Course Announcements

- Read Piazza note 227 for all updated course policies
- Homework 6 now due on Wednesday 3/25, as is Homework 7
- Friday discussion sections will be posted online, don't connect to Zoom
- Reminders about how we will use Zoom for this class
  - Keep your video camera off and microphone muted during lecture
  - The best way to ask a question is to type it in the chat window. I will pause at regular intervals to solicit and answer questions.

Lecture 14: Authenticated Key Exchange

1. File-level encryption
2. Protecting data in transit
3. Forward + backward secrecy and deniability
4. Key exchange
5. Public key cryptography
6. Public key digital signatures

Google.com in Firefox:

Technical Details

Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)

BU login page in Firefox (2017):

Technical Details

Connection Encrypted (TLS_RSA_WITH_AES_256_CBC_SHA, 256 bit keys, TLS 1.2)

1. File-level Encryption

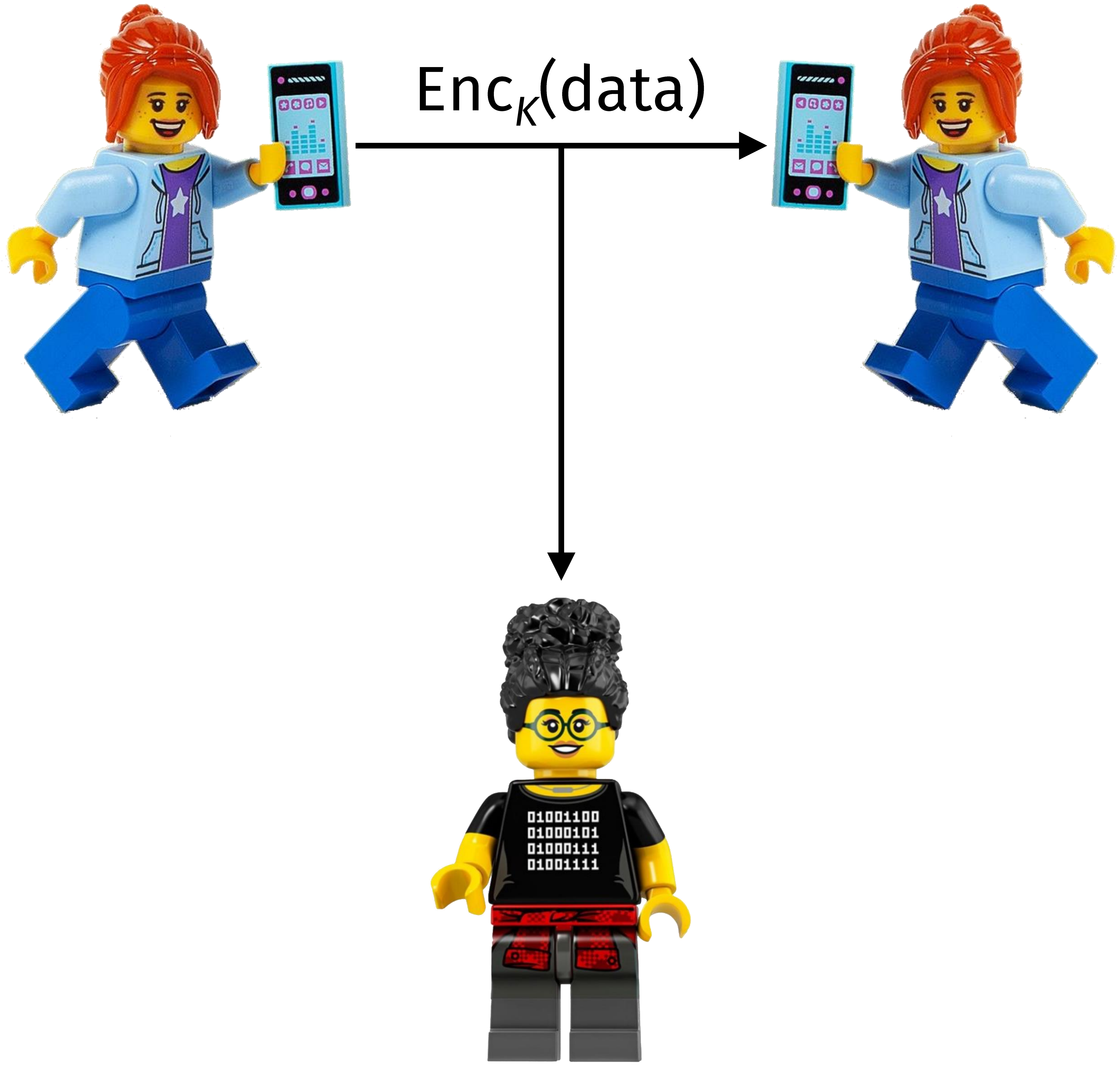
Recall: Data at rest scenario

Alice

- Knows a secret key K
- Encrypts each sector or file of her hard disk using K
- Inputs K on every device boot

Mallory

- Steals device while powered off
- Cannot exfiltrate or tamper data



Recall: Key wrapping

- Key wrapping = protect one key under another
  - Ordinary encryption does not suffice to protect keys
  - Need Deterministic Authenticated Encryption (use SIV mode)

$\text{Wrap}_{K1}(\text{ek})$

$\text{Wrap}_{K2}(\text{ek})$

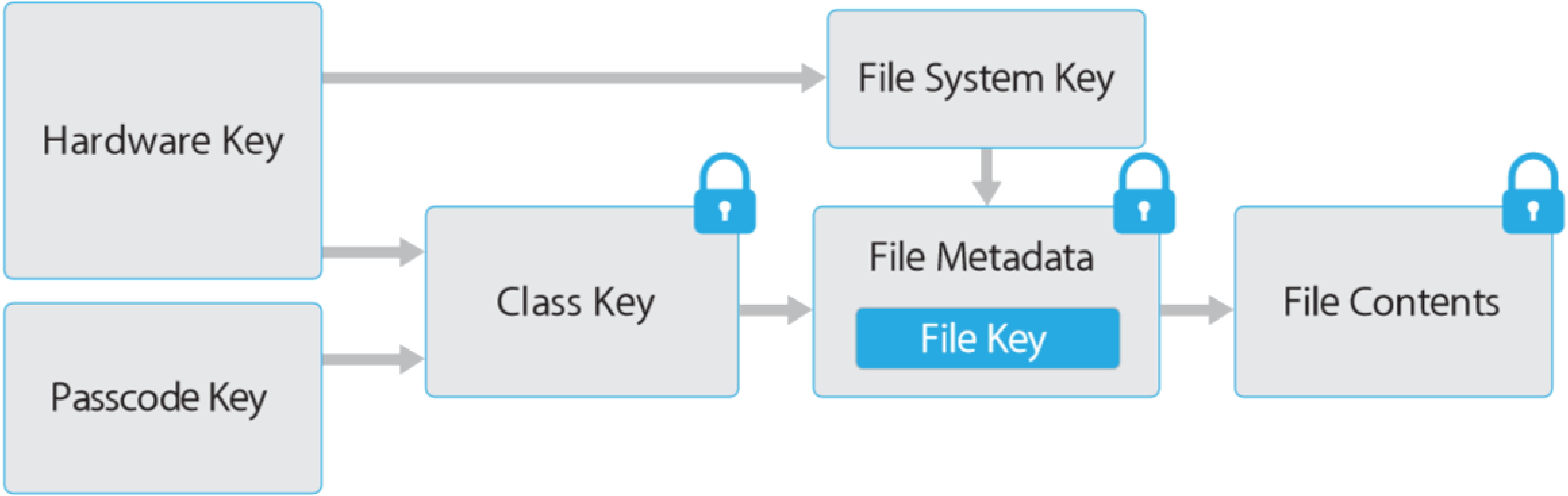
$\text{Wrap}_{K3}(\text{ek})$

$\text{Enc}_{ek}(\text{sector})$

Case study: Apple's iOS

- Encrypt data at the filesystem level
- Key wrapping ensures that data is only decryptable in the right *context*
- Hardware provides some protection against brute-forcing passwords

Hierarchy of keys



Master key generated from things that you know, are, and have

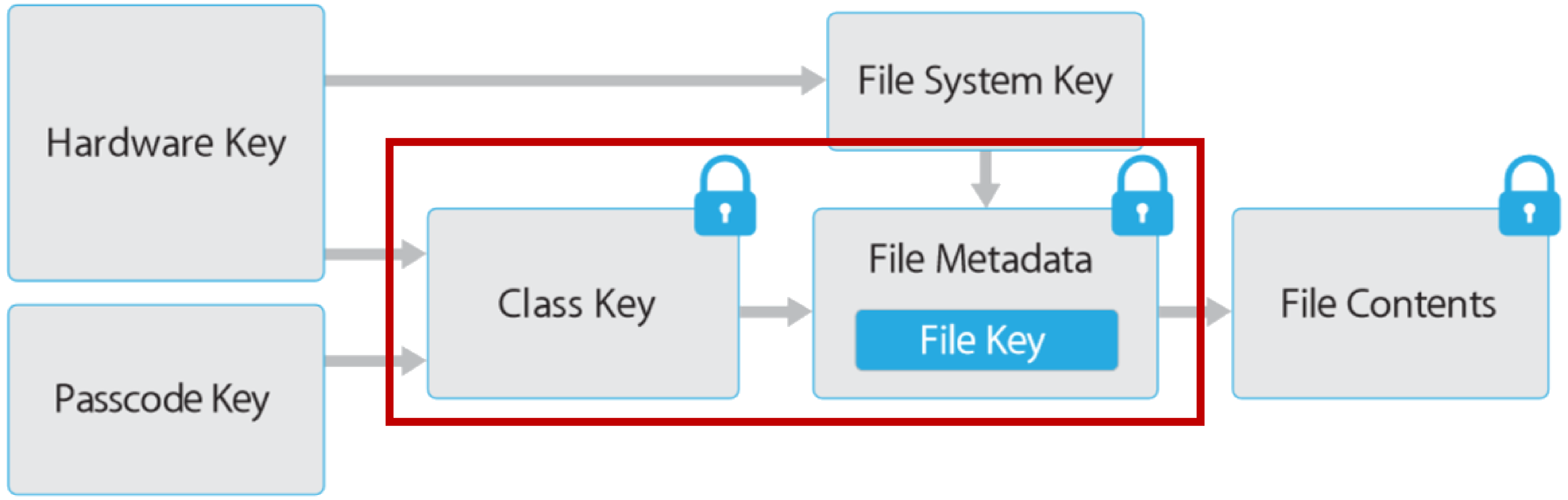
Keys are available for a limited time, and only when needed

Each file is encrypted with a unique key using AES-XTS. Working within filesystem gives space to store metadata on the side.

