Course Announcements

- HW 6 and 7 both due on Wednesday 3/25
- Homework 8 posted today, due Wednesday 4/1
- Reminder: if you want to ask a question during the virtual lecture, type it in the chat window

Lecture 15: Authenticated Key Exchange

1. Key exchange
2. Public key cryptography
3. Public key digital signatures
4. Digital certificates & the PKI

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

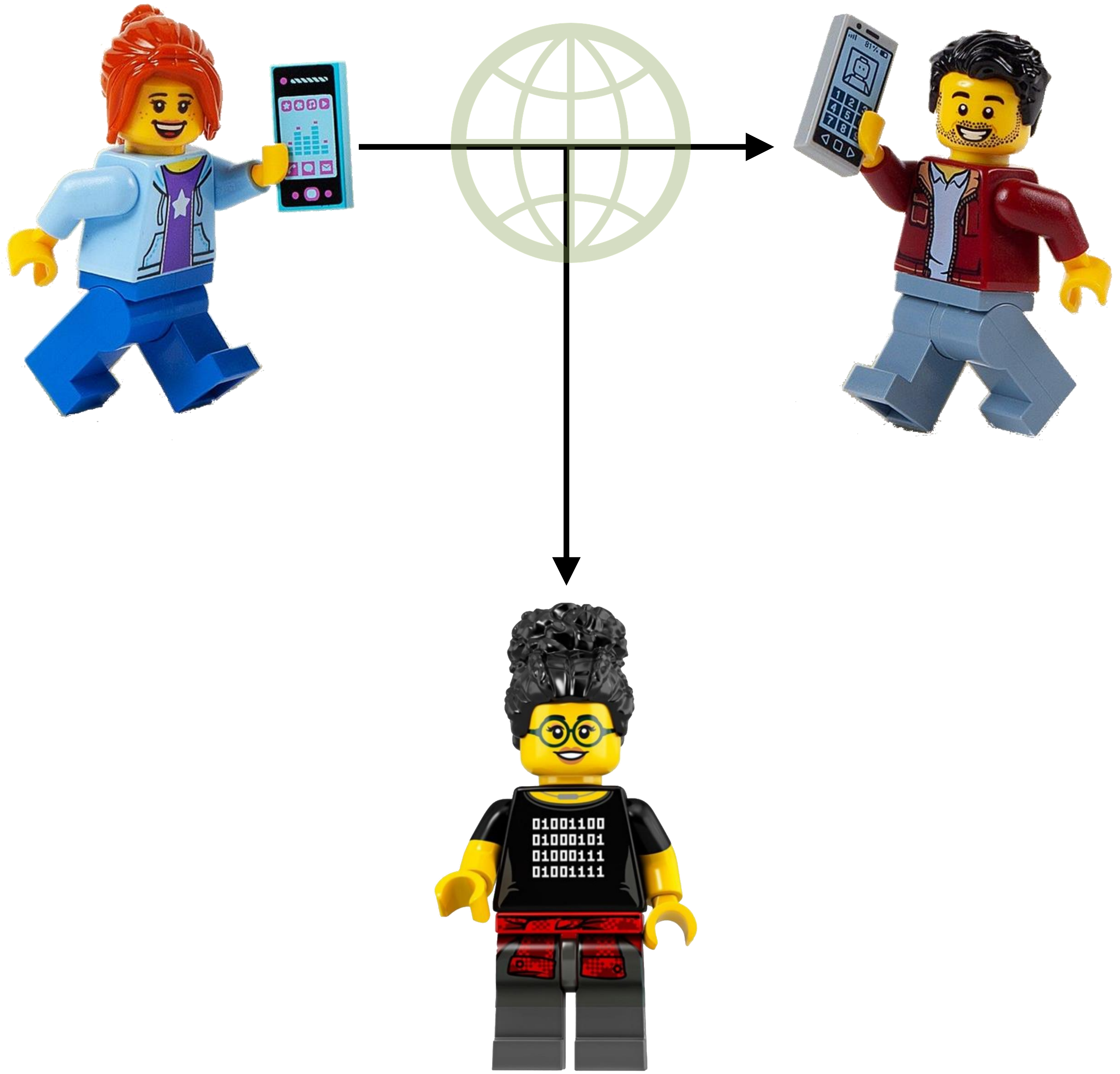
Connection Encrypted (TLS_RSA_WITH_AES_256_CBC_SHA, 256 bit keys, TLS 1.2)

