack

Digital Signatures
- Problem Statement
- Syntax
- ECDSA

Public Key Encryption
Much More on Digital Signatures
Secure Instant Messaging
Post Quantum Cryptography

On the Bit Security of Plain DH Keys (and DLog Problem)



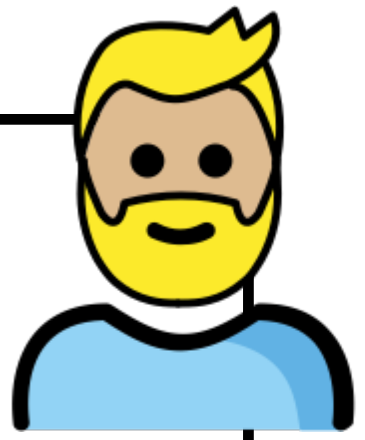
$a \leftarrow \$\mathbb{Z}_q$

Compute $A = g^a \mod p$

Compute the shared secret $K = B^a$

$\xrightarrow{\quad A \quad}$

$\xleftarrow{\quad B \quad}$



$b \leftarrow \$\mathbb{Z}_q$

Compute $B = g^b \mod p$

Compute the shared secret $K = A^b$

Bad news: it is (computationally) easy to find the least significant bit (LSB) of K

$dLog_g(K)$ is even iff K is *quadratic residue* in $\mathbb{Z}_p^*$

Worse news: It is easy to compute the s LSBs or MSBs of K when $p - 1 = 2^s \cdot q$, with q odd.

"Hardness of Distinguishing the MSB or LSB of Secret Keys in Diffie-Hellman Schemes" [FPSZ06]

Securing DH Keys

Good news: Finding some bits (aka **hard-core** bits) is as hard as computing the whole dLog

e.g. computing the $s+1$-th LSB or MSB of K, when $p-1=2^s \cdot q$, with q odd is **hard**

Even better news:

Heuristics show that $H(K)$ provides a good cryptographic key if $H:\{0,1\}^{|p|} \rightarrow \{0,1\}^{256}$ is a cryptographic hash function

+ there are several ways to boost security for DH and dLog

Triple Diffie-Hellman (X3DH)

$\mathbb{G}$ is made of points on an elliptic curve

Choosing Parameters

Setting Let p be a large prime (2048-bits long). Find a generator g of a subgroup of prime order q in $\mathbb{Z}_p^*$. Let p, q, g be all public information.

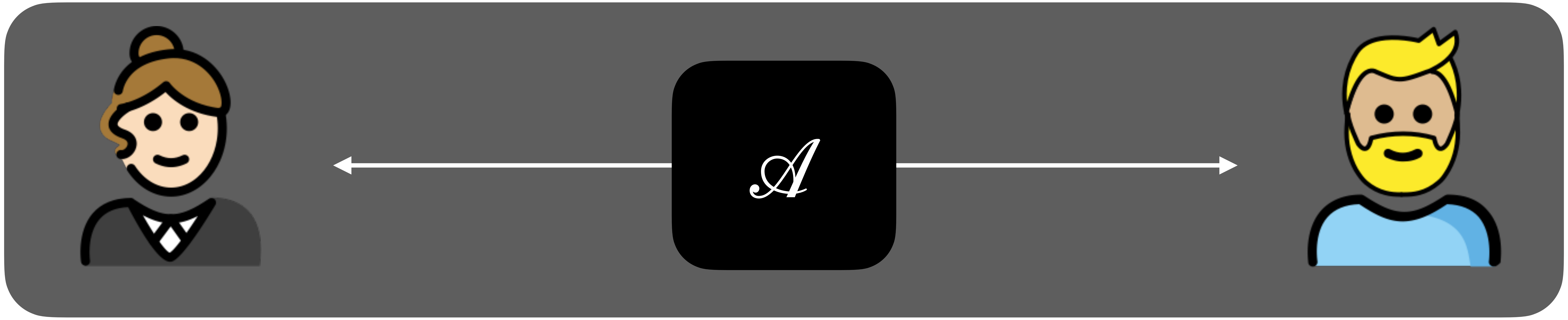
🧐 *Why are we working in $(\mathbb{Z}_q, \cdot\,)$ instead of $(\mathbb{Z}_p^*, \cdot\,)$*