Class keys → file keys

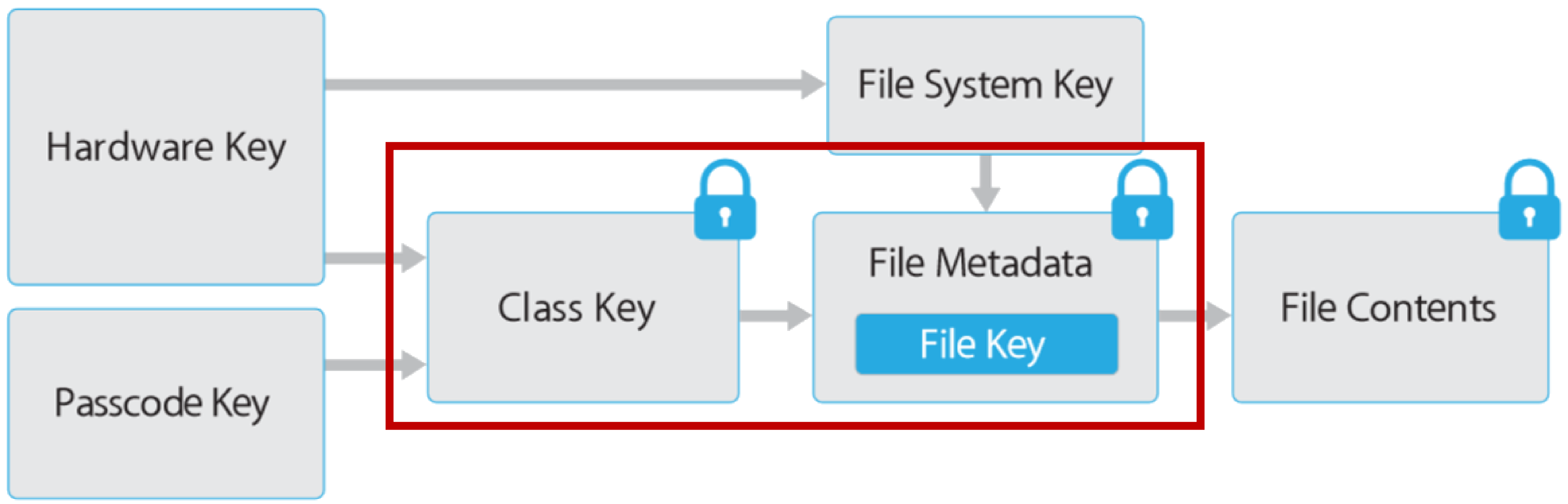
Per-file key is wrapped with 1 of 4 "class keys" based on availability

Availability	Example	Key erased if phone is...
Always	SIM PIN	Wiped
After 1st unlock	Wifi password	Shut down
When unlocked	Browser bookmarks	10s after lock (without biometric)
When locked	Incoming email	(works differently)

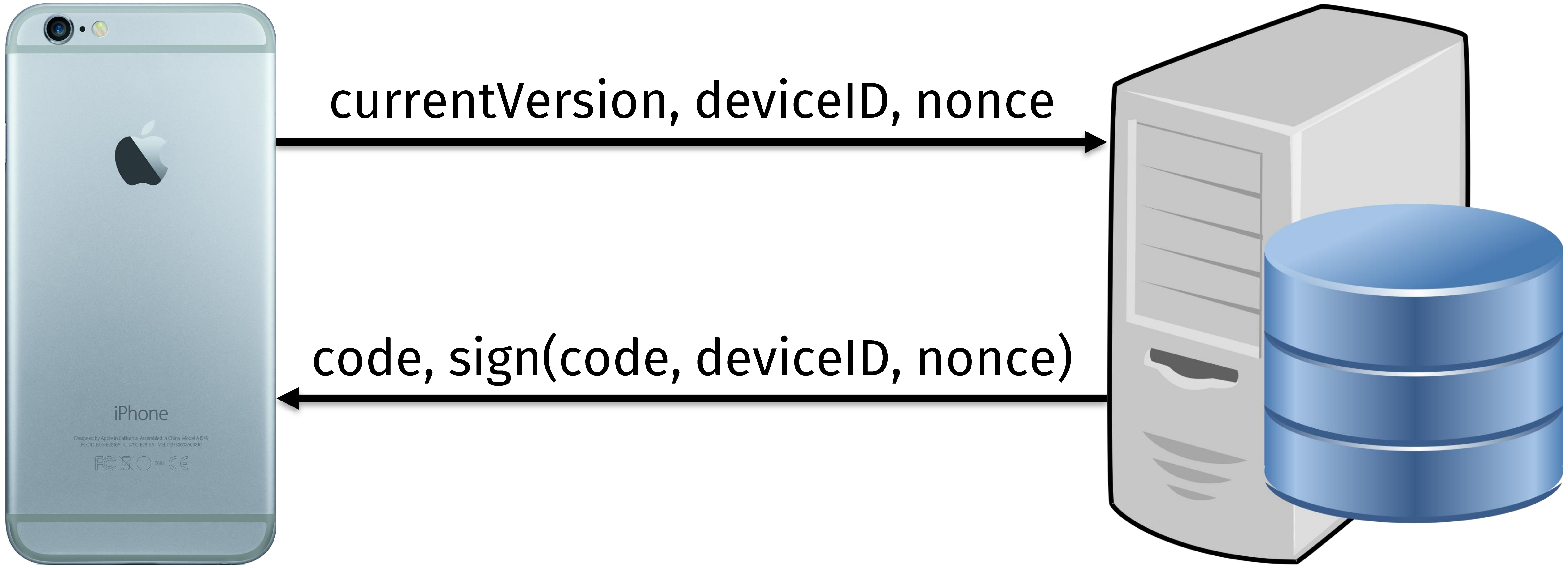


Secrets → class keys

- Rather than *storing* keys on the device, *derive* it from secrets
  - 256 bit string fused into phone
  - Alice's password
- Slow Mallory using
  - Crypto: 80ms per attempt
  - Hardware: increasing delays between attempts
  - Optionally, wipe the phone



iOS Software Updates



- Device ID *personalizes* the server's response to this particular phone
- Nonce ensures that response is *fresh*, prevents replay attacks
- Need: a *public signature scheme* that anybody can verify!

2. Protecting Data in Transit

Confidentiality

- Message privacy
- Withstand device compromise
- Deniability of transmission
- Entity privacy (aka anonymity)

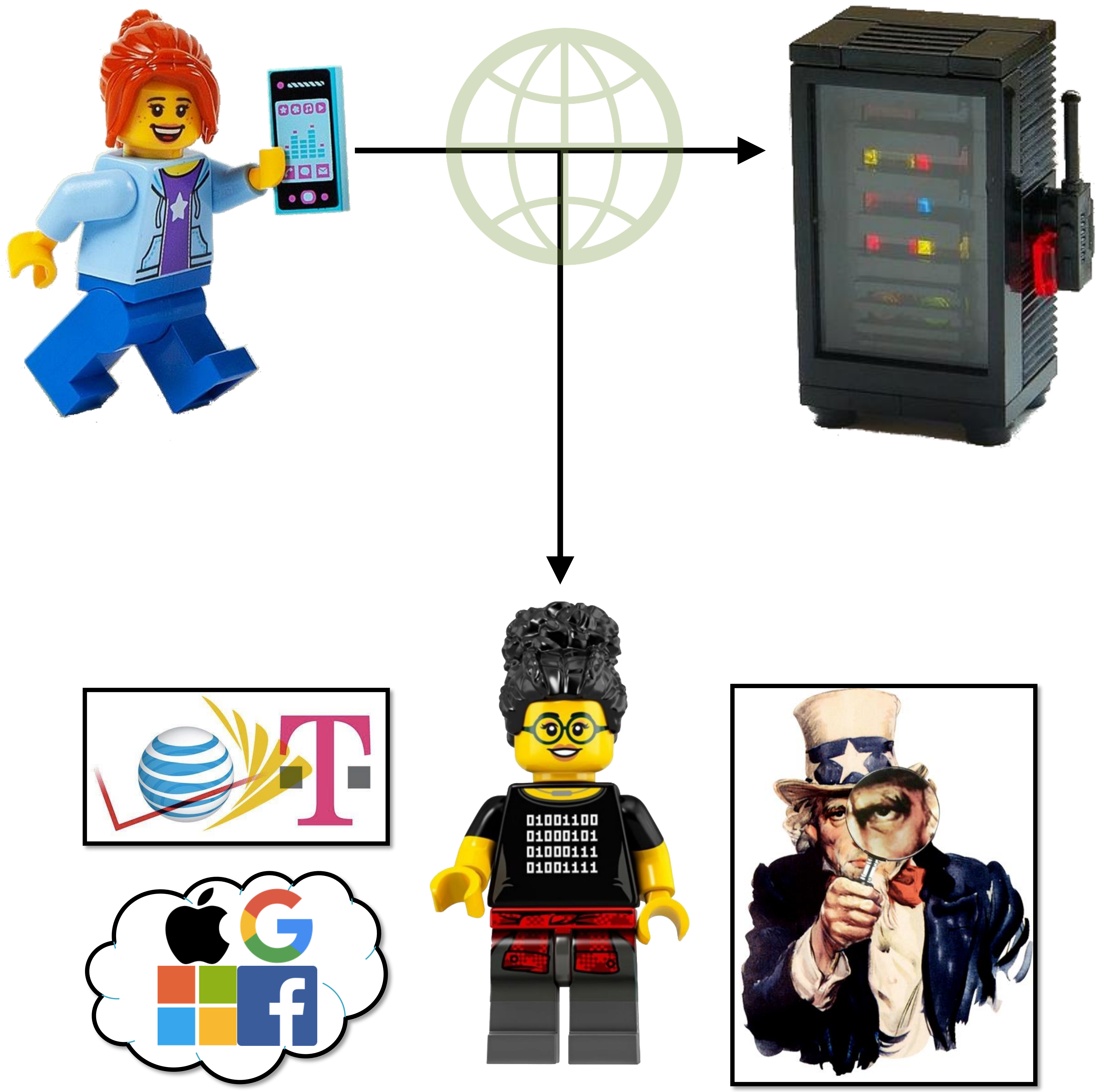
Integrity

- Message authenticity
- Entity authenticity
- Message binding / non-malleability
- Message freshness

Availability

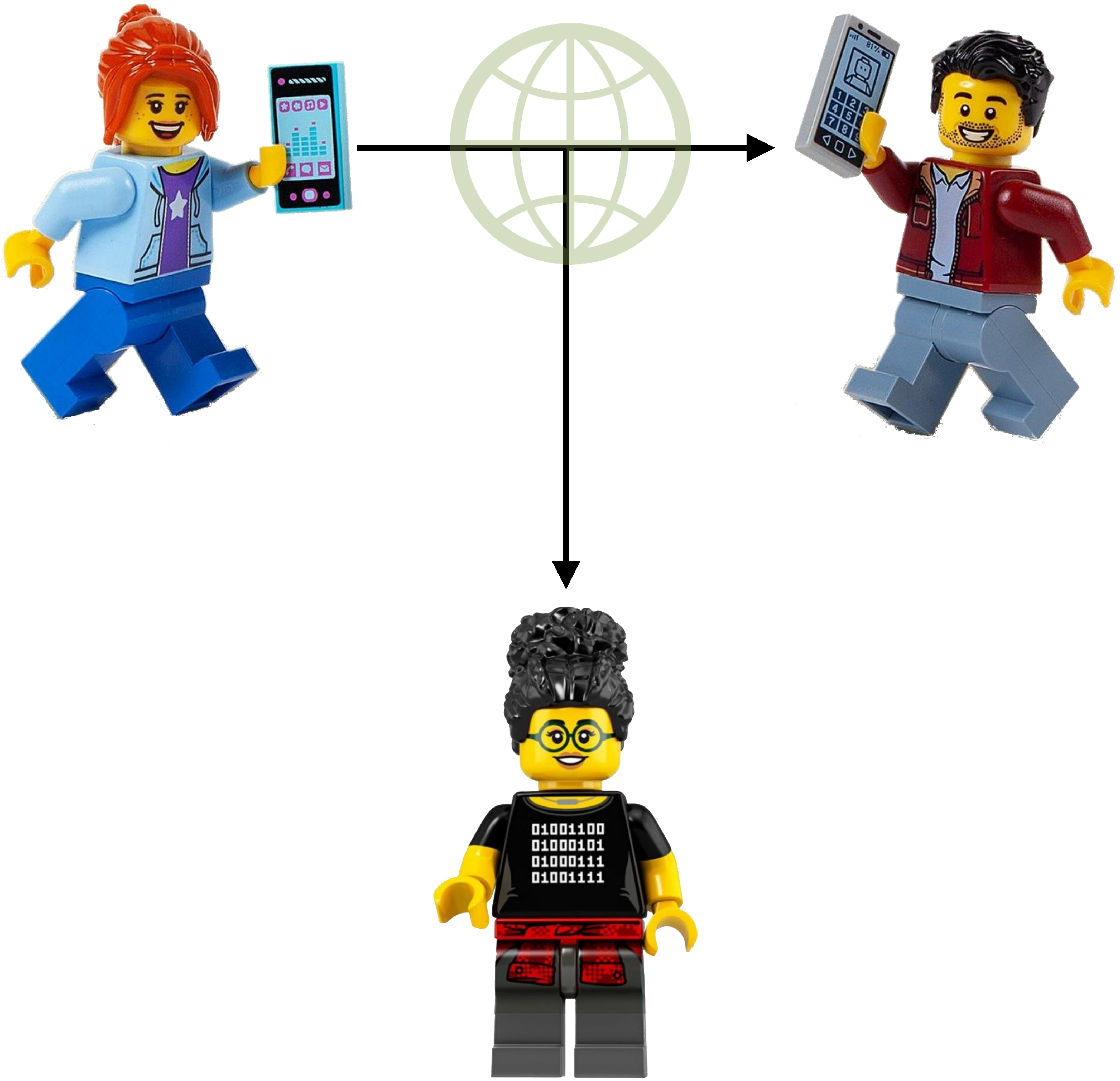
Scenario: client-server

- Alice wants to talk with a server Bob over the Internet
- They do *not* yet possess a shared secret key
- Our adversary Mallory can read, tamper, add, drop data in transit
  - Mallory is a stand-in for anyone that owns Internet infrastructure



