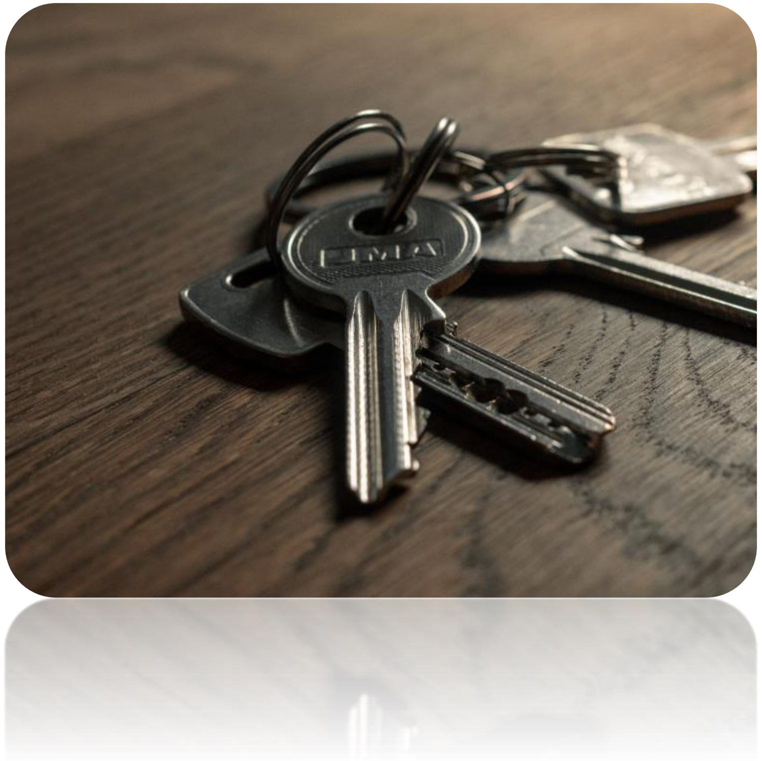
Network Security

Lecture 2: Cryptography

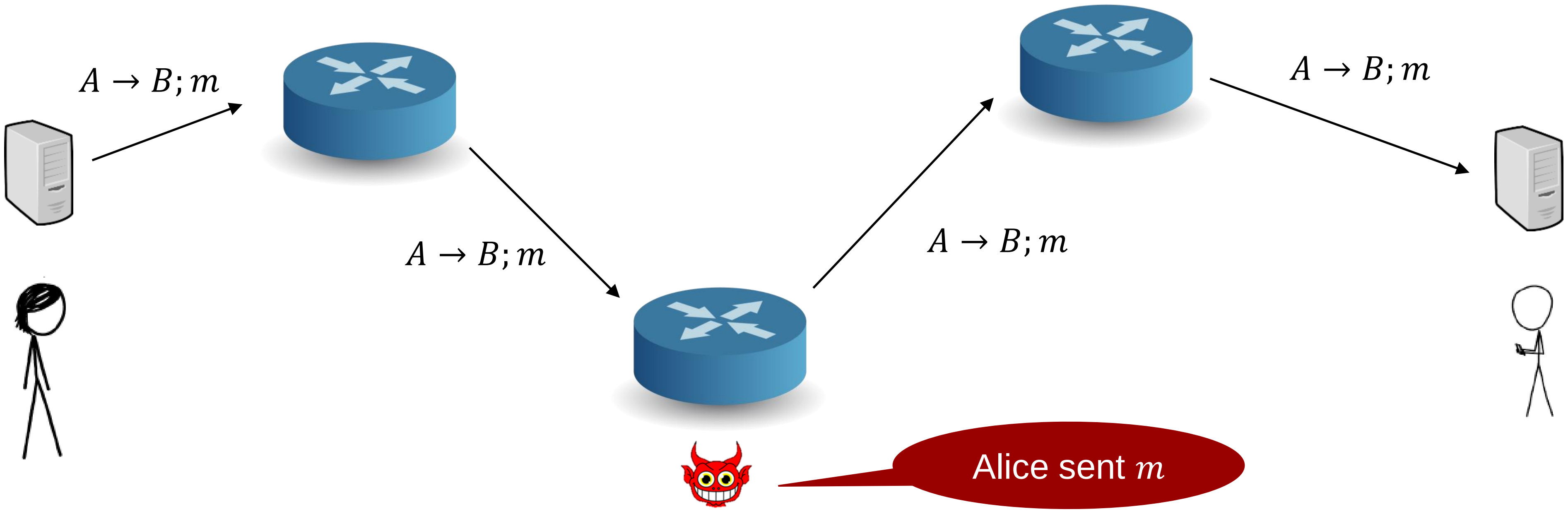
Schedule for today

1. Cryptography – what is it good for?
2. Talking about security of cryptography
3. Block ciphers
4. Hash functions & authenticated encryption

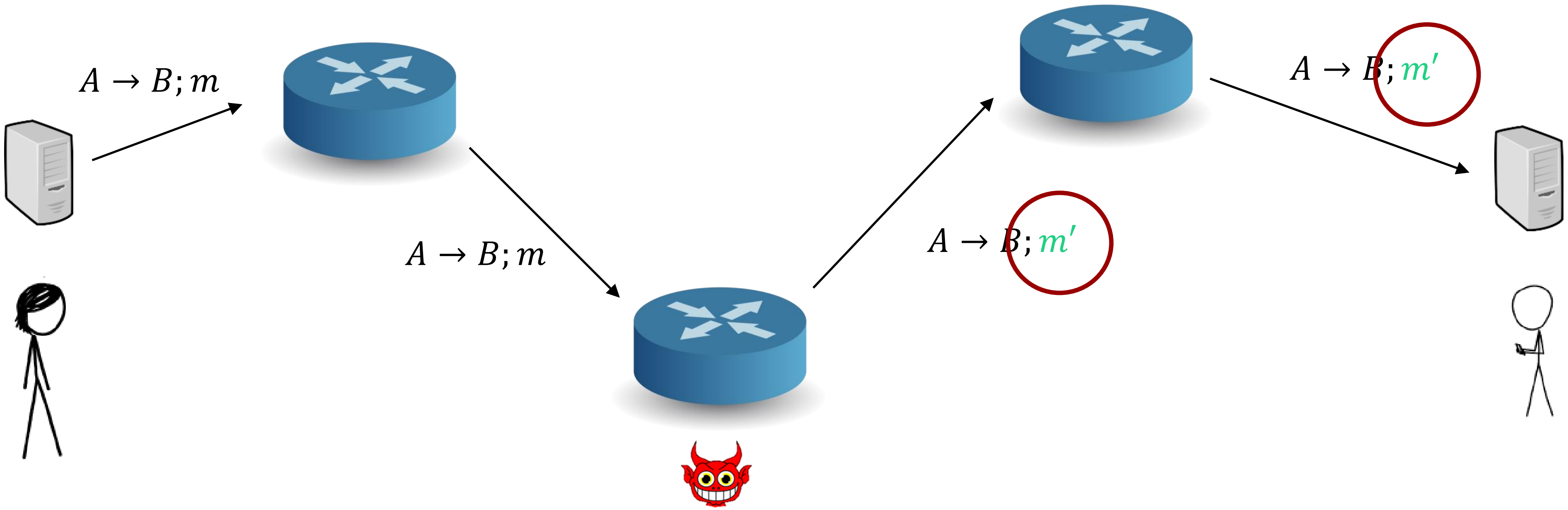
5. Agreement on keys
6. Sending messages without key agreement
7. Achieving non-repudiation through digital signatures