Recall: end-to-end (e2e) data protection

- Alice and Bob want to have a private digital conversation

- They would like to use AuthEnc
  - Provides privacy + authenticity vs. Mallory with full network control
  - Provides partial sender deniability even if Mallory coerces Bob

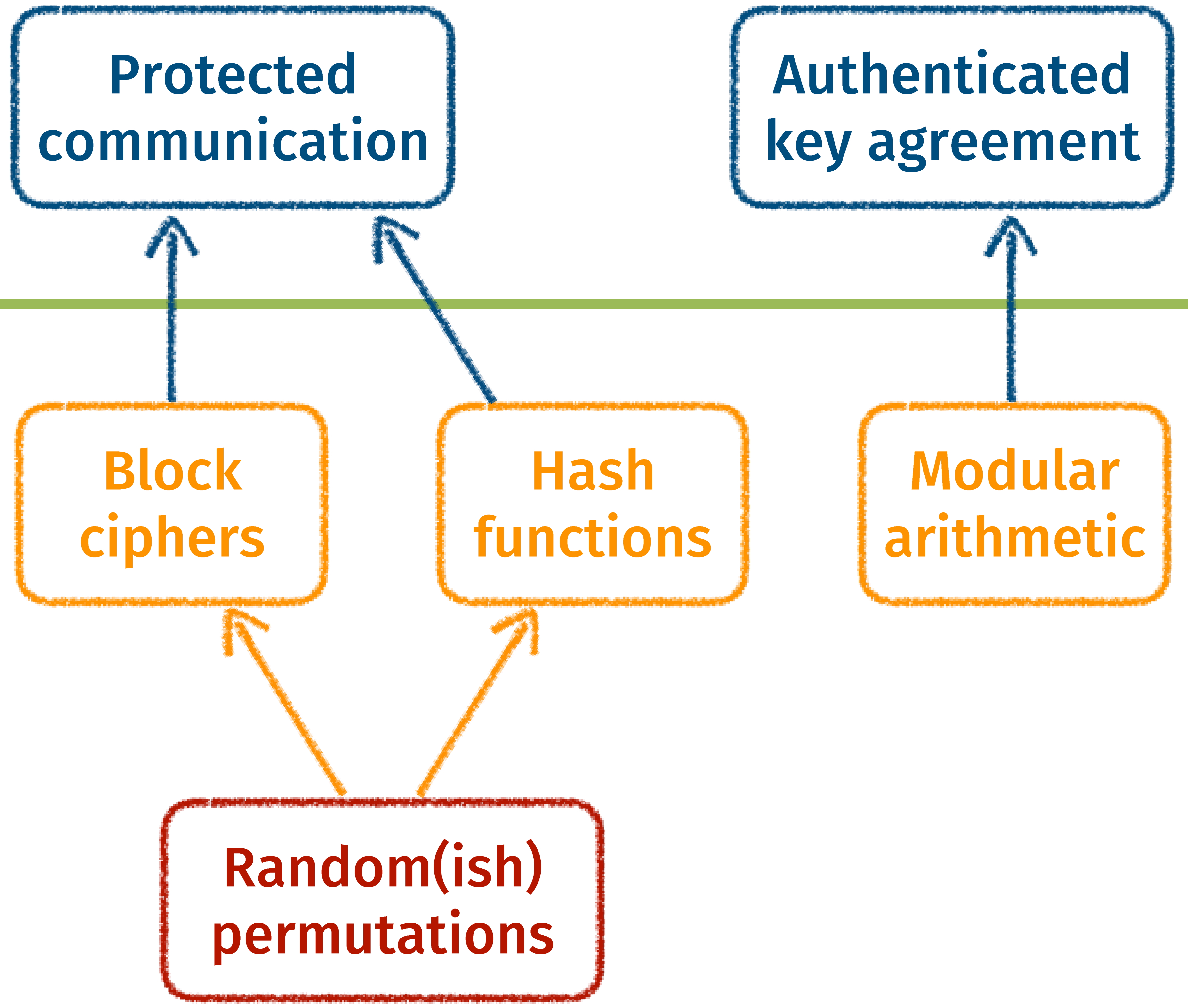
- Remaining issues
  - Alice and Bob don't yet have a shared (*symmetric*) key
  - Need forward + backward secrecy