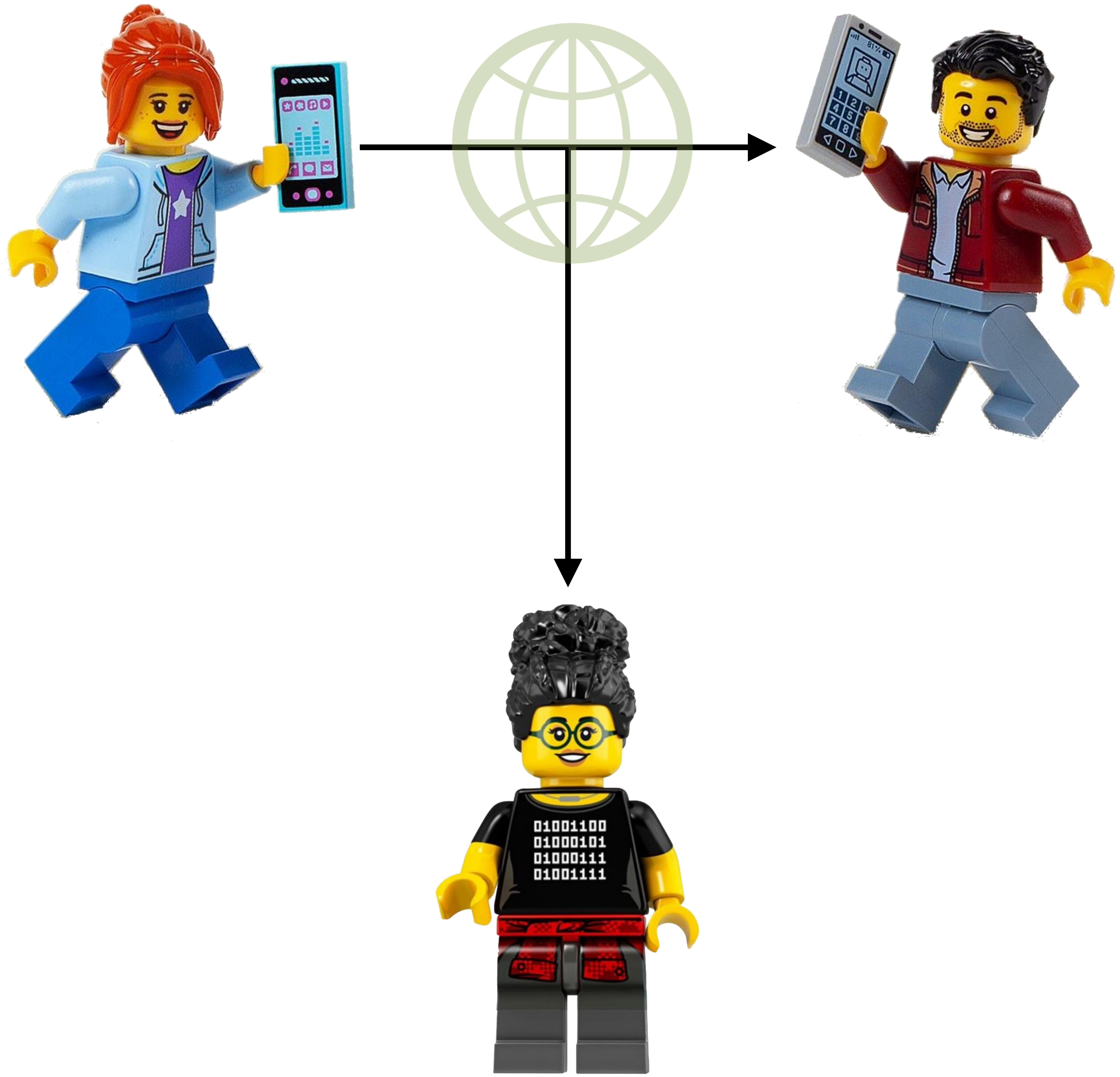
Scenario: end-to-end

- Now Alice is communicating with Bob’s personal laptop/phone
  - They have never met before in person to exchange a key
- Protecting both scenarios involves almost identical crypto
  - I will focus on the end-to-end scenario in this class



Objectives

- Protection from network
  - Message confidentiality
  - Sender authenticity + msg binding
- Protection from endpoints
  - Secrecy before/after compromise
  - Sender deniability
- Non-goals
  - Hiding metadata (e.g., Alice and Bob's identity, message size)
  - Stopping replay, delay, re-ordering



Objectives

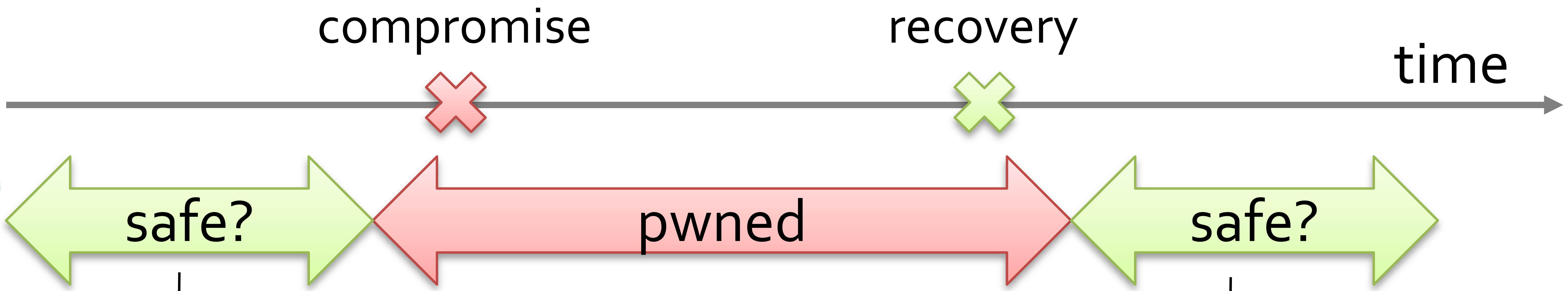
- Protection from network
  - Message confidentiality
  - Sender authenticity + msg binding
- Protection from endpoints
  - Secrecy before/after compromise
  - Sender deniability
- Non-goals
  - Hiding metadata (e.g., Alice and Bob's identity, message size)
  - Stopping replay, delay, re-ordering

AuthEnc will protect communication on the network, if Alice and Bob already have a shared key K