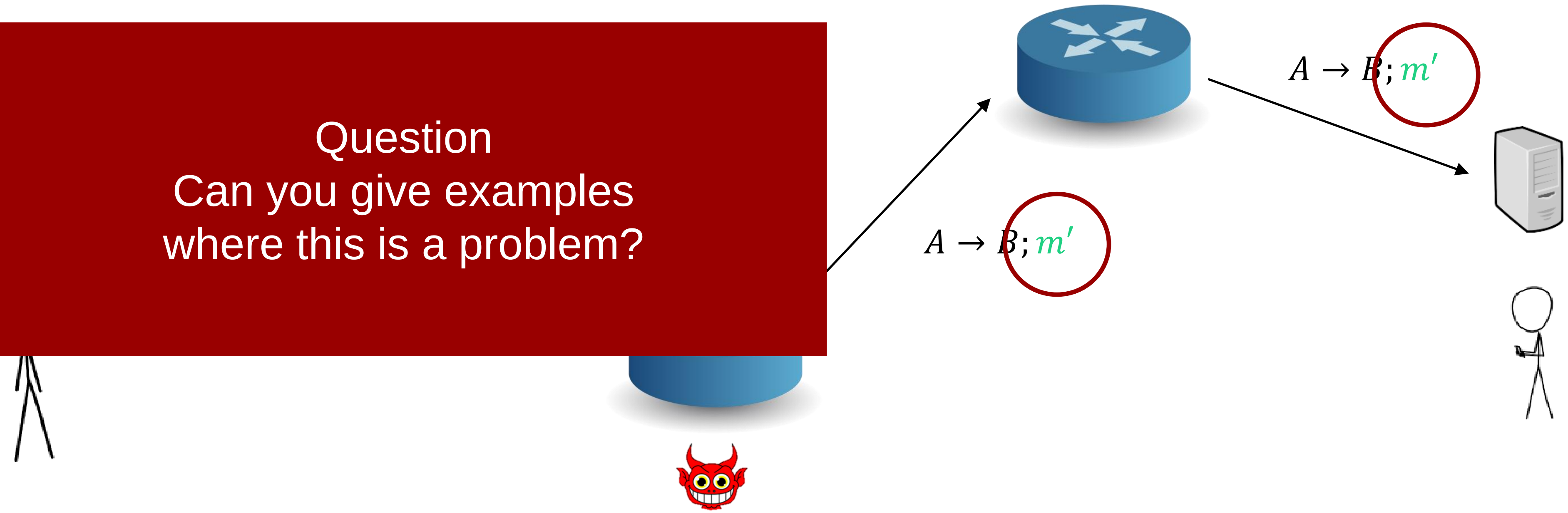
Network communication



Network communication

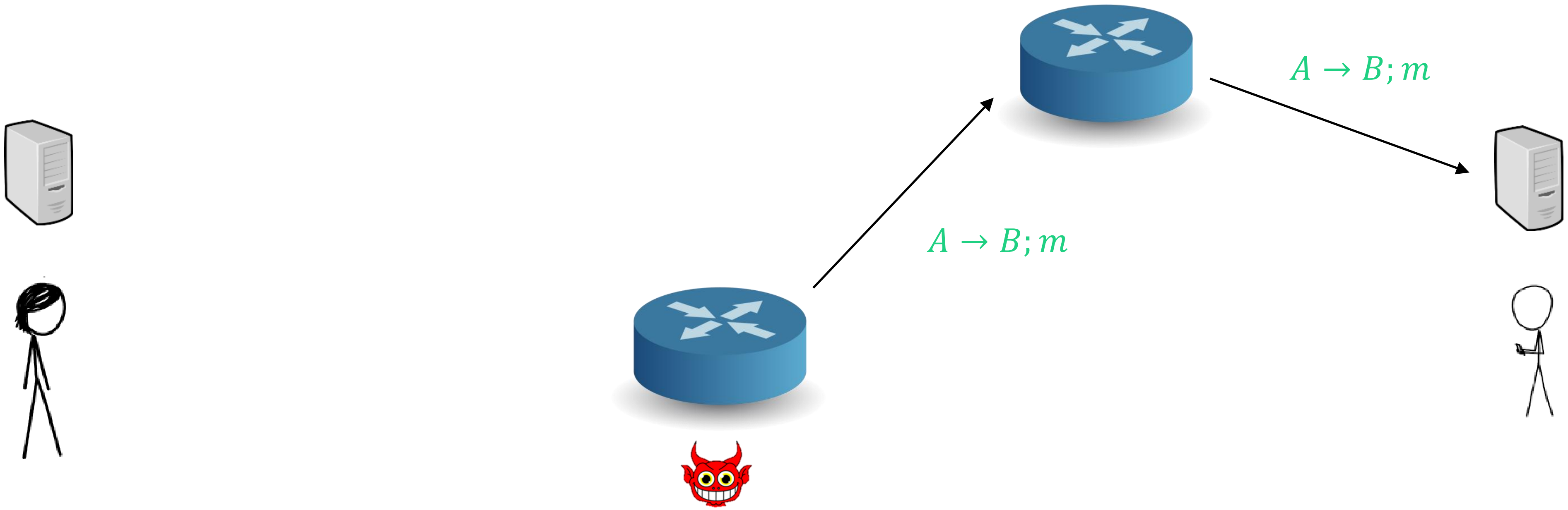


Network communication

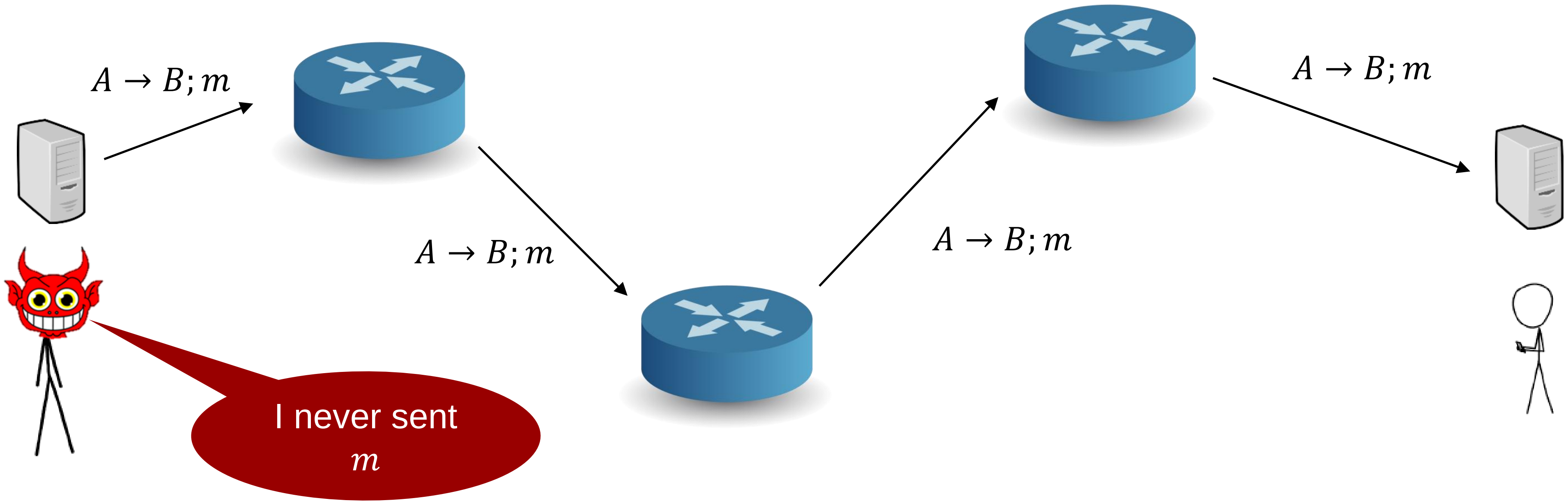


Question
Can you give examples
where this is a problem?

Network communication



Network communication



Goals of Cryptography in communication

– **Confidentiality**
  • Message can be read only by the intended receiver

– **Integrity**
  • Receiver can verify that the message has not been altered

– **Authentication**
  • Receiver can verify the identity of the sender

– **Non-repudiation**
  • Sender cannot deny that it sent the message

Question
Do you see other, related goals worth protecting?

Definitions

- Sender (Alice) and receiver (Bob) wish to **communicate securely** over an insecure medium
  - Messages can be eavesdropped, copied, altered, injected, etc by attacker

- They use **cryptographic algorithms** to protect their communication

Depending on goal, these algorithms have different names

- Cryptographic algorithms generally use a **secret** to achieve protection, called a **key**

Protection against whom?

1. Specify what the **goal** of the adversary is:
   Break confidentiality or integrity?
2. Identify **capabilities and knowledge** of the attacker:
   Knows part of message? Knows parts of key? Knows algorithm?
3. Choose protection level against the attacker.

Modern cryptographic algorithms

- Kerckhoff's Principle: your attacker knows the cryptographic algorithm!
- Your attacker should **not** have the secret key