Here $\cdot$ denotes the operation of the group (multiplication)

- We want to work with prime orders (p, q should both be prime). This guarantees a nice mathematical structure for computing exponentiations.
- Having $p \neq q$ lets us better balance security vs size of the exchanged messages:
  - p needs to be large enough for DLog to be hard.
  - q can be fairly small for efficient exponentiation, yet not too small as it upper bounds the length of the key material we can derive.
  - For realistic sizes today, we have $|p| = 2048$ bits and $|q| = 256$ bits.

Man-in-the-Middle Attack Against the DH Key Exchange

Alice and Bob want to find a way to share a secret key k without relying on a previously shared secret **AND** they want to do so, using a public channel, that is monitored* by the Adversary



🧐 *What goes bad if $\mathscr{A}$ is active?*

**For the sake of this lecture, we only consider passive $\mathscr{A}$ (eavesdropper)*

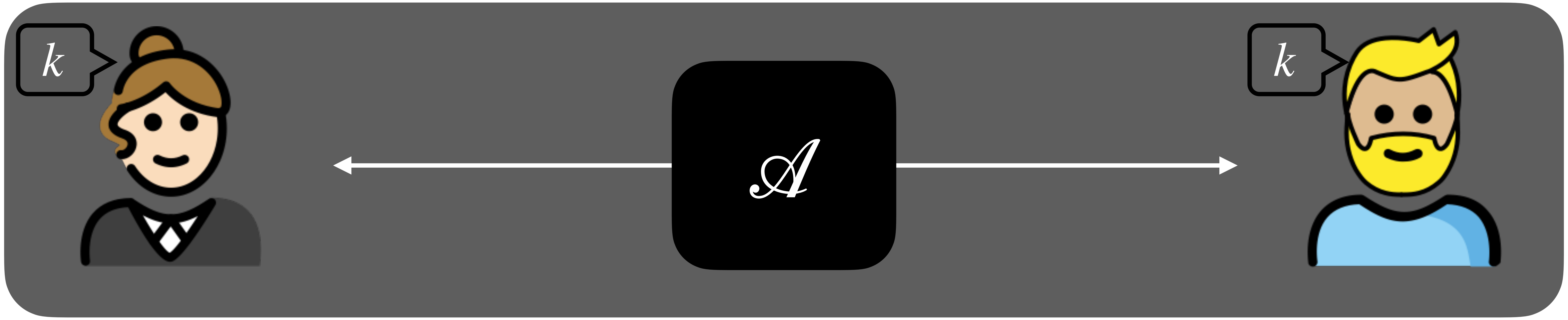
Man-in-the-Middle Attack Against the DH Key Exchange

$\mathscr{A}$

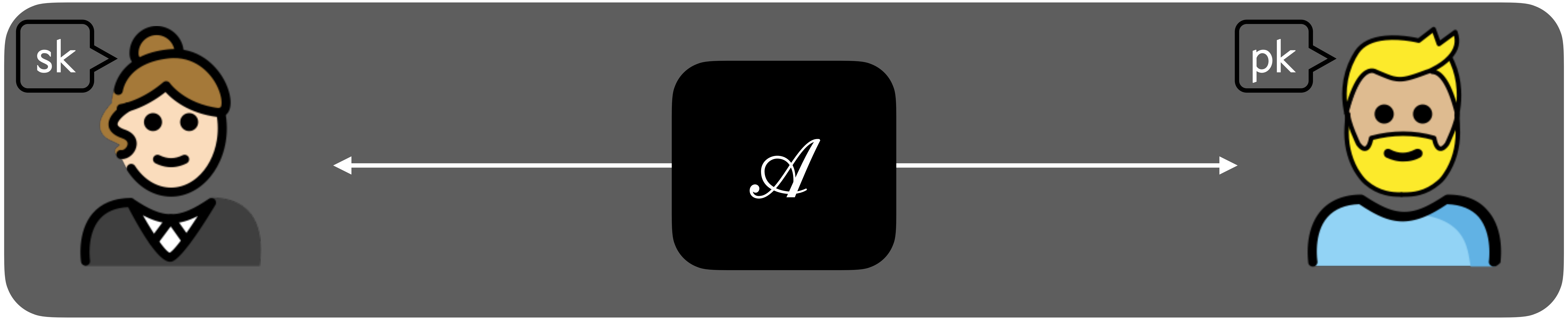
$a \leftarrow \$\mathbb{Z}_q$ $A = g^a \mod p$ $K_A = (B^*)^a$	$\longrightarrow A \longrightarrow$ $\longleftarrow B^* \longleftarrow$	$a^* \leftarrow \$\mathbb{Z}_q$ $A^* = g^{a^*} \mod p$ $K_B = B^{a^*}$ $b^* \leftarrow \$\mathbb{Z}_q$ $B^* = g^{b^*} \mod p$ $K_A = A^{b^*}$	$\longrightarrow A^* \longrightarrow$ $\longleftarrow B \longleftarrow$	$b \leftarrow \$\mathbb{Z}_q$ $B = g^b \mod p$ $K_B = (A^*)^b$