This is new to us…

3. Forward + backward secrecy and deniability

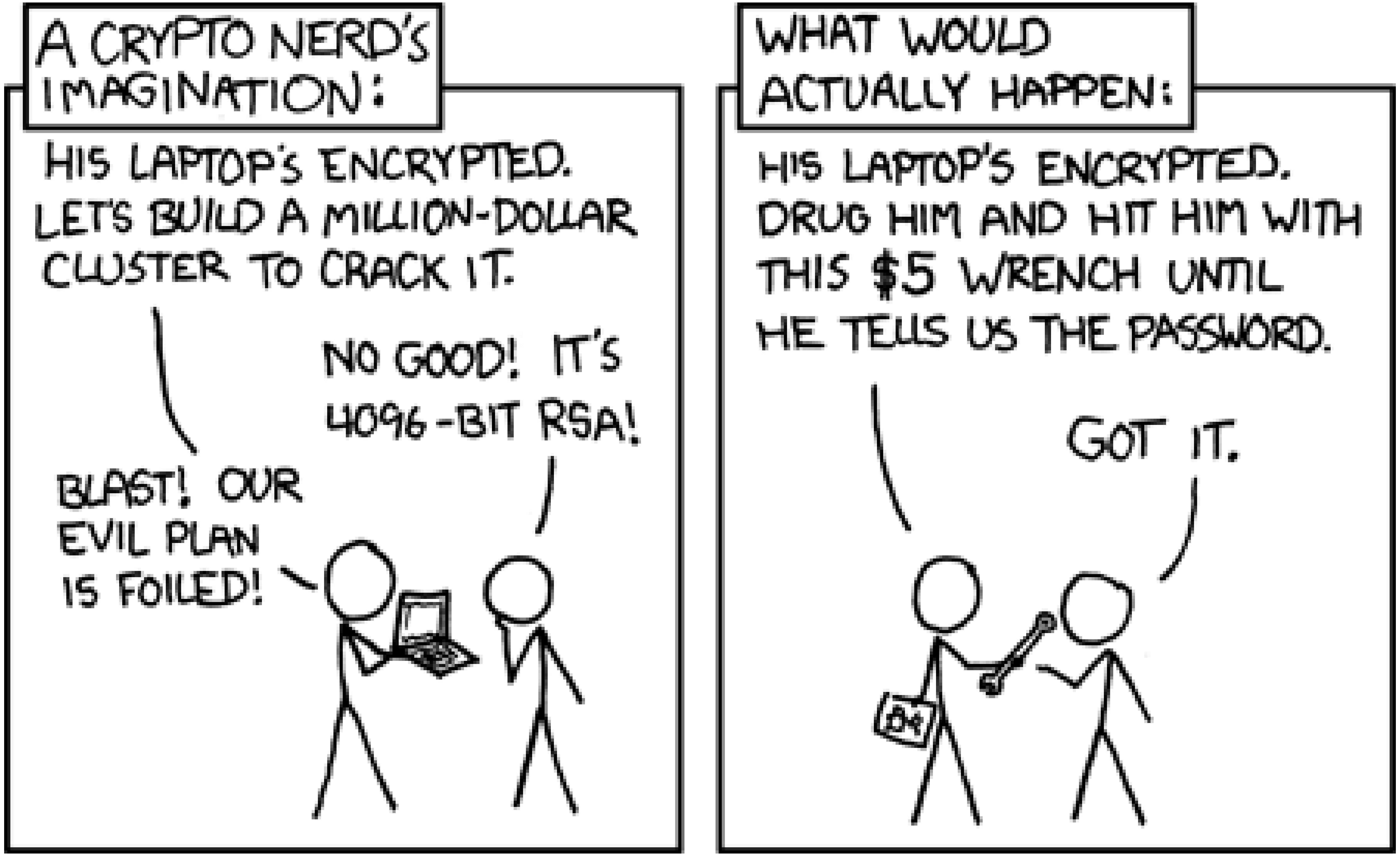
Forward and backward secrecy



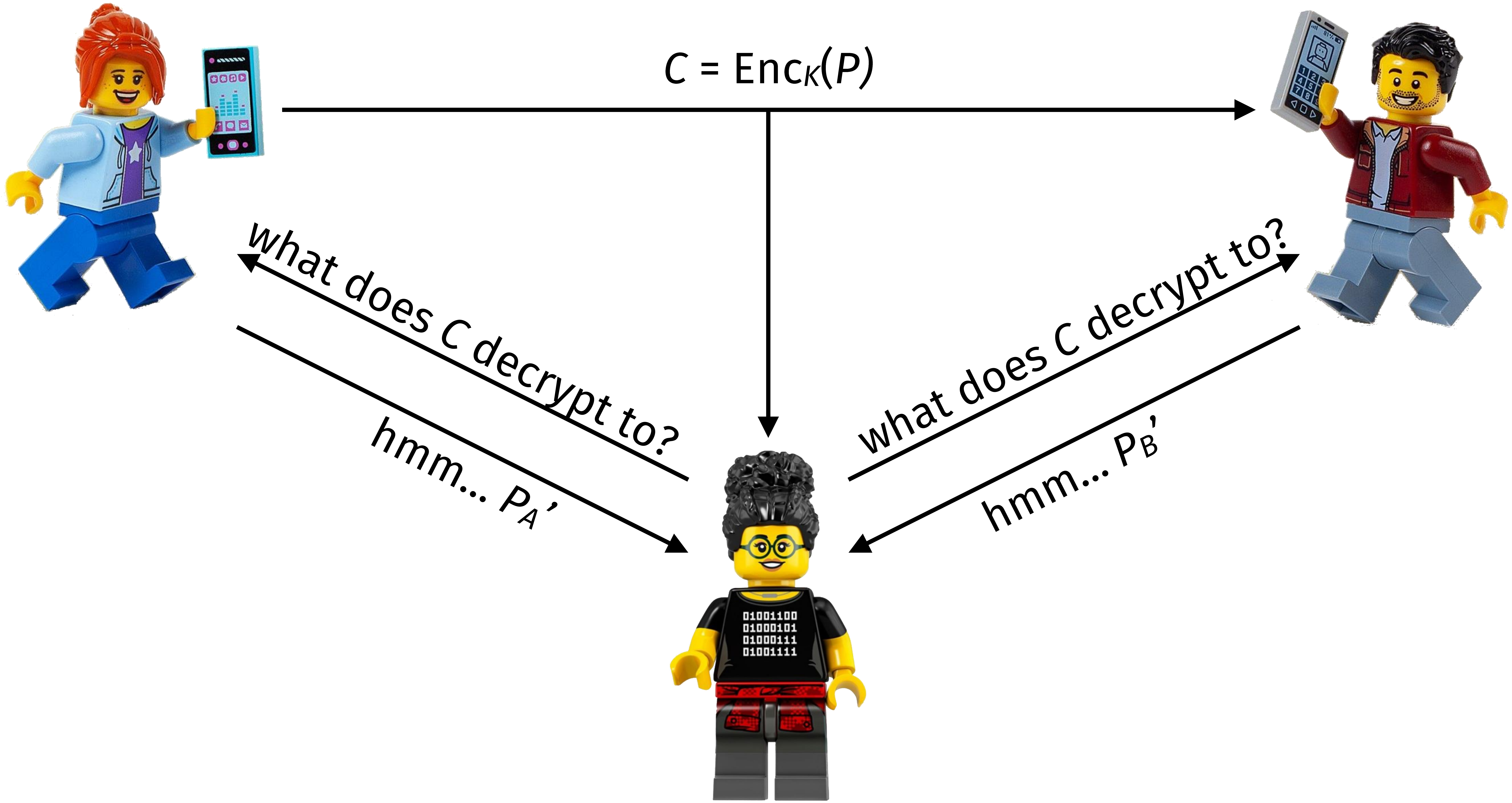
Forward (pre-compromise) secrecy
past messages remain private even if secret state is exposed later

Backward (post-recovery) secrecy
future messages remain private even if secret state is compromised and then the device recovers

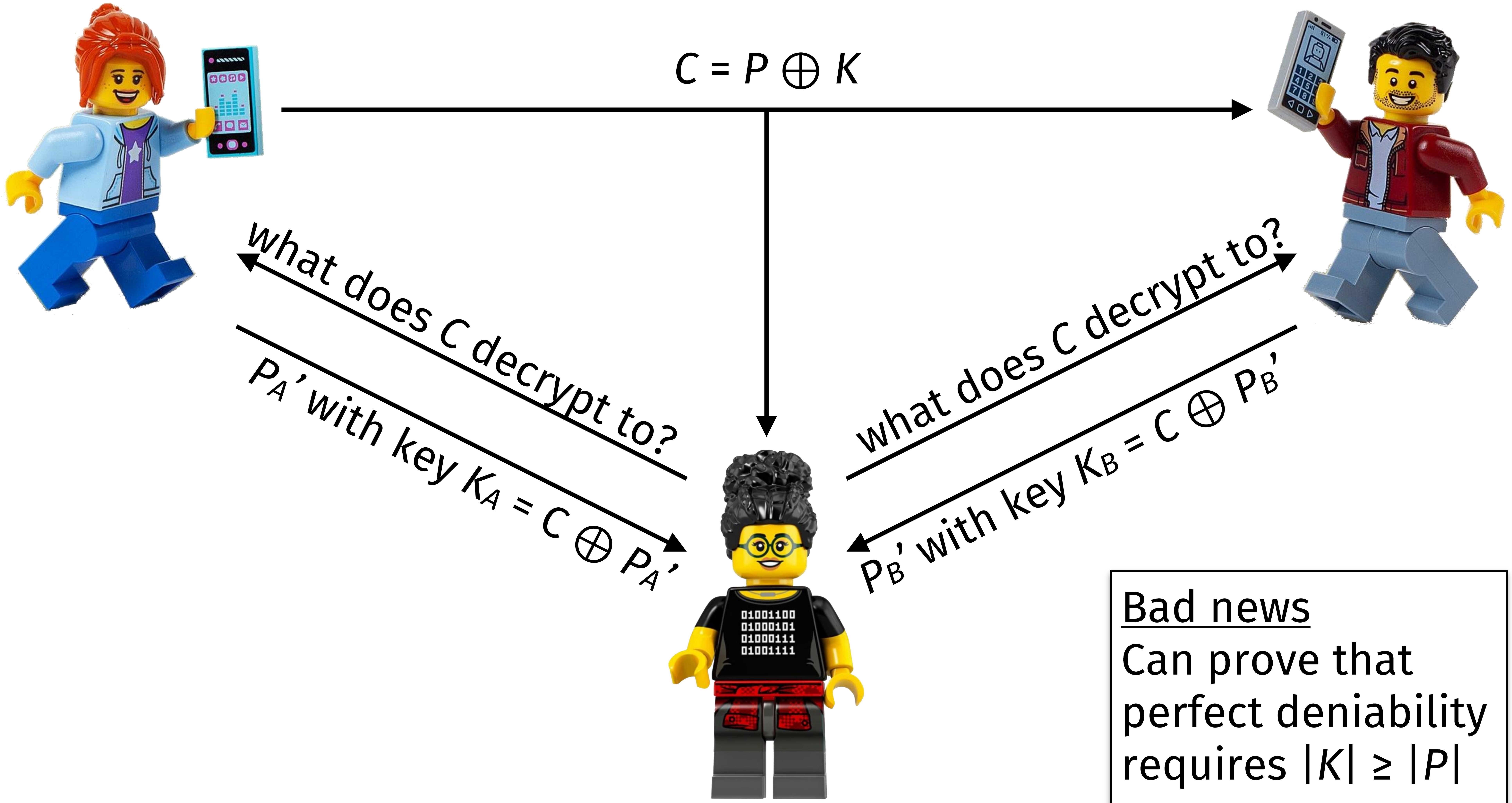
Non-deniable crypto (xkcd.com/538)