- The attacker may have access to previous communication and all information about it
- The attacker still cannot break the security property, *except using impossible amounts of computation*

Degree of protection

Modern cryptographic algorithms

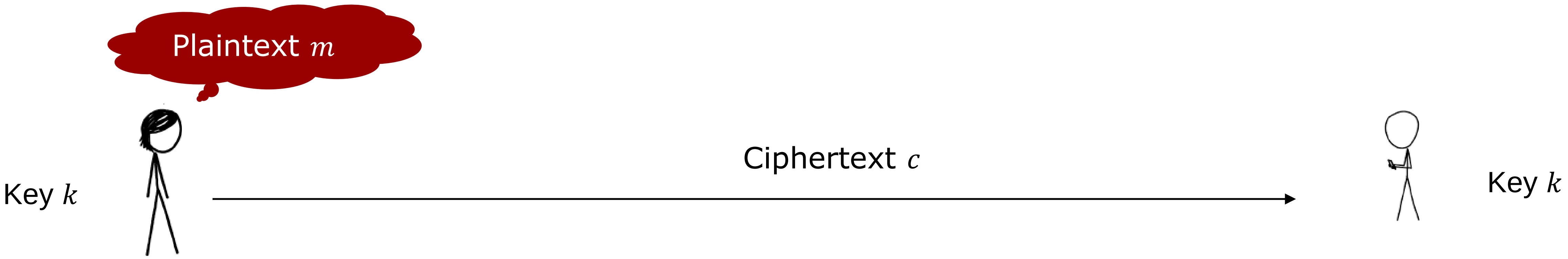
- Kerckhoff's Principle: your attacker knows the cryptographic algorithm!
- Your attacker should **not** have the secret key
- The attacker may have access to previous communication and all information about it
- The attacker still cannot break the security property, *except using impossible amounts of computation*

Level of impossibility: key-length (bit security)

Roughly:
128 bit keys (128 bit security) $\approx$ attacker needs computation proportional to 2^{128} steps

Helpful tool in practice: https://www.keylength.com/

Achieving confidentiality: Symmetric Encryption



<u>Encryption</u>

Transform m using k into c

<u>Decryption</u>

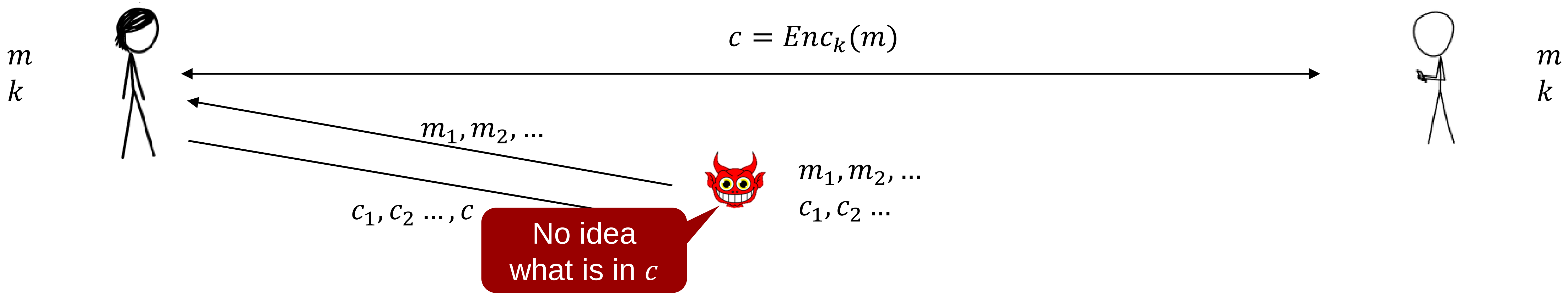
Transform c using k into m

Requirement: semantic security

Even if the **attacker can ask for encryptions** of $m_1, m_2, m_3, \ldots$ under k
it cannot learn anything about **plaintext of fresh** c
except with computation proportional to $2^{bit\ security}$

Models

1. Security against brute-force attacks on key
2. Known plaintext/ciphertext pairs (HTTP headers etc)

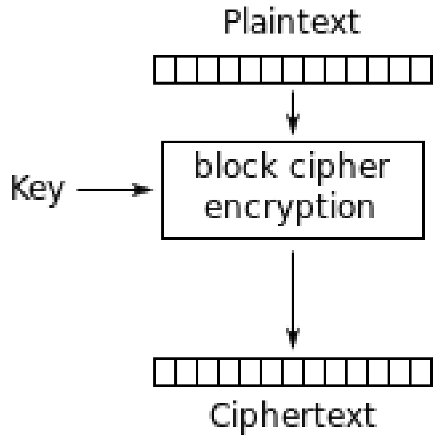


Modern symmetric ciphers: Block Ciphers

- Plaintext and ciphertext have a **fixed length** (e.g. 128 bits)
- Keys have a fixed length (128 bits)