What's enabling this attack? DH does not authenticate whom you are doing a key exchange with

Authenticating the Source of Information Over the Internet



Problem: if both Alice and Bob know *k*, then cryptographically they are the same person. Bob cannot convince a third party that it was Alice producing something (e.g. a MAC) for that requires the knowledge of *k. Whatever Alices produces, Bob can produce it as well*!



With **public key cryptography** Alice is the only one to know *sk.* If she uses it to do something that is (computationally) impossible to do without *sk*, then everyone can be convinced she did it.

Digital Signature - Syntax

Definition: Digital Signature

A digital signature scheme is a triple of PPT algorithms $(KeyGen, Sign, Ver)$ defined as follows:

- $KeyGen(n) \rightarrow (\mathsf{pk}, \mathsf{sk})$ is a probabilistic key generation algorithm
- $Sign(\mathsf{sk}, m) \rightarrow \sigma$ is a (possibly) probabilistic algorithm that outputs a signature σ for a message m
- $Ver(\mathsf{pk}, m, \sigma) = 1$ if σ is accepted as a valid signature for m against pk, 0 (reject) otherwise.

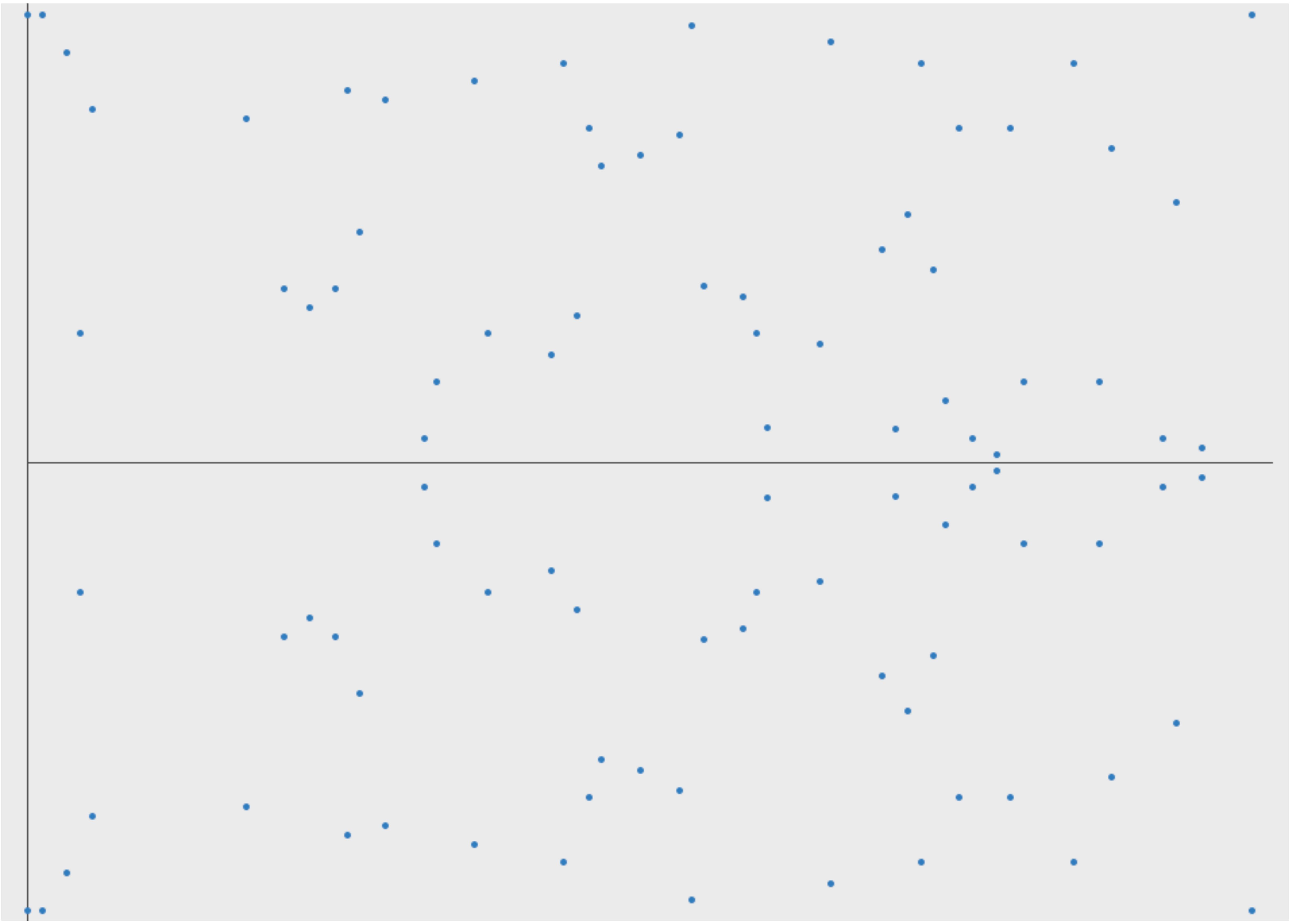
Correctness

For all key pairs $(\mathsf{pk}, \mathsf{sk}) \leftarrow KeyGen(n)$ it holds that: $Ver(\mathsf{pk}, m, Sign(\mathsf{sk}, m)) = 1$