Deniable crypto = can pretend you said something else



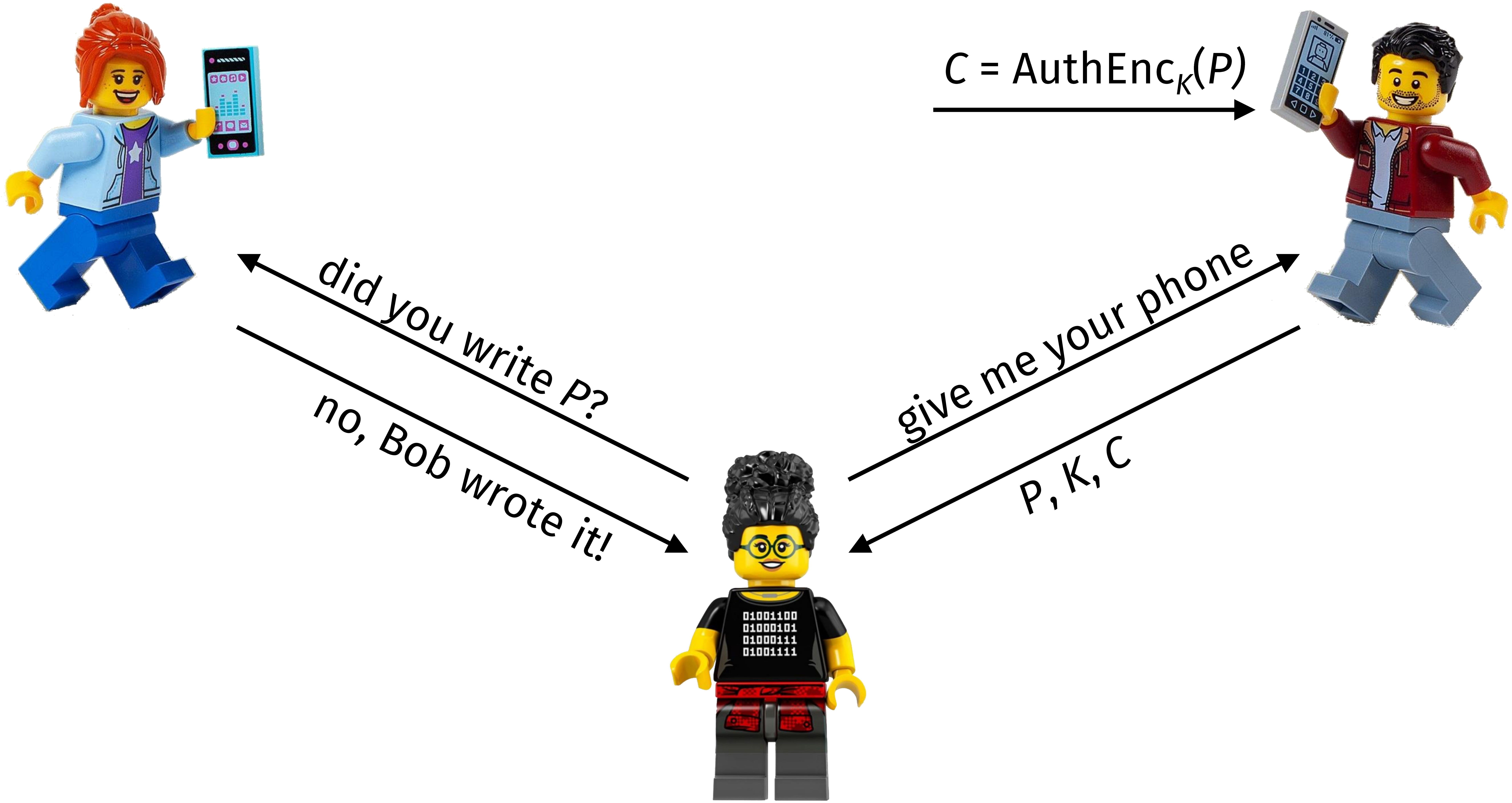
One time pad → perfect deniability



Bad news

Can prove that perfect deniability requires $|K| \geq |P|$

Auth encryption → partial sender deniability



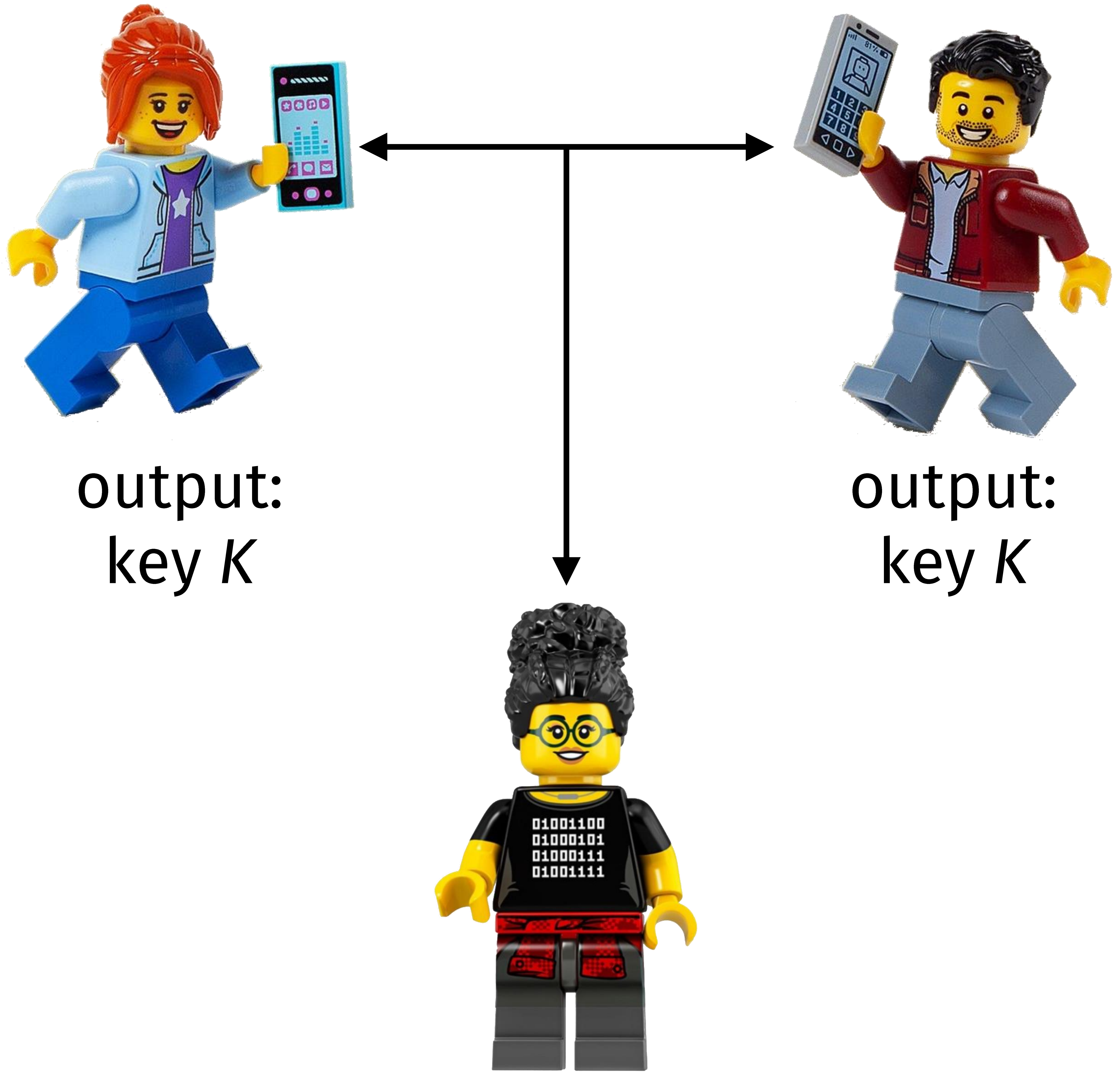
Use authenticated encryption… but with what key?

- Key exchange
  - Alice and Bob want to generate a shared key without ever having met before
  - Assistance from a partially-trusted entity that mediates this connection
- Key evolution (aka ratcheting)
  - Use each key to protect just 1 message, then *delete it*!
  - Protect message privacy + integrity against device compromise in past + future
  - Generate a new key for the next message

4. Key Exchange

Scenario

- Alice and Bob want to agree on a shared symmetric key by talking over the Internet
- Adversary observes all network communications
- Can Alice and Bob learn the key without Eve/Mallory doing so?

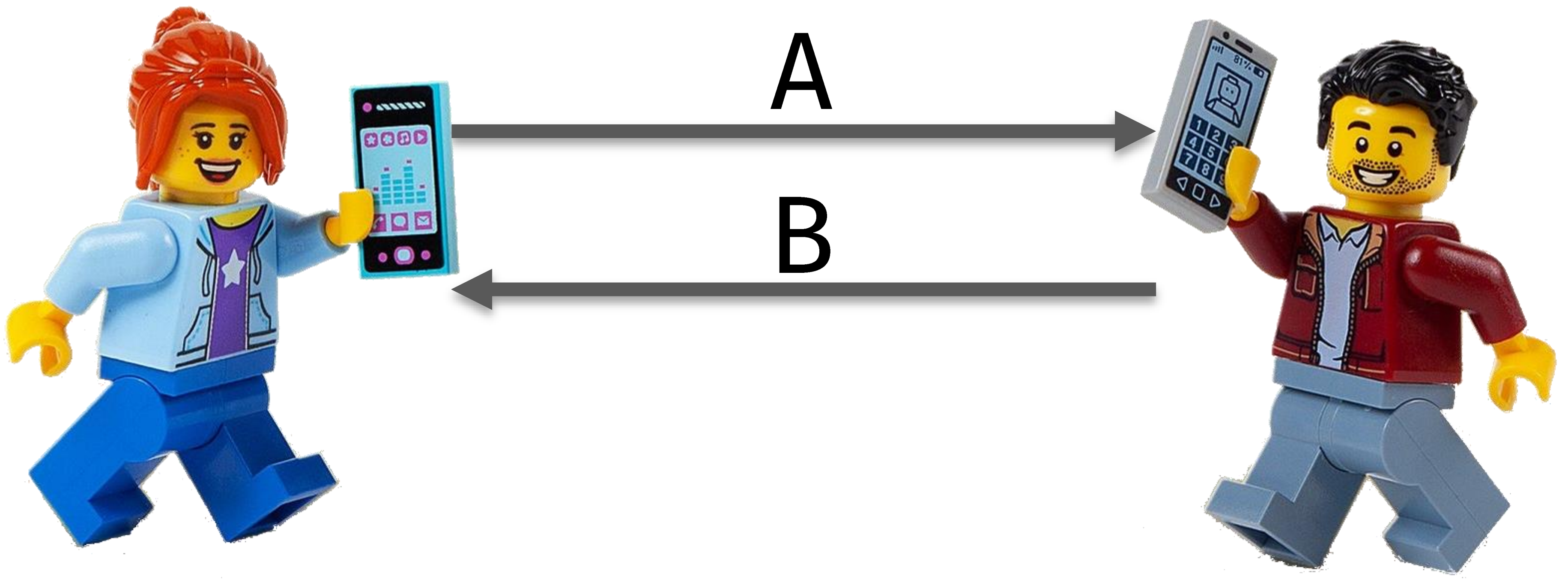


Diffie-Hellman key agreement (vs passive Eve)

Protocol (for a publicly known g)

Choose *a* randomly
Compute A = g^a

Choose *b* randomly
Compute B = g^b



Output *K* = B^a

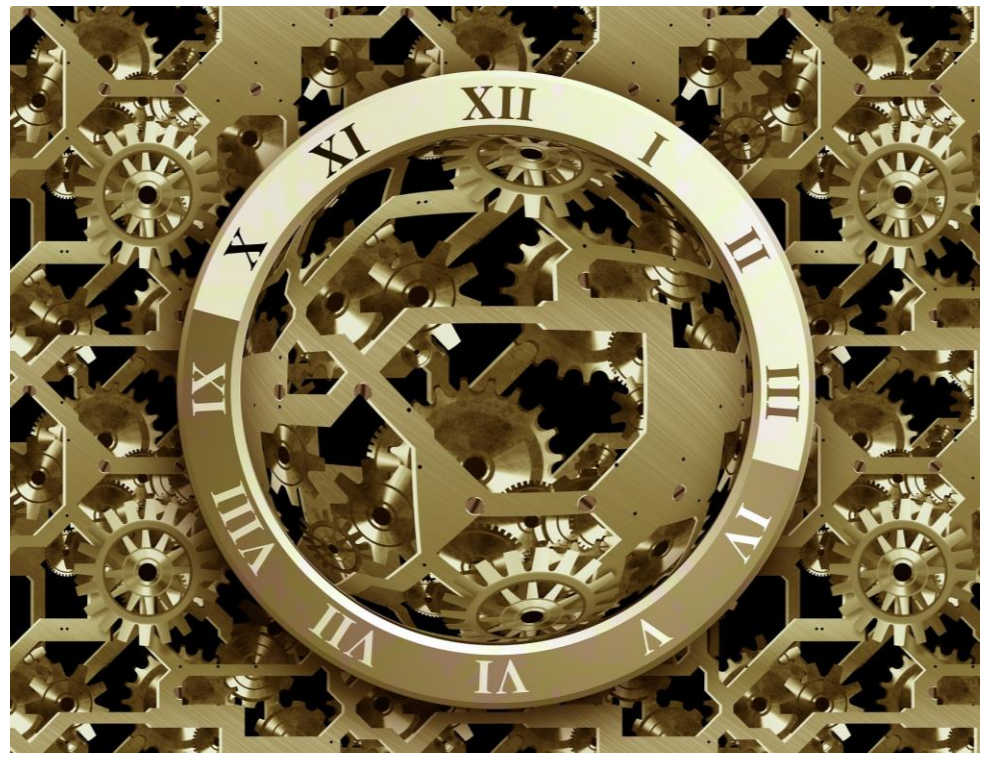
Output *K* = A^b

Analysis

- *Correctness*: shared secret since $A^b = (g^a)^b = g^{ab} = (g^b)^a = B^a$
- *Secrecy*: to learn *K*, a passive Eve given g, g^a, g^b must find g^{ab}
  - There exist mathematical spaces in which this problem is hard!
- *Forward secrecy*: Choices of a, b are ephemeral, delete afterward so even you cannot compute *K*

How to perform key exchange securely?