De-facto standard: the AES cipher

- Applies operations to plaintext over multiple rounds
- Each round depends on parts of key
- Hardware-support (AES-NI)
- Also available for 192 and 256 bit keys



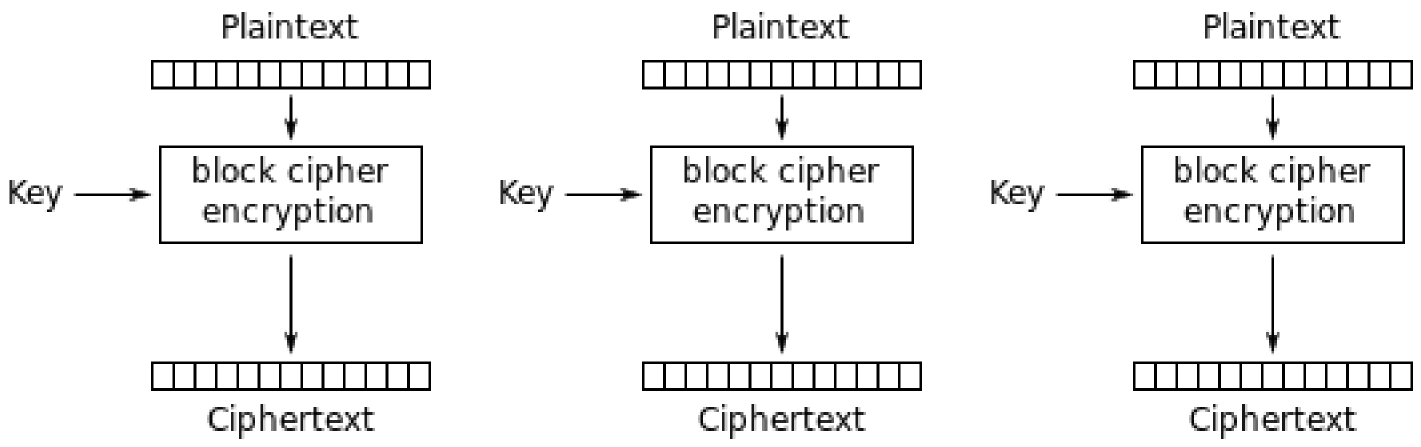
Obsolete

DES, 3DES encryption

Usually you don’t just encrypt 128 bits

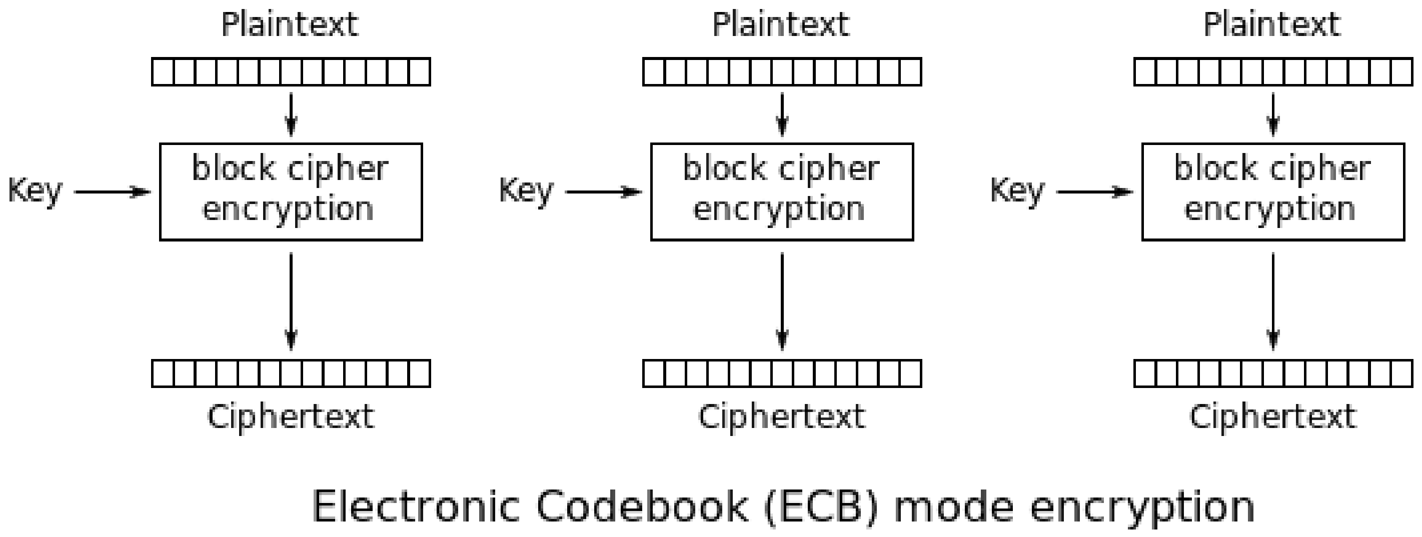
Block Ciphers Modes of Operation

- A message is partitioned into a **series of blocks**
  - Padding is used if needed
- Each block is encrypted separately
  - Ciphertext blocks are put together in an encrypted message



Electronic Codebook (ECB) mode encryption

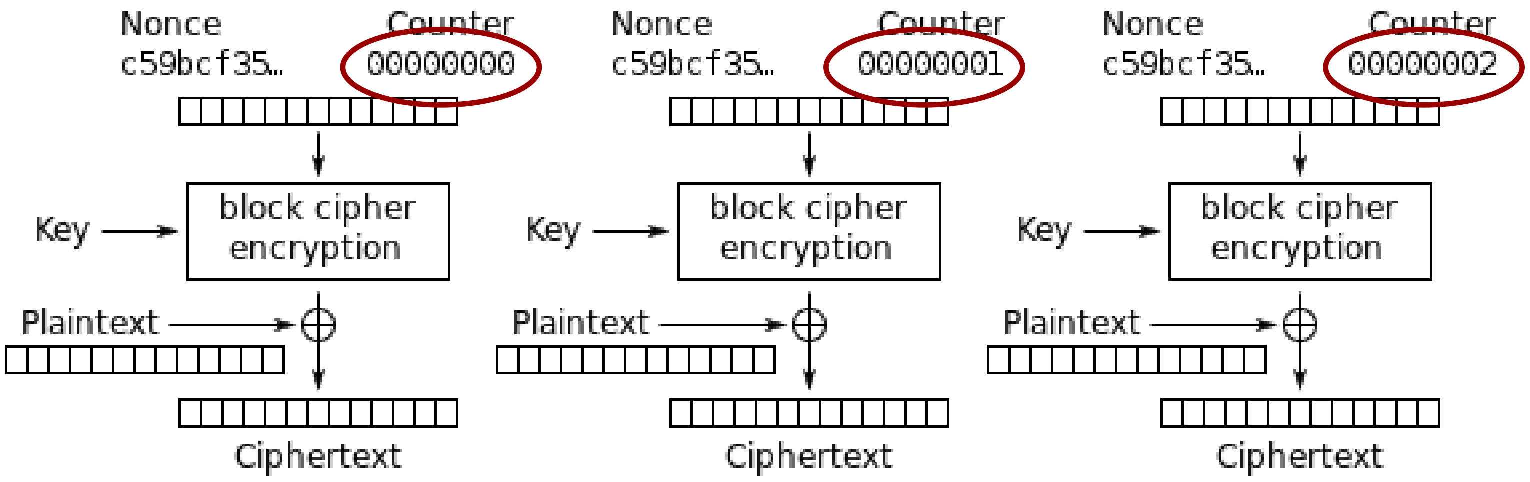
Why not to use ECB



Question
What can be the problem?