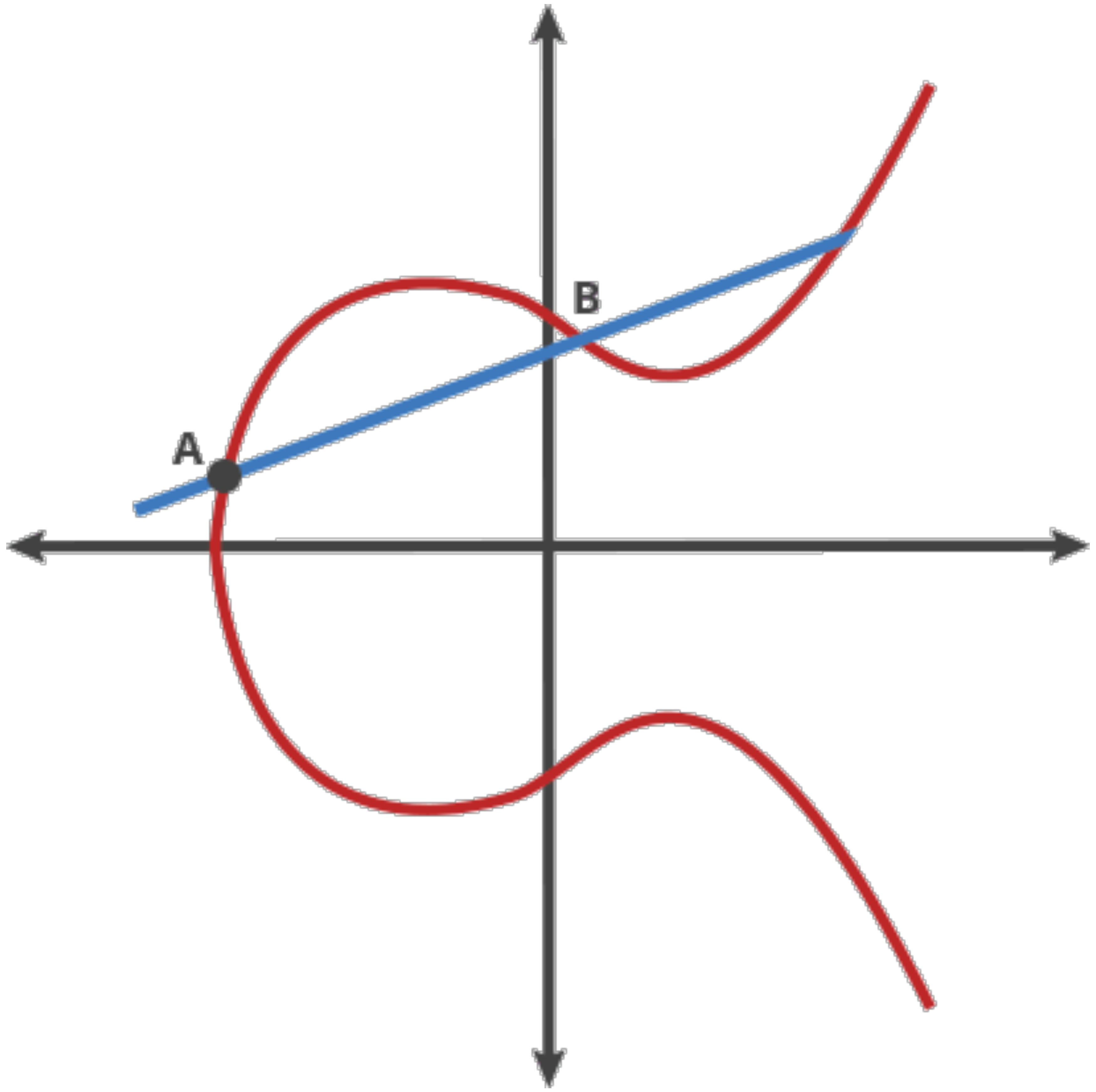
$$Pr[Ver(\mathsf{pk}, m, \sigma) = 1 \,|\, \sigma \leftarrow Sign(\mathsf{sk}, m)] = 1$$