Modular arithmetic

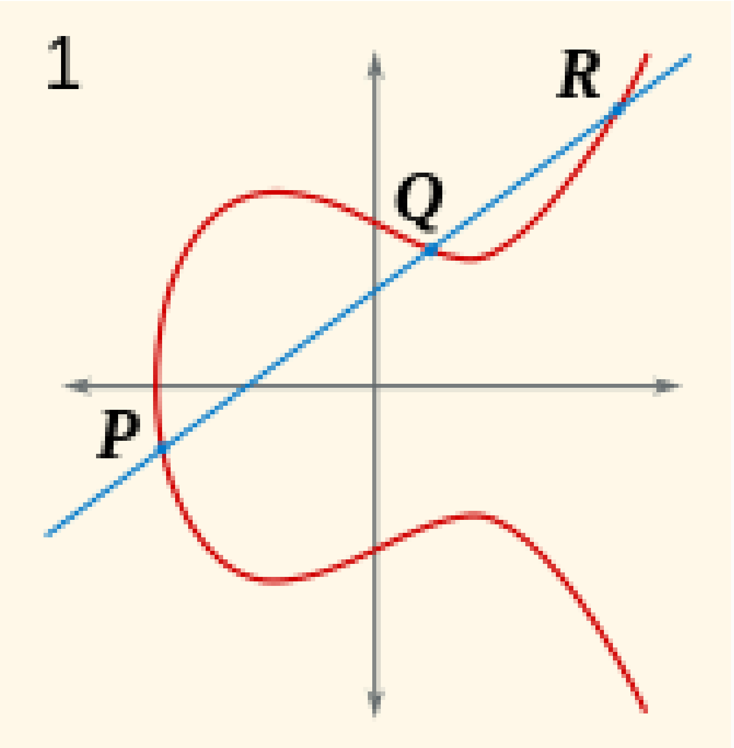


- Raise a constant to any power, e.g. $x \mapsto 3^x \pmod 7$

x	1	2	3	4	5	6
3^x	3	2	6	4	5	1

- Permutation, but hard* to invert

Elliptic curves

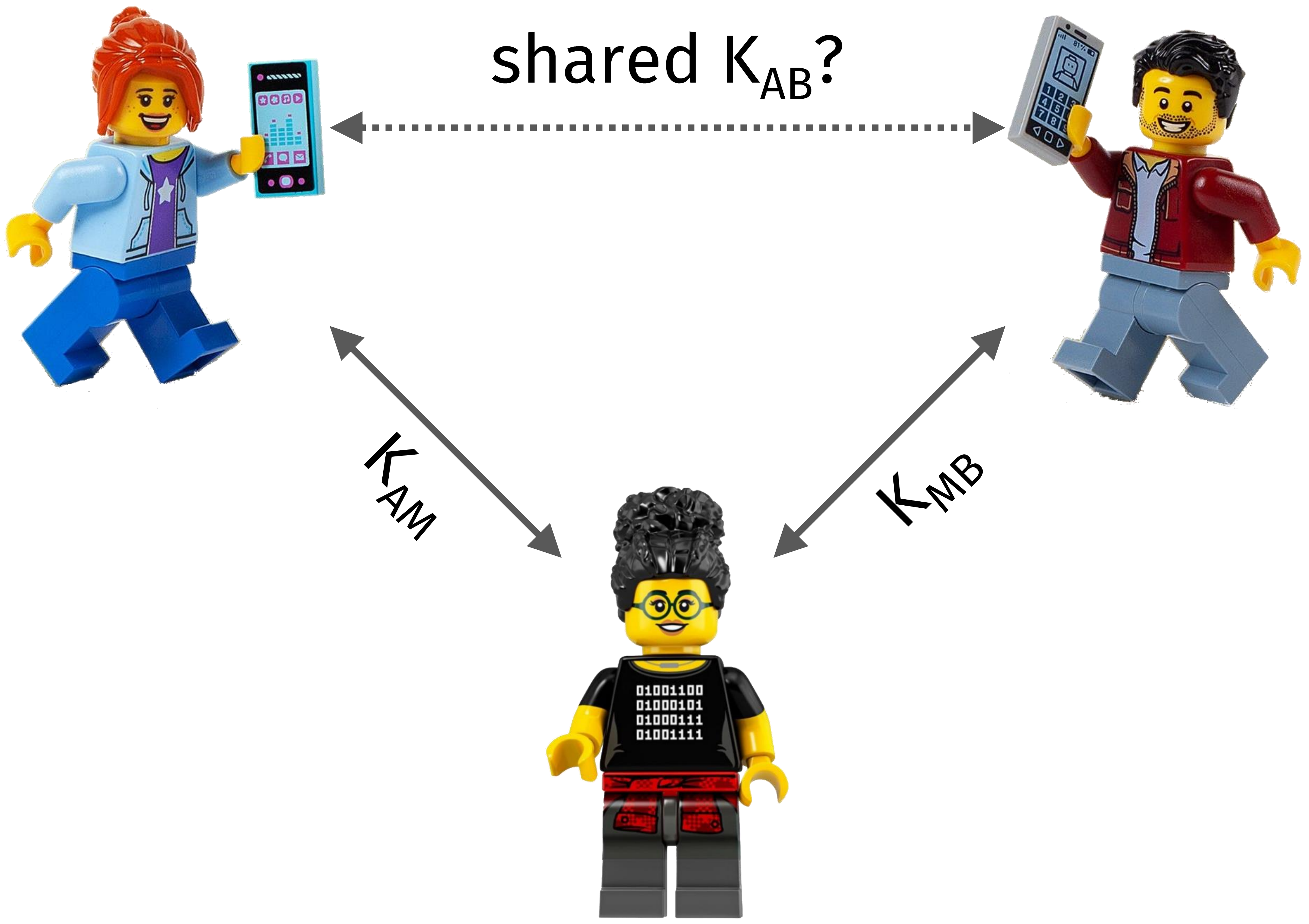


- Elliptic curve: a cubic equation $y^2 = x^3 + ax + b \pmod p$
- Consider set of points on this curve
- We can "multiply" points using the rule $P \cdot Q \cdot R = 1$

* = really need to take the group of quadratic residues (i.e., the even half of the truth table)

Diffie-Hellman key agreement (vs active Mallory)

Active attacker causes problems!



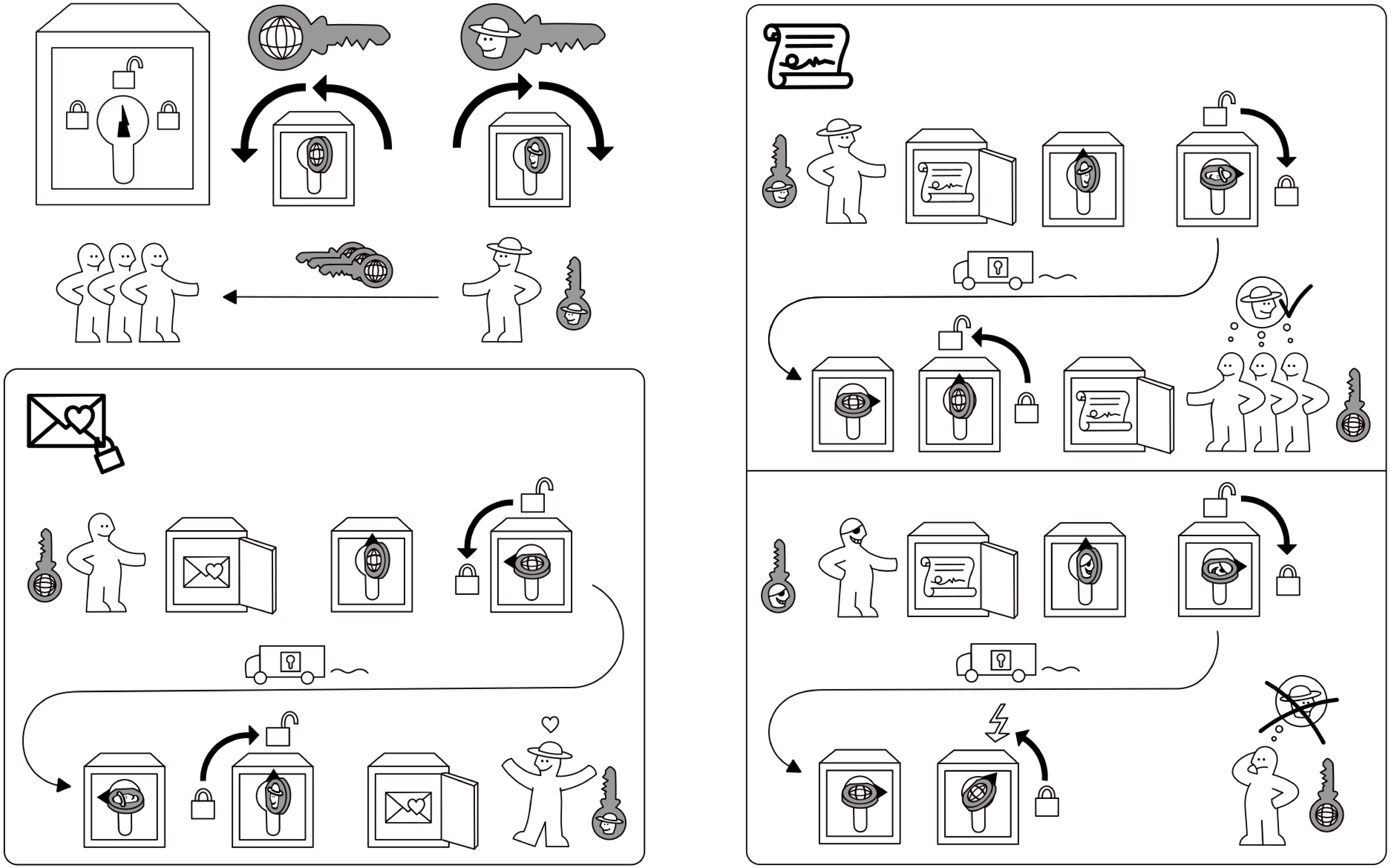
- Q: How do Alice and Bob verify they're talking with each other?
- A: Use a MAC?