Counter Mode

- CTR combines the input with a counter to make ciphertext unique
  - Choose $Nonce$ before encryption
  - Encrypt $Nonce||ctr$, then XOR output with plaintext
  - CTR mode can be parallelized

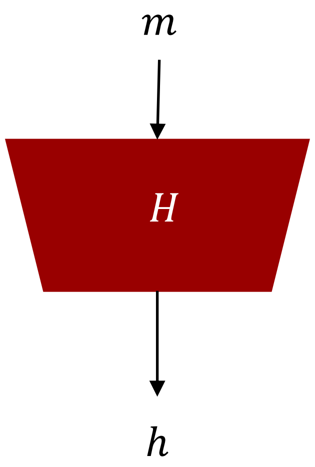


Counter (CTR) mode encryption

Towards integrity: Hash Functions

Hash function: a function H that is

1. Efficient to compute
2. $|m| \gg |h|$



<u>Naming convention</u>

- m message
- h hash value or digest

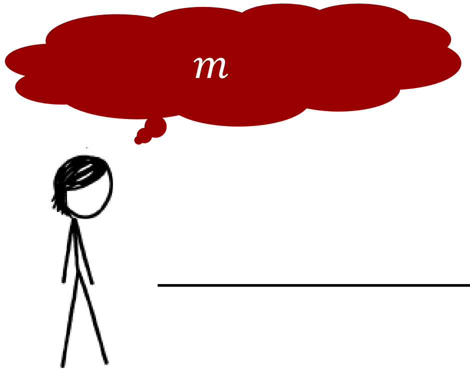
Preimage resistance: given h it's hard to find m

Collision resistance: it's hard to find m_1, m_2 such that $H(m_1) = H(m_2)$

Hash functions & authenticity

Integrity

Receiver can verify that the message has not been altered



Key k

$c, h = H(c)$

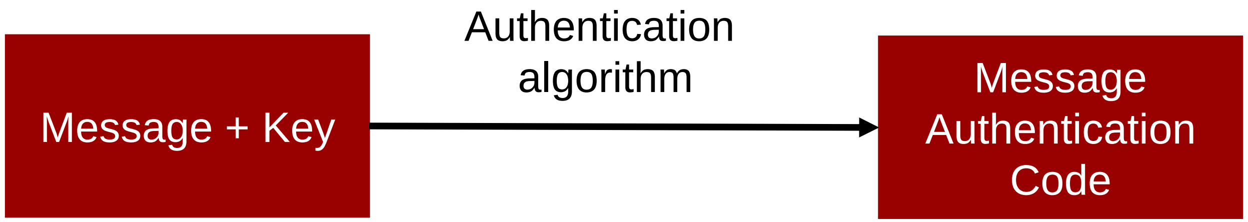
Key k

First verify hash, then decrypt

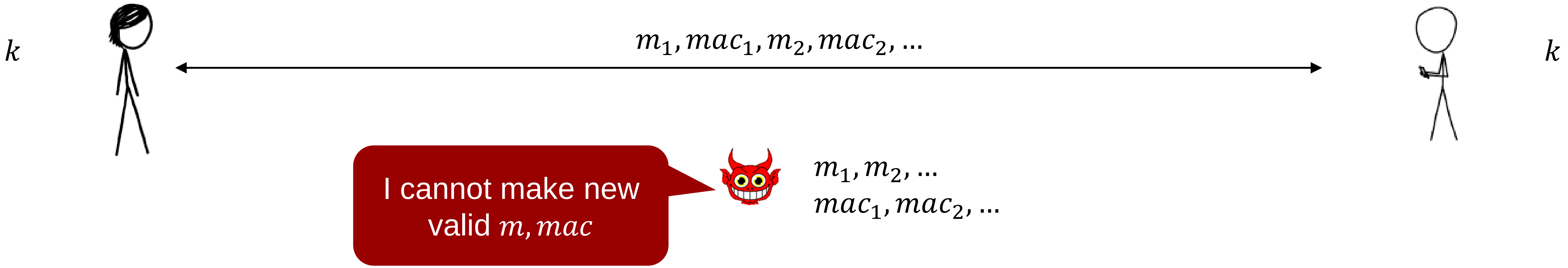
Question
Why does this not give integrity?

Message Authentication (MACs)

- Authenticate messages using **Message Authentication Codes (MACs)**
- Requires a pre-shared key similar to symmetric encryption

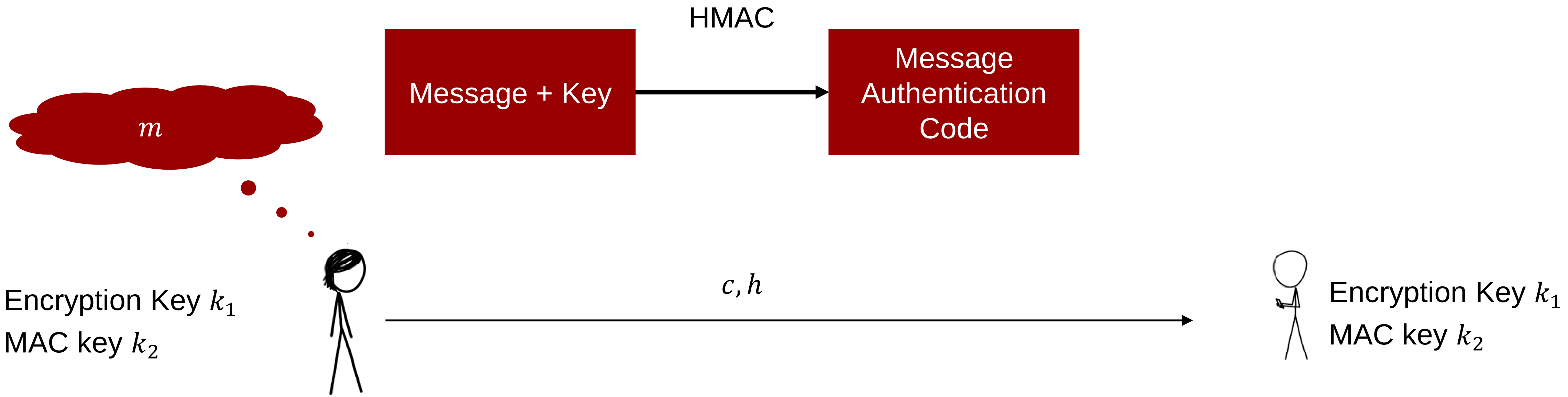


Security



Hash-Based Message Authentication (HMAC)