ECDSA - Background on Elliptic Curve Cryptography

$y^2 = x^3 - x + 1 \mod 97$



$y^2 = x^3 + ax + b$



*Elliptic curves have a **group** structure*

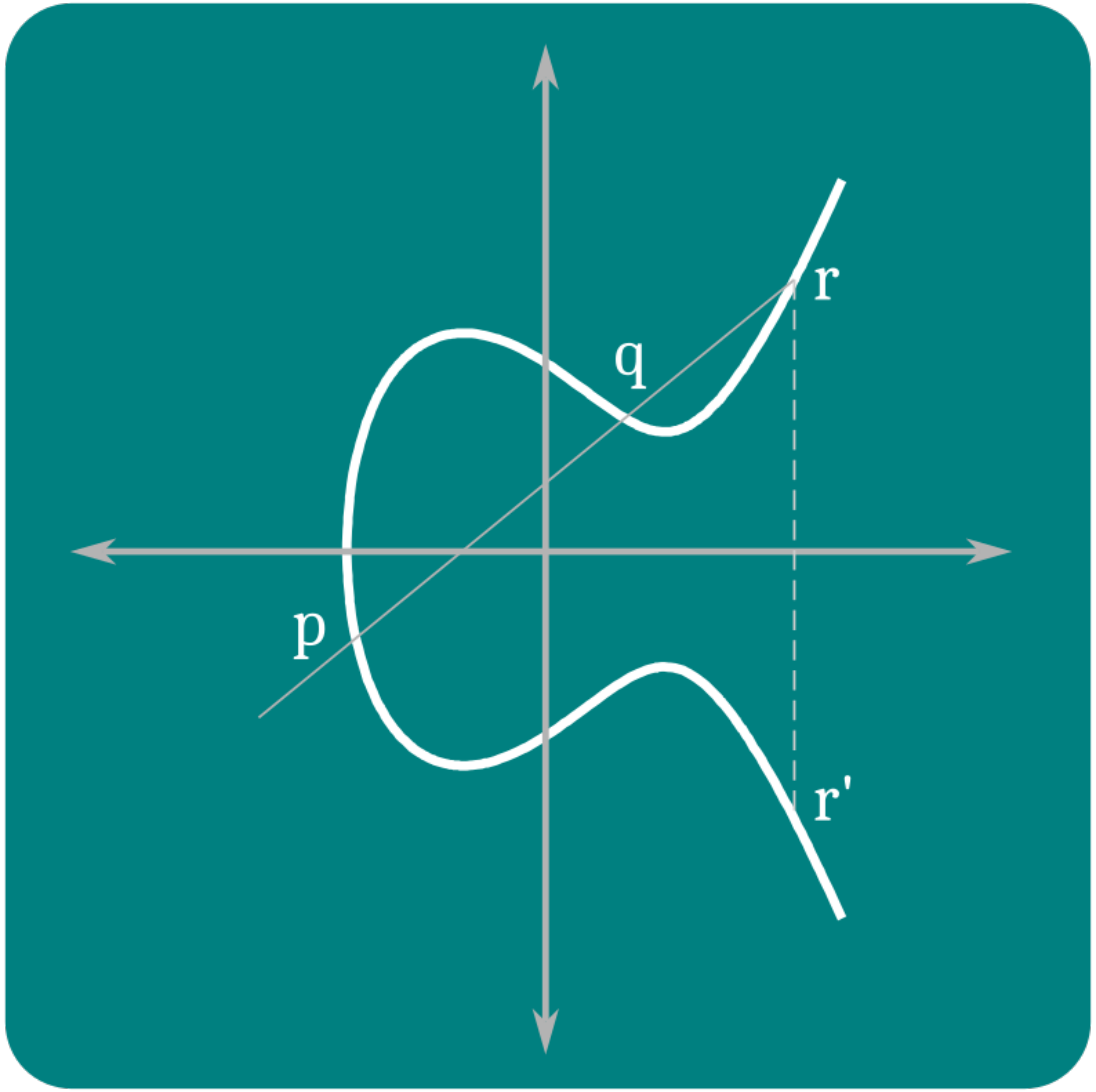
[gifs from arstechnica]

ECDSA - Algorithms

```
KeyGen (sec.par) ⇨ (sk, pk)

  d ←$── [0 ... n-1]
  sk = d
  pk = Q = d*G
```

```
Sign (sk, msg) ⇨ sgn

  k ←$── [0 ... n-1]
  R = k*G
  r = R_x mod n
  z = sha256(msg)
  s = inv(k)·(z + d·r) mod n
  sgn = (r, s)
```

```
Verify(pk, msg, sgn) ⇨ {0, 1}

  z = sha256(msg)
  T = [z·inv(s) mod n]*G
  P = [inv(s)·r mod n]*Q
  if R == T+P return 1
  else return 0
```



ECDSA - the Good

- ★ Shorter keys and better security than the RSA signature scheme
- ★ Non malleable
- ★ IoT friendly
- ★ In wide adoption (TLS, DigiCert (Symantec), Sectigo (Comodo) …)

ECDSA - the Bad

PS3 hacked through poor cryptography implementation

repeated nonce attack Bonus 2

A group of hackers named fail0verflow revealed in a presentation how they ...