5. Public Key Cryptography

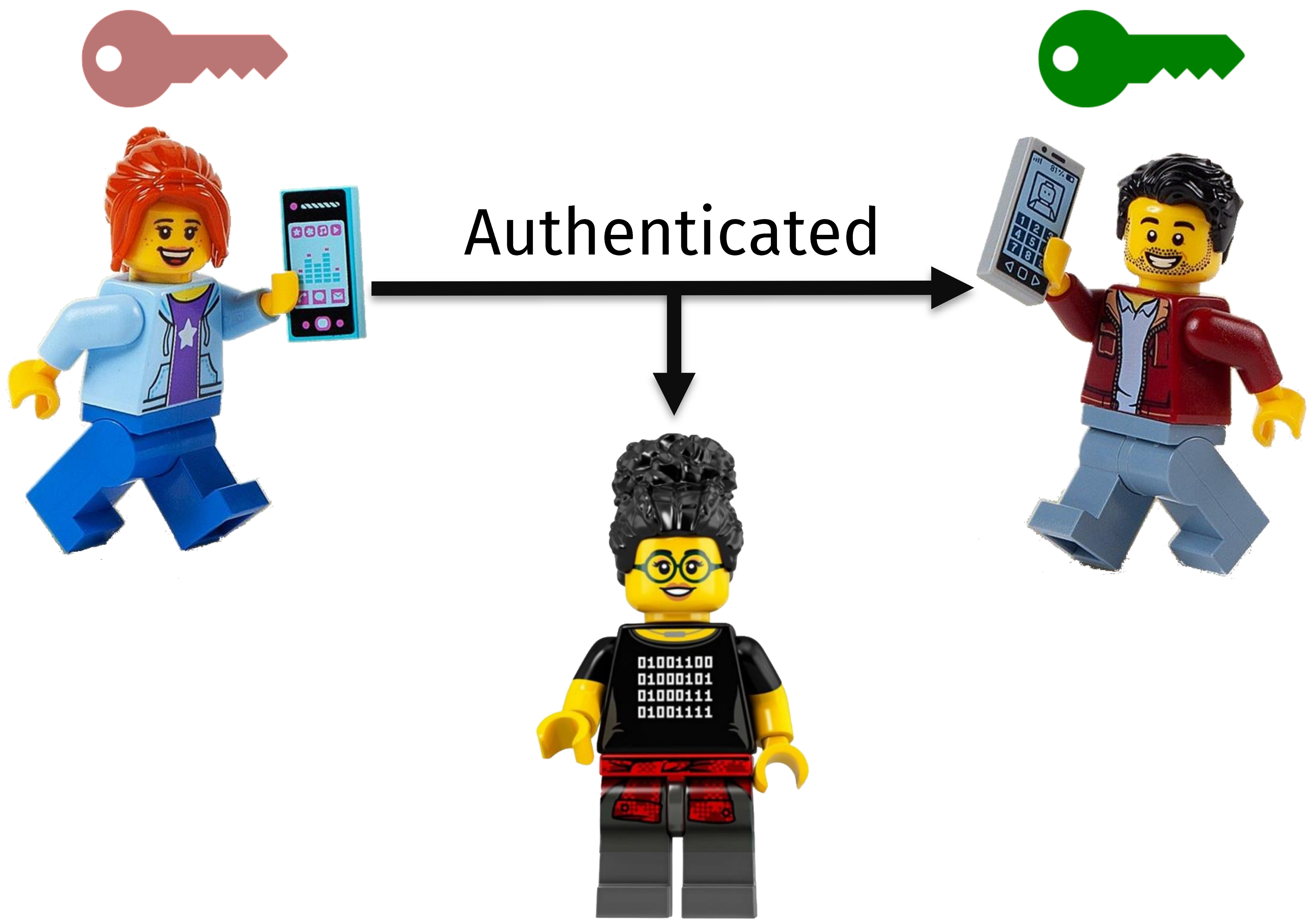
PUBLIC KEY KRÜPTO

idea-instructions.com/public-key/

IDEA

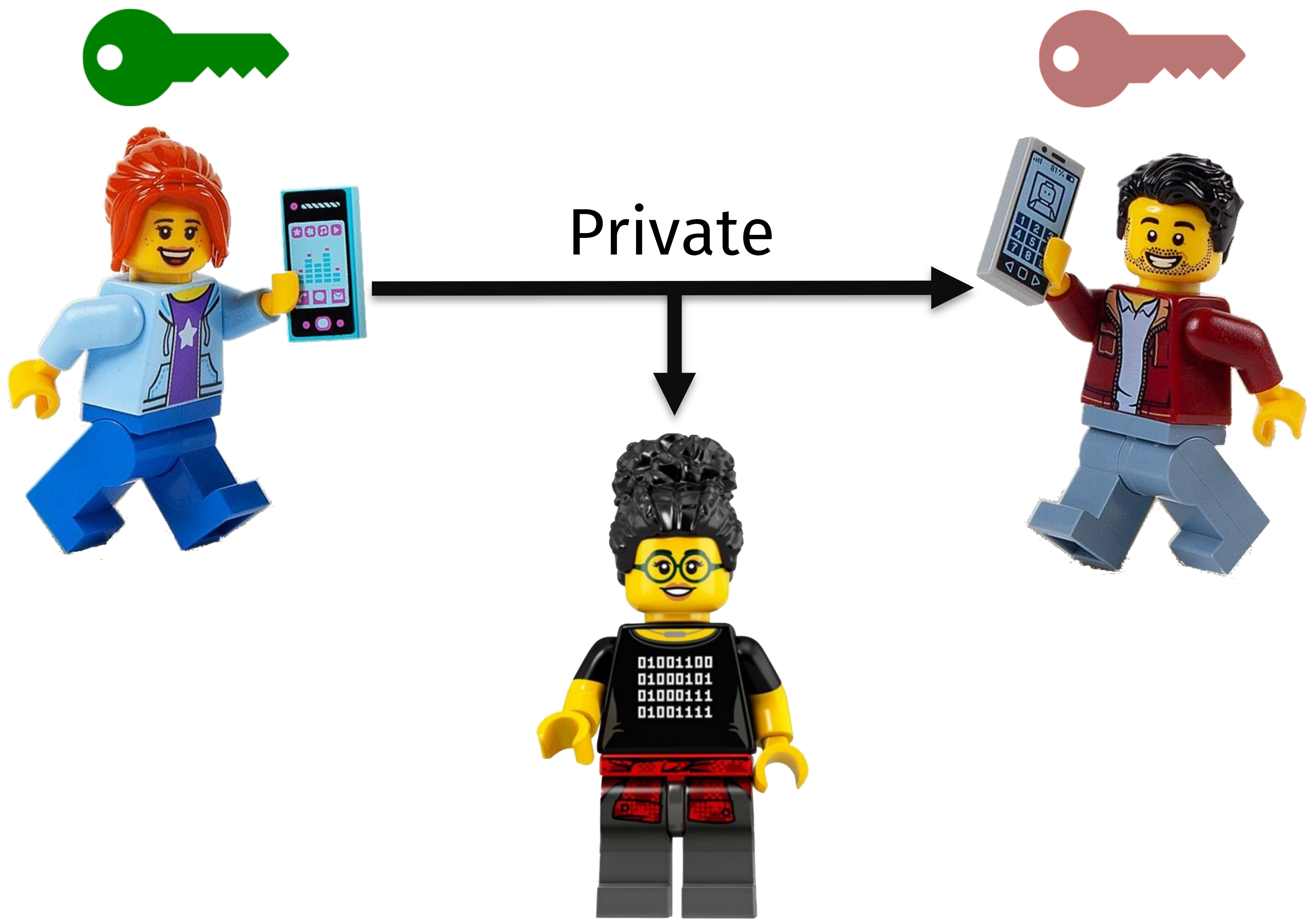


Public key signatures



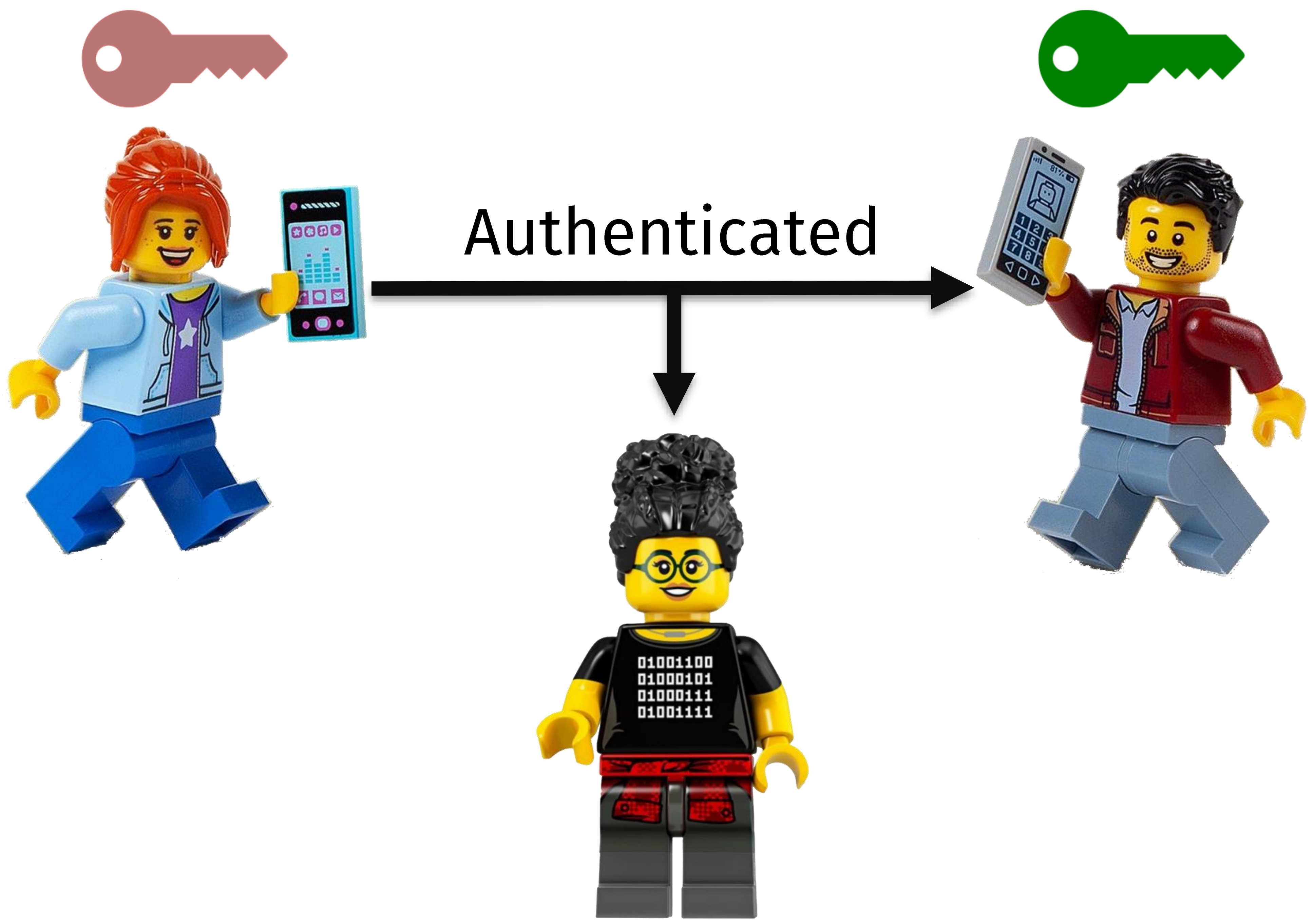
- Only Alice can generate signatures
- Anybody can verify
- Security guarantee: EU-CMA

Public key encryption



- Anybody can send ciphertexts
- Only Bob can decrypt + read
- Security guarantee: CPA or CCA

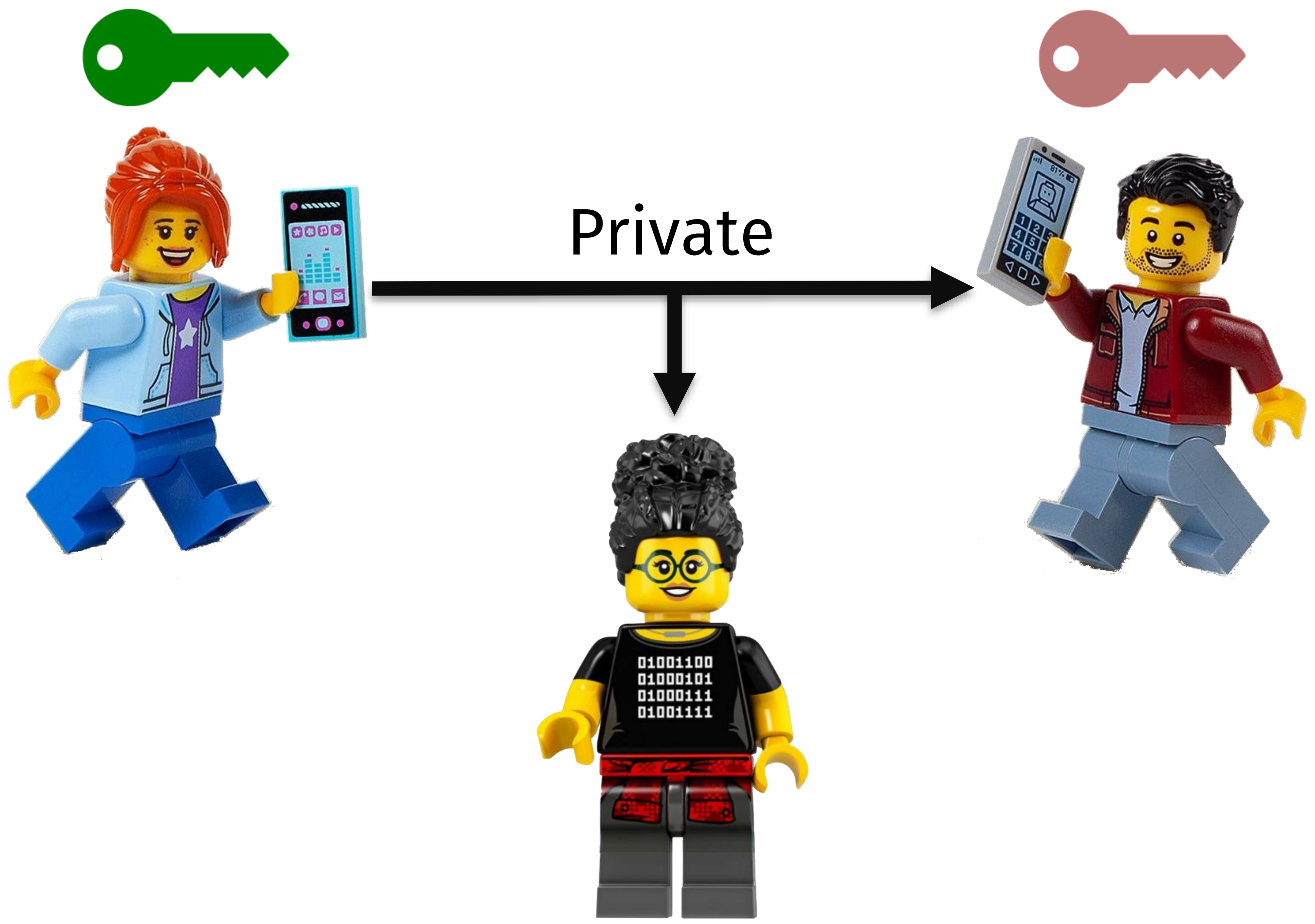
Public key signatures



Consequence if Mallory gets ?