- A way to create **Message Authentication Codes (MACs)** using Hash functions
  - Requires a pre-shared key similar to symmetric encryption



$c = Enc_{k_1}(m)$
$h = HMAC_{k_2}(c)$

1. Check $HMAC$ h using k_2
2. Decrypt using k_1

Disadvantages of Symmetric Cryptography

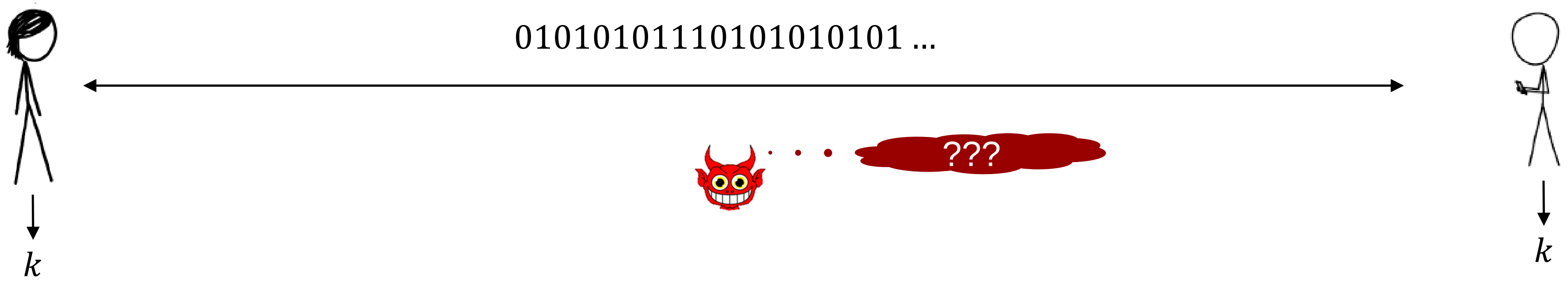
- The **chicken-and-egg** problem
  - You need a shared key k to establish a secure channel
  - You need a secure channel to share the key??

- **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

- Cannot offer **non-repudiation** in e.g. HMAC
  - The key is shared among (at least) two parties, sender can deny that is the author

BREAK

The key-agreement problem



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

<u>MUST assume</u>

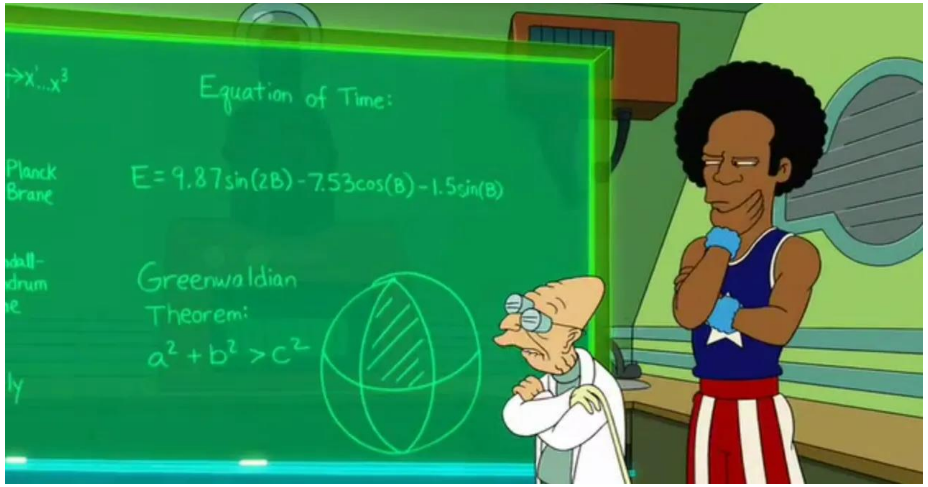
Attacker cannot alter/drop messages

Question: What happens if attacker can alter messages?

The key-agreement problem

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

Solution: Math!



1976: Diffie and Hellman have an idea…

12

New Directions in Cryptography

Whitfield Diffie (Member, IEEE),
Martin E. Hellman (Member, IEEE)

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

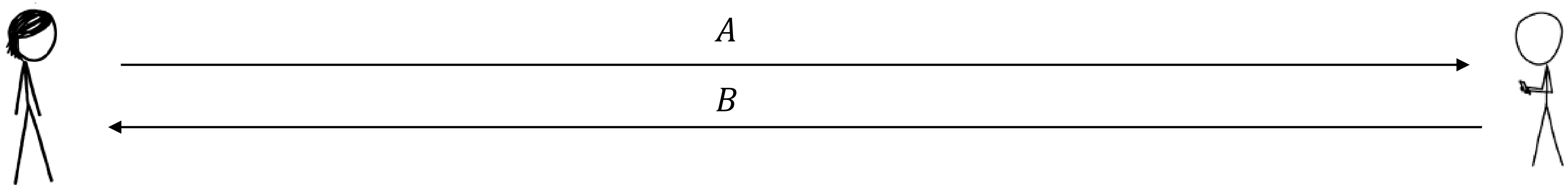
Turing Award in 2015

Diffie Hellman key agreement

Fix a large primes $q, p = 2q + 1$ (thousands of bits long)

Fix $g \in \{2, \ldots, p-1\}$ such that $g^{q-1} = 1 \; mod \; p$ but $g^{(q-1)/2} \neq 1 \; mod \; p$

Public information!



1. Choose random $a \in \{0, \ldots, q-1\}$
2. Compute $A = g^a \; mod \; p$
3. Output $k = B^a \; mod \; p$

1. Choose random $b \in \{0, \ldots, q-1\}$
2. Compute $B = g^b \; mod \; p$
3. Output $k = A^b \; mod \; p$

Diffie Hellman key agreement

What we need for security

If p, g, g^a are known then it should be hard to compute a

Why it works

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

Example

$p = 23, g = 5$

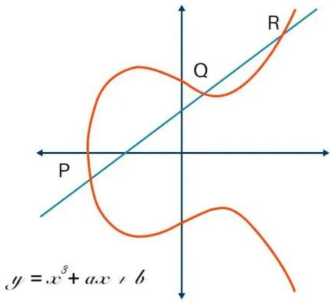
Alices chooses $a = 4$, Bob chooses $b = 7$

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

$17^4 = 4^7 = 8\, mod\ 23$

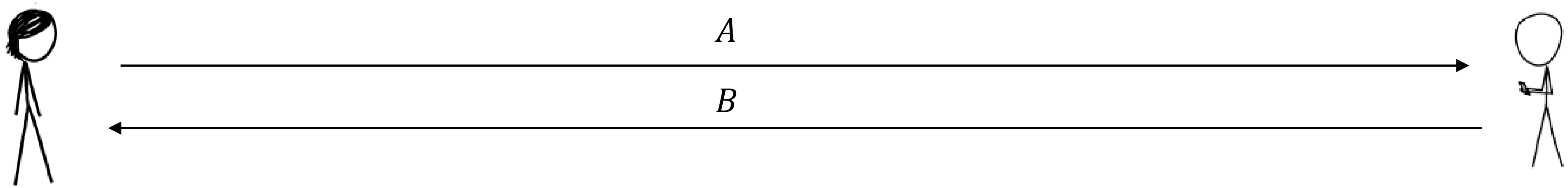
Real parameters:
keylength.com

Diffie Hellman in practice



If you want security until 2030, recommendation of $\log_2 p$ up to 15000 bits

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



They deliver 128 bit security, but A, B only ≈ 260 bits long

Question: what other improvements could we get?