CASEY JOHNSTON - 12/30/2010, 6:25 PM

{* SECURITY *}

Android bug batters Bitcoin wallets

Old flaw, new problem

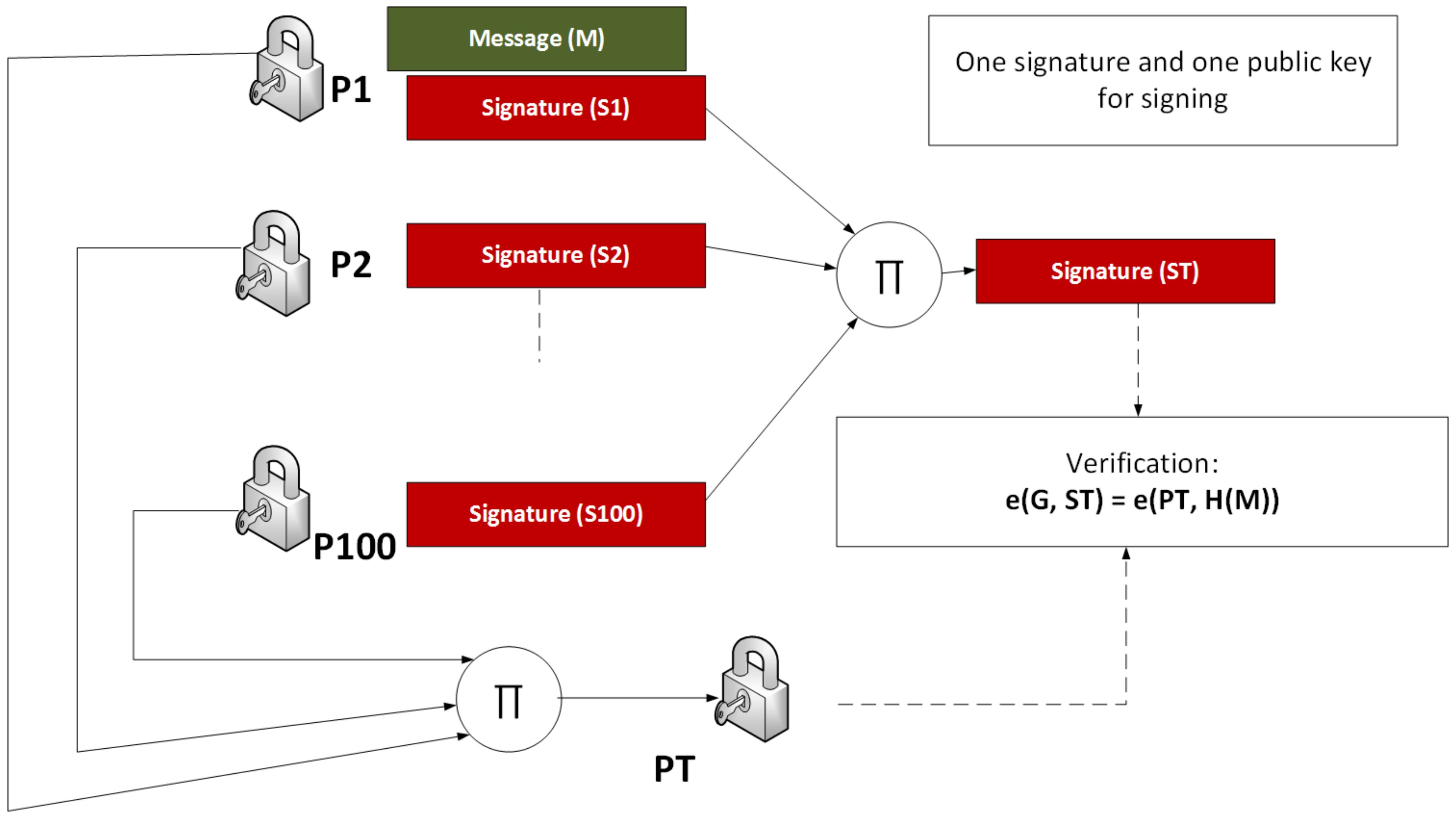
Richard Chirgwin Mon 12 Aug 2013 // 00:43 UTC

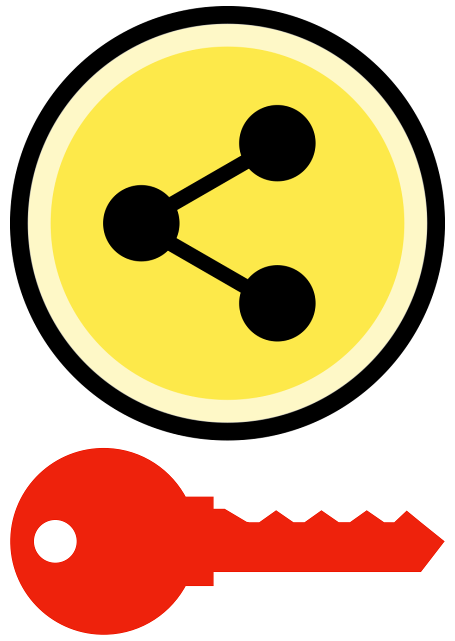
LadderLeak: Side-channel security flaws exploited to break ECDSA cryptography

Charlie Osborne 28 May 2020 at 14:07 UTC
Updated: 28 June 2021 at 09:05 UTC

ECDSA - What's Next?