Course roadmap

Elegant protocols

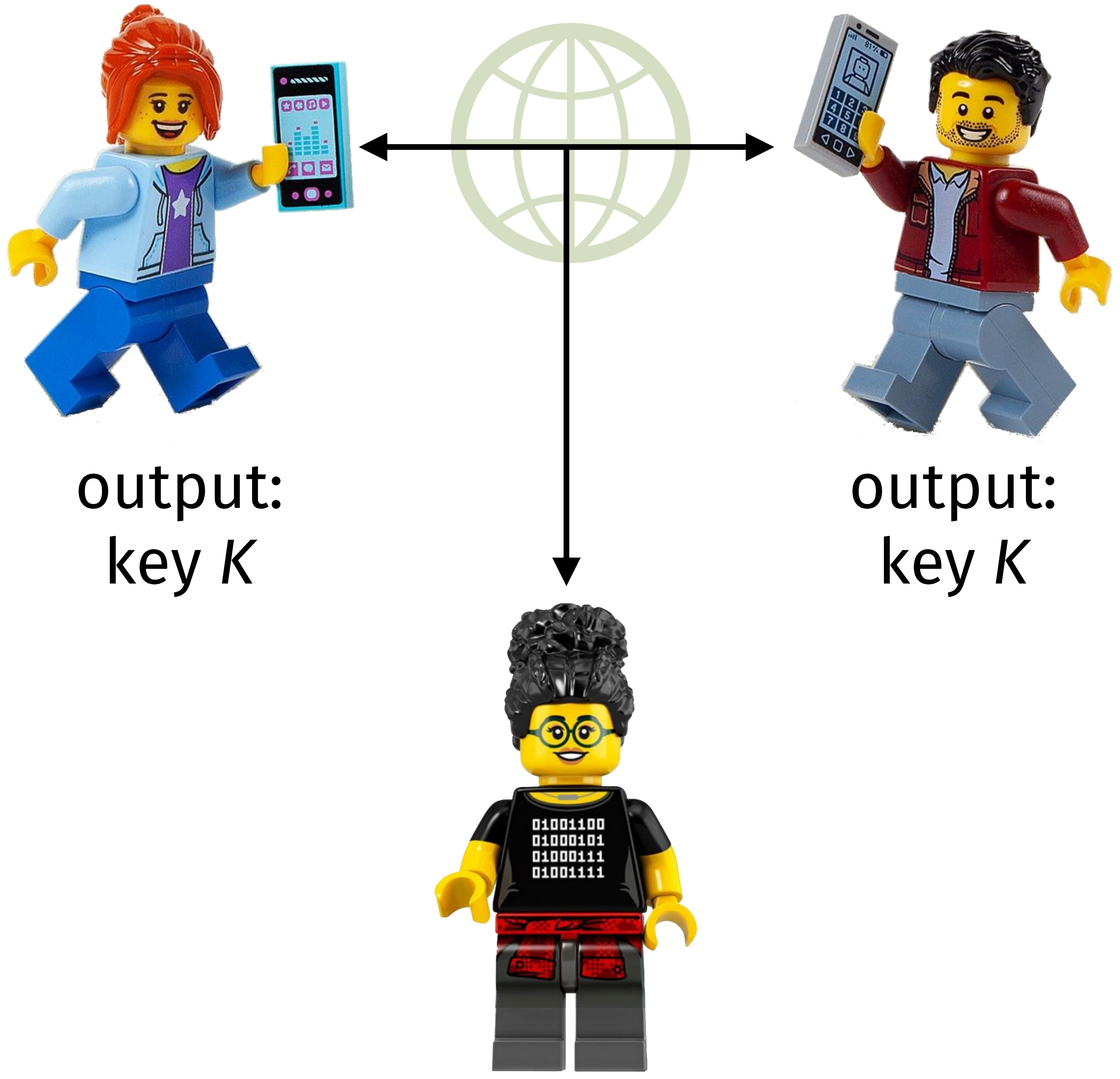
Utilitarian tools



1. Key Exchange

Generating the first shared secret

- Alice and Bob have
  - Never met in person, or else they could exchange a key face-to-face
  - Lack any shared secrets, or else they could run PBKDF2 on them
- They do have individual secrets!