<u>Caveat:</u>

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

Post-Quantum Key Agreement

Cryptography used today must also be secure
in >20 years

Attacker can save network traffic and analyze later

No general, efficient quantum computer exists yet,
but might be in 20+ years

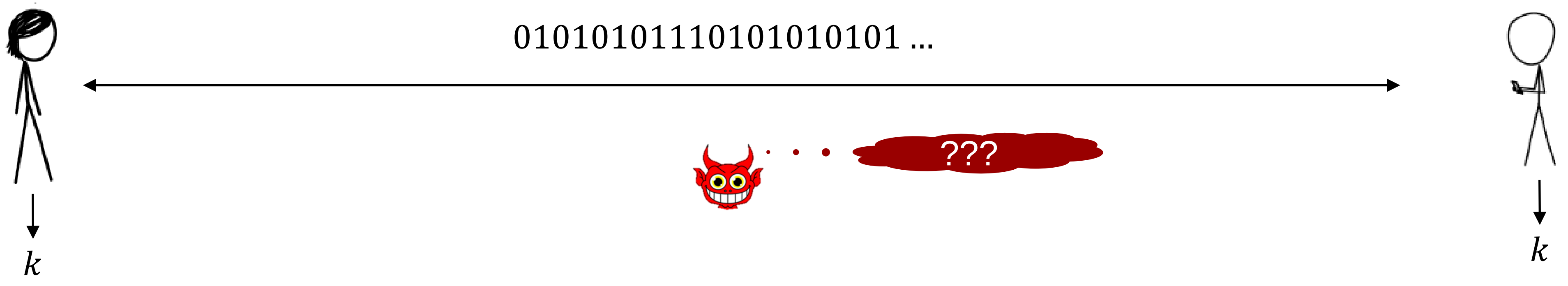
Recently (2022)

NIST standardized CRYSTALS-Kyber key agreement

Other good alternatives

NTRU, McEliece, Saber

Key Agreement is not enough



Need to exchange messages before we can encrypt

What if both are not online
at the same time?

Public key (asymmetric) encryption

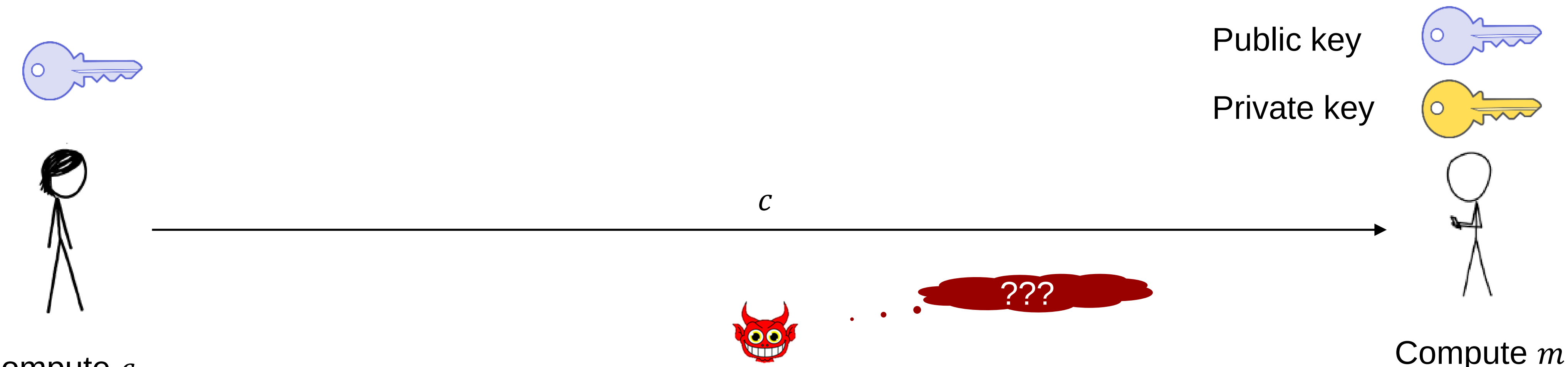
Involves two separate but mathematically related keys **per user**

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

<u>Confidentiality</u>

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

Public key encryption



Compute c using pk

Compute m using sk

Public-key security:

No adversary with pk should be able to have any information about message in c

The RSA cryptosystem

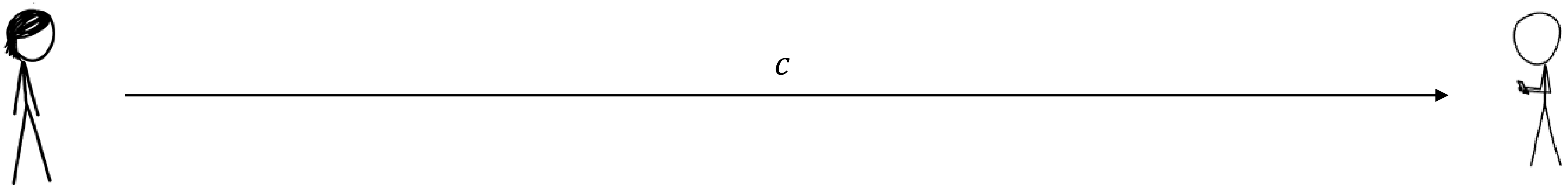
Turing Award in 2002

Invented 1977 by Rivest, Shamir & Adleman

<u>Key Generation</u>

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

Public key N, e
Private key d



Compute $c = m^e\ mod\ N$

Compute $m = c^d\ mod\ N$

The RSA cryptosystem

<u>Security of RSA</u>

Given N it is hard to find p, q

And: Given N, e it's hard to find d

Implemented in many software packages

Currently secure parameters

$\log_2 N \approx 4096$

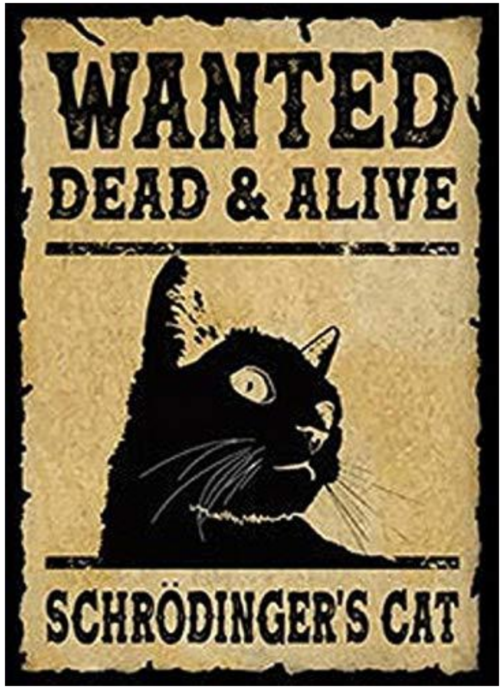
In practice one often uses $e = 2^{16} + 1$.
It works for $e = 3$ but is insecure (Coppersmith's attack)

Quantum computers strike again!