Threshold Signatures

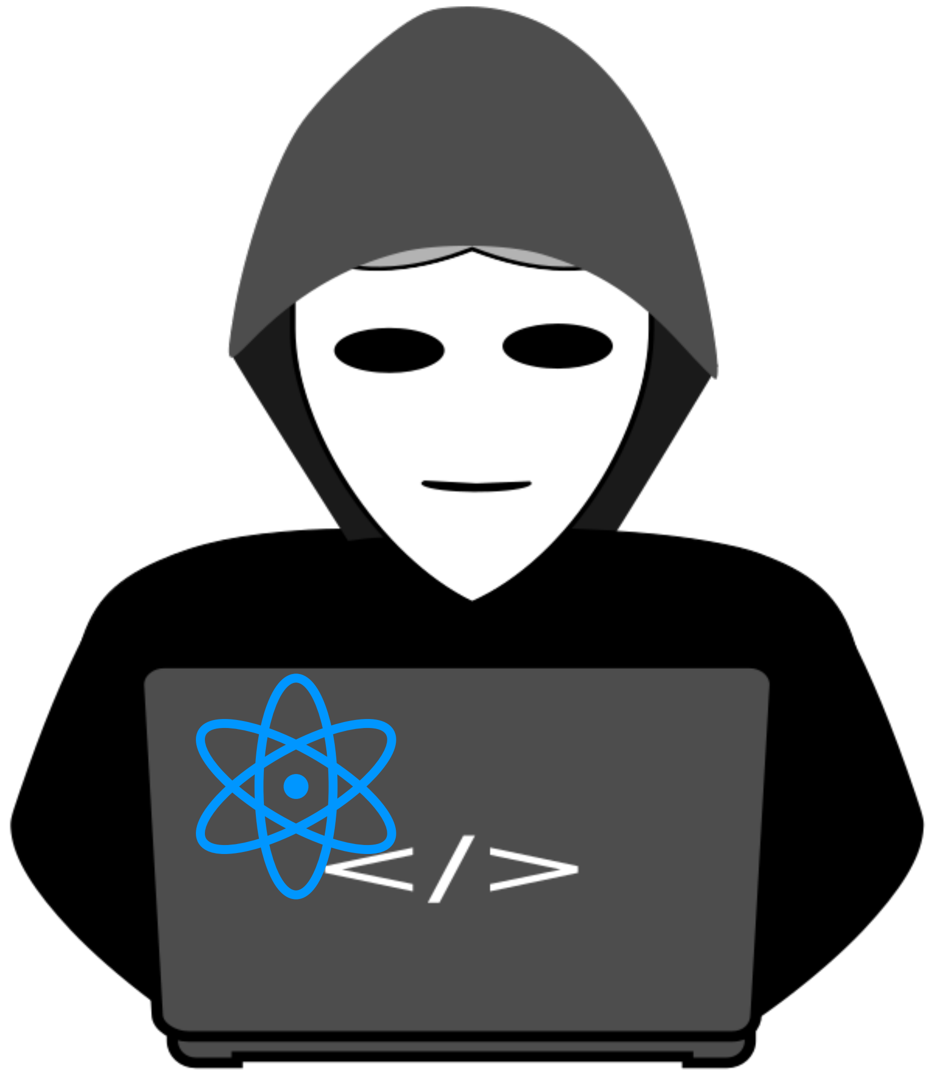




LBS signature
Schnorr signature

Post Quantum Secure Signatures

FALCON
Fast-Fourier Lattice-based
Compact Signatures over NTRU



DILITHIUM