- Problem: Eve forges msgs in *future*
- Response: Alice can *revoke* her key

Public key encryption

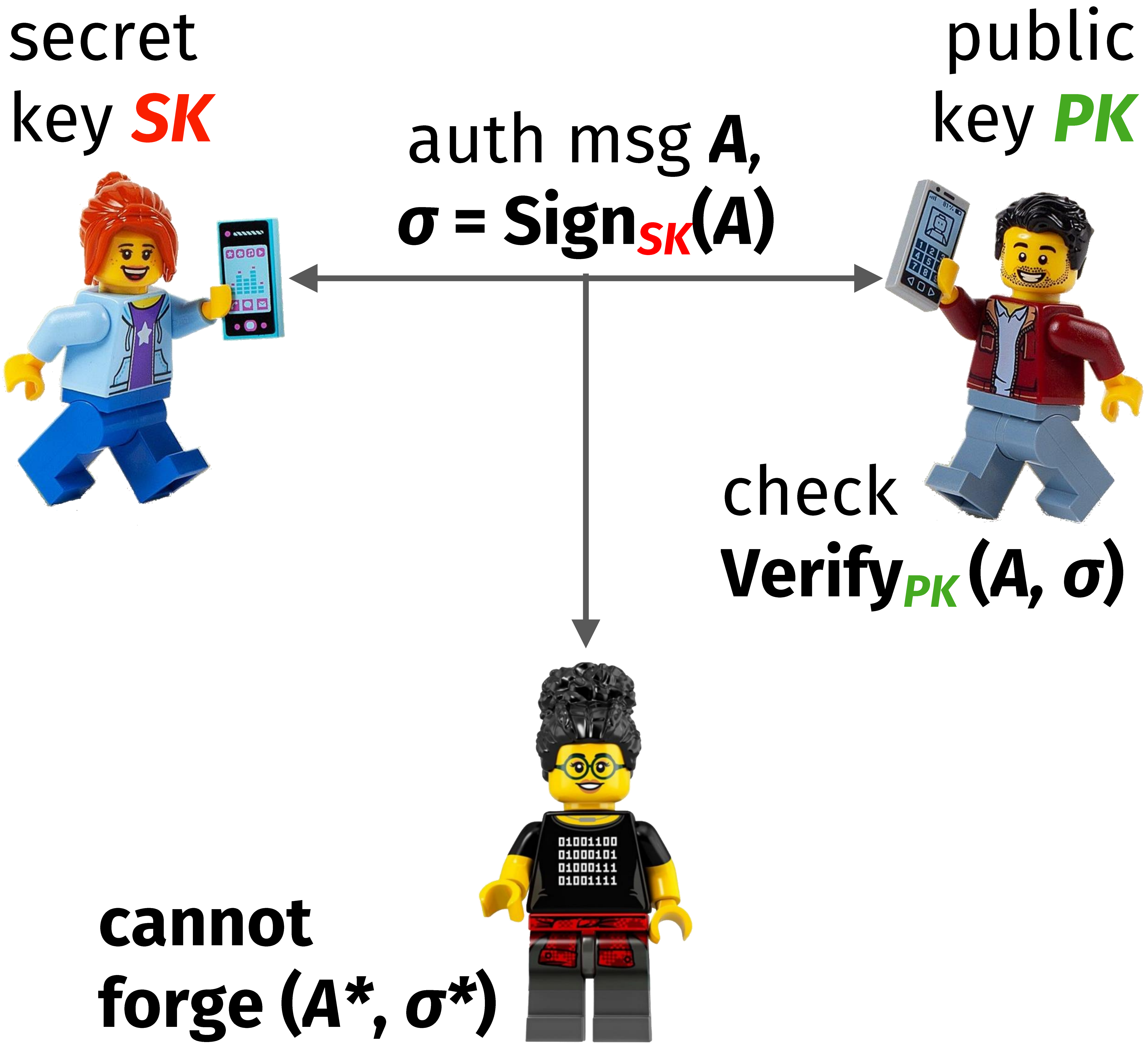


Consequence if Mallory gets ?

- Problem: Eve reads msgs from *past*
- Response: ??? (dangerous to use!)

6. Public key digital Signatures

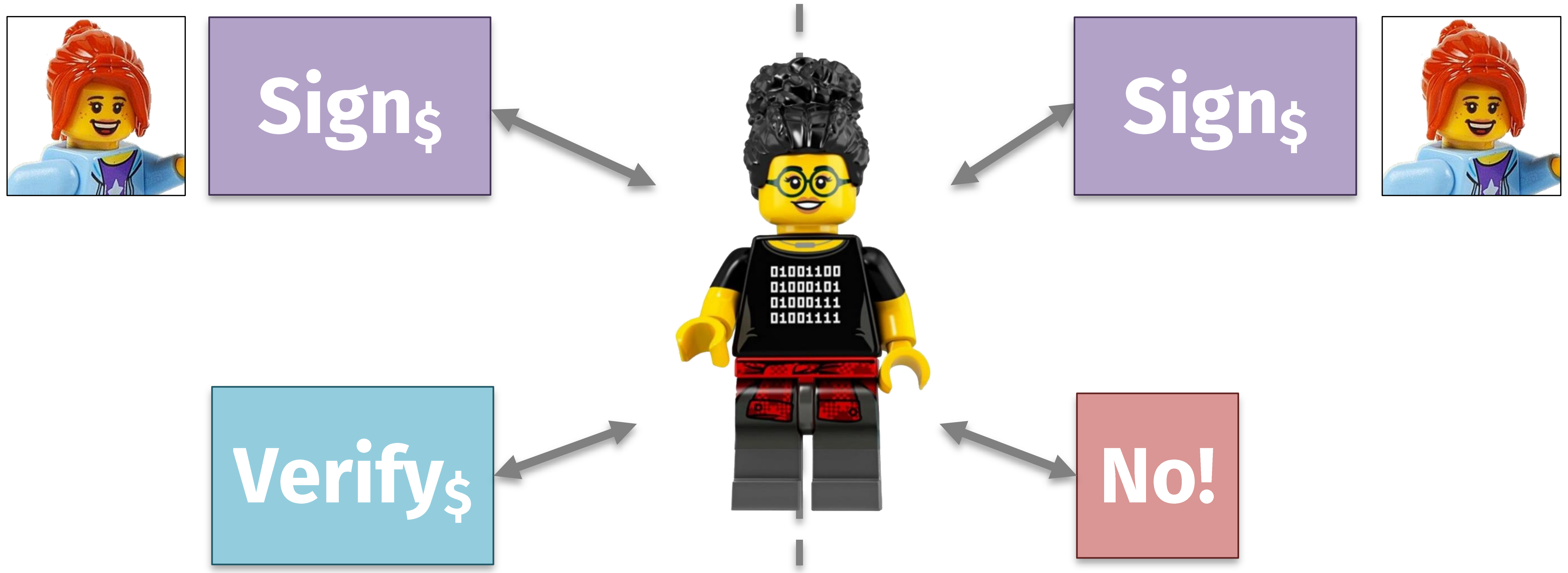
Digital signatures provide *public* authentication



- Symmetric MACs: Alice & Bob have shared key, nobody else knows it
  - MACs provide full authenticity and partial sender deniability
- Public signatures: Alice has secret *SK* and everyone knows *PK*
  - Message + sender authenticity: only Alice can make valid signatures
  - No receiver authenticity: we don't know the intended destination of *A*
  - Not deniable: Bob cannot make σ

EU-CMA security for digital signatures

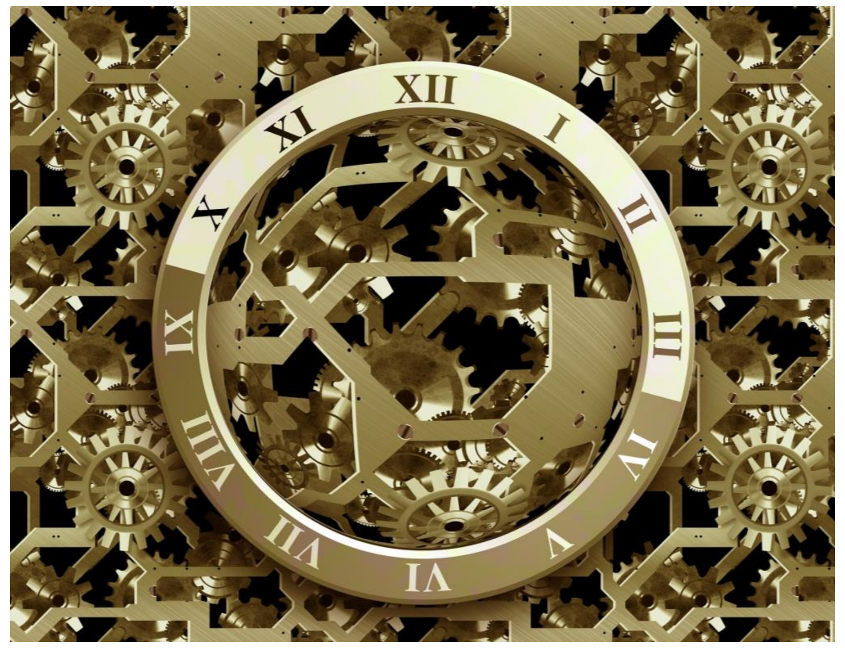
Security for digital signatures is formalized the same way as for MACs



How to make digital signatures?

Modular arithmetic

- Similar math as with key exchange



- Two common methods
  - (EC)DSA — NIST standard
  - Schnorr signatures — simpler but patented

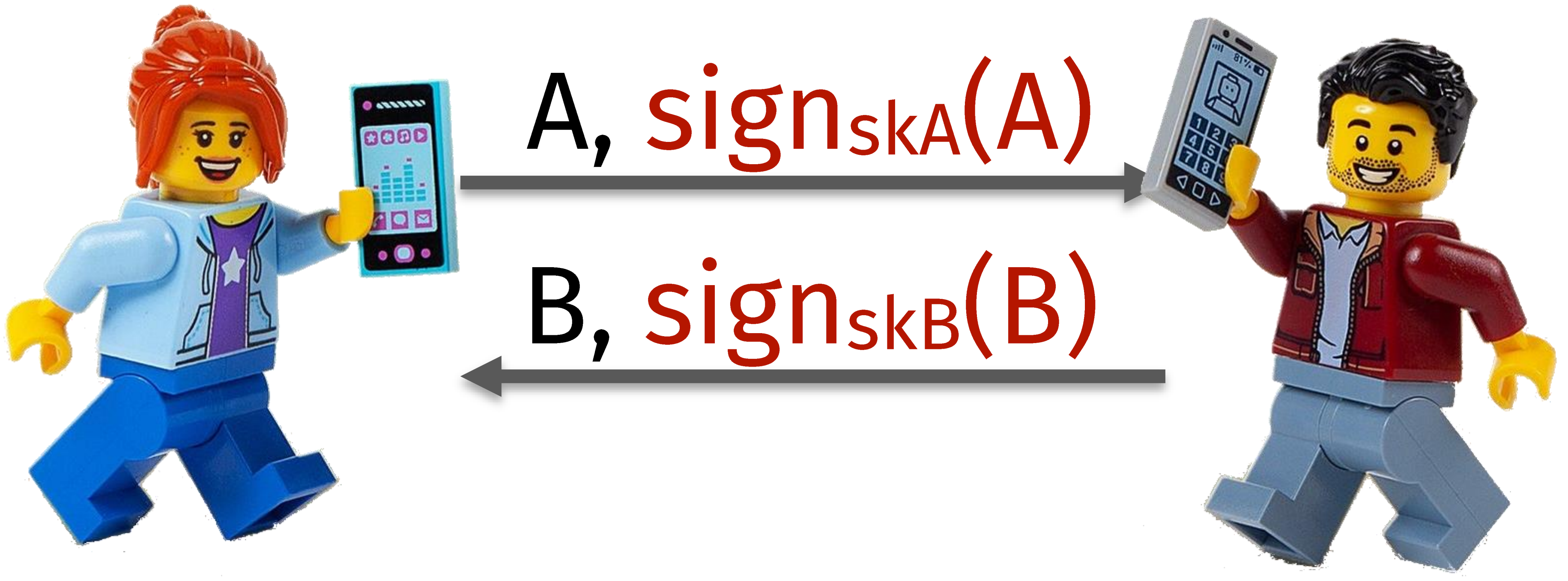
RSA (Rivest, Shamir, Adleman)

- Relies (more or less) on the hardness of factoring N = p q
- Less commonly used nowadays
- Will explore in Friday's recitation section and HW 7

Authenticated key exchange

Choose a randomly
Compute $A = g^a$

Choose b randomly
Compute $B = g^b$



1. Alice and Bob sign their messages during Diffie-Hellman key exchange
2. Alice and Bob verify the signature of each other's message
3. Use shared key $A^b = B^a$ for (deniable) symmetric authenticated encryption

Google.com in Firefox:

Technical Details

Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)

BU login page in Firefox (2017):

Technical Details

Connection Encrypted (TLS_RSA_WITH_AES_256_CBC_SHA, 256 bit keys, TLS 1.2)

Question for next time: how do Alice and Bob learn each other's public keys?