Like Diffie-Hellman, quantum computers can break RSA

CRYSTALS-Kyber, NTRU, McEliece, Saber are all secure alternatives

Follow the news about developments in the area ☺



What we have until now

Confidentiality (AES, RSA, Key Agreement)

Message can be read only by the intended receiver

Integrity (Hash functions)

Receiver can verify that the message has not been altered

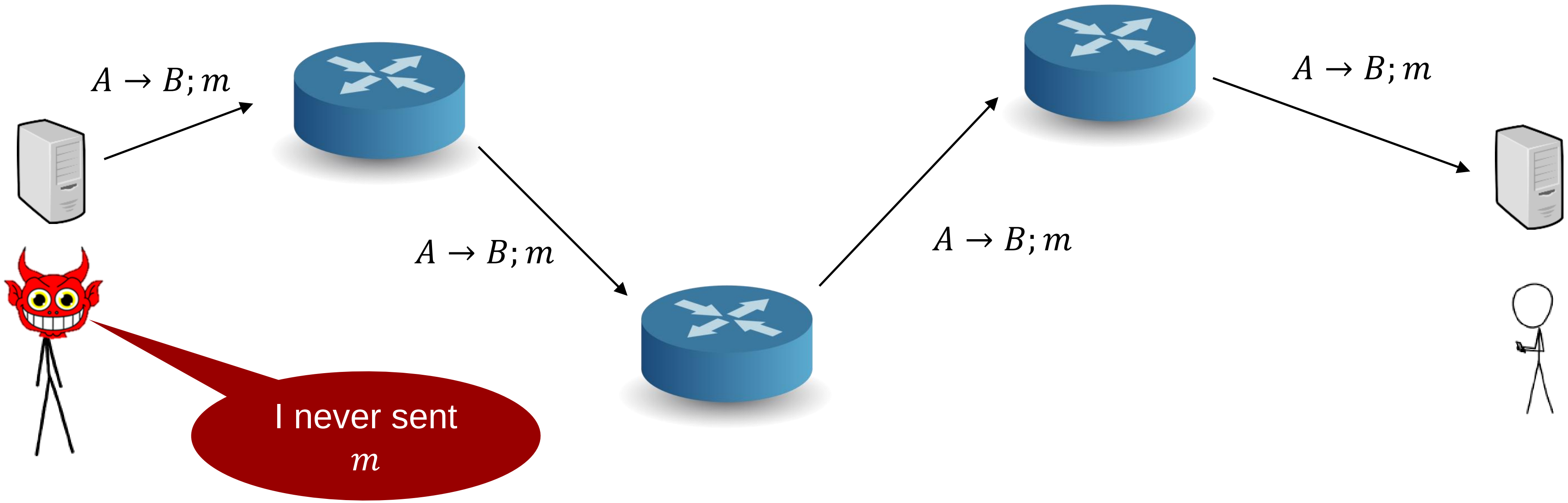
Authentication (HMAC, GCM)

Receiver can verify the identity of the sender

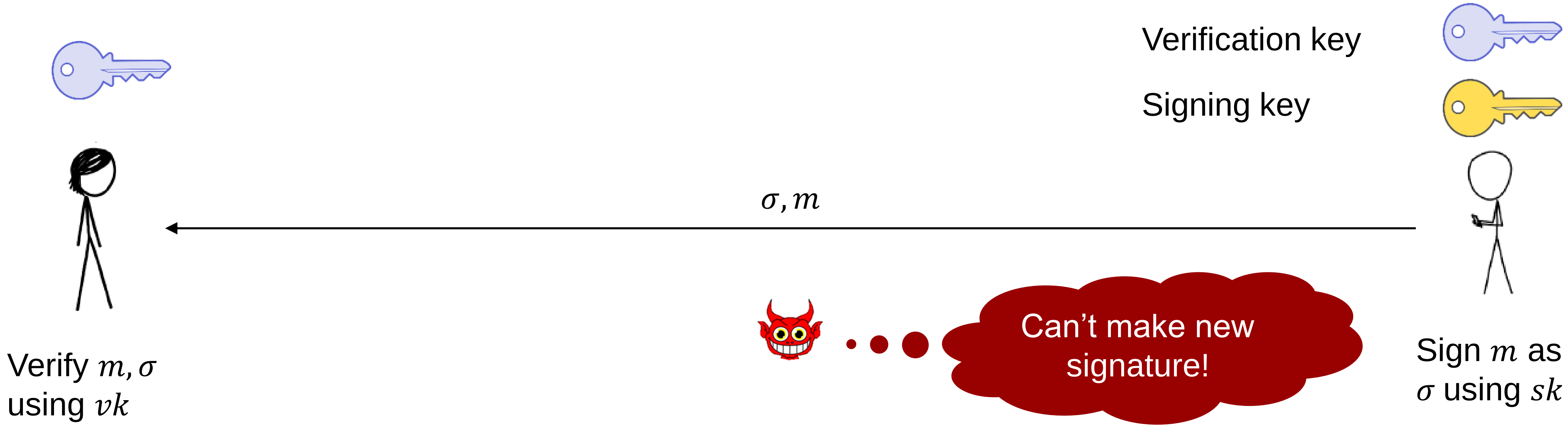
Non-repudiation ???

Sender cannot deny that it sent the message

Non-repudiation



Solution: Digital Signatures



Unforgeability:

No adversary with vk and message/signature pairs $m_1, \sigma_1, \ldots$ should be able to make new m, σ

Digital Signatures using RSA

Signing key: secret d

Verification key: N, public e

Cryptographic hash: H

Message m $\xrightarrow{\text{Hash } H}$ $h = H(m)$ $\xrightarrow{\text{Encrypt with private key}}$ $\sigma = h^d \mod N$

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

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

Alternatives to RSA signatures

Similar construction to Diffie-Hellman: DSA

More efficient:

EC-DSA, Schnorr Signatures

Post-Quantum alternatives

Dilithium, Falcon, SPHINCS+ (,FAEST ☺)

Summary of today

1. Goals of cryptography: Confidentiality, Integrity, Authenticity, Non-Repudiation
2. Security of cryptography: keys and key-lengths
3. Confidentiality: Symmetric-key encryption
4. Integrity & Authenticity: Hash functions

5. Key agreement: create secret key over a public channel
6. Public key cryptography: send messages to people you have never met
7. Digital signatures: how to be sure about the sender