- Question: can Alice and Bob generate a symmetric key *K* and keep it secret from Eve/Mallory?

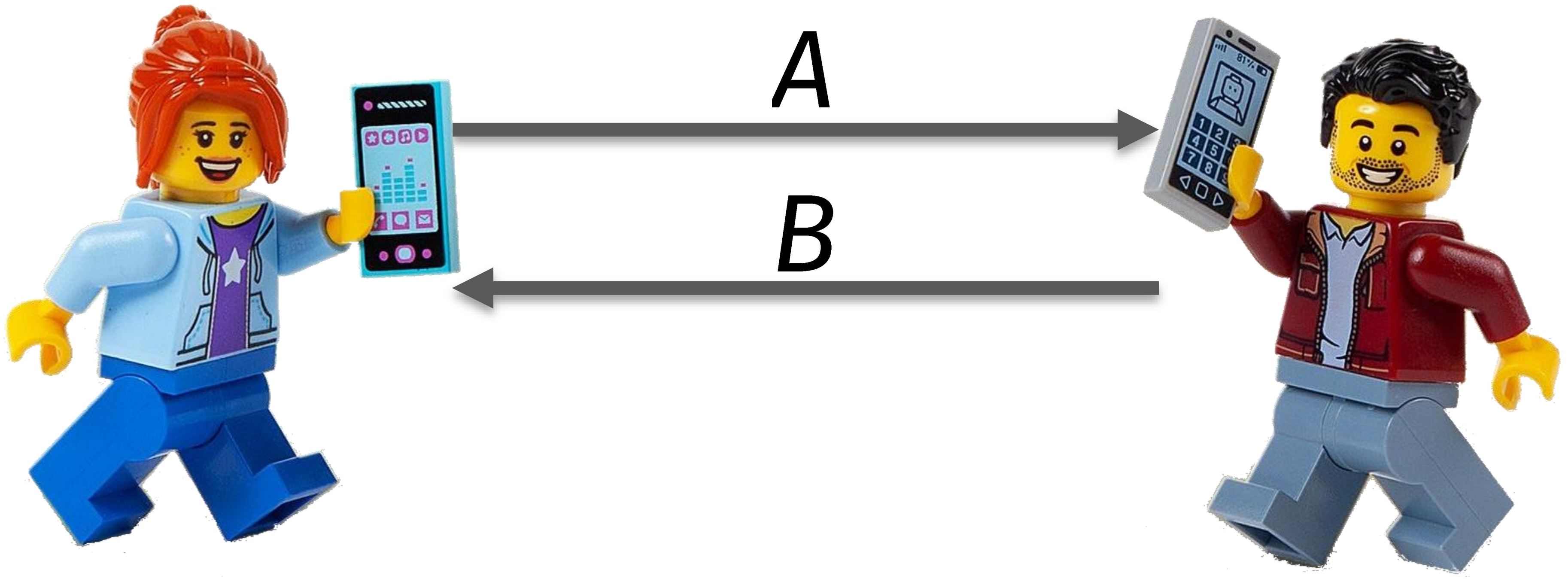


Diffie-Hellman key agreement (vs passive Eve)

Protocol (given a public const g)

Choose a randomly
Compute $A = g^a$

Choose b randomly
Compute $B = g^b$



Output $K = B^a$

Output $K = A^b$

$\text{AuthEnc}_K(\text{P})$

Delete a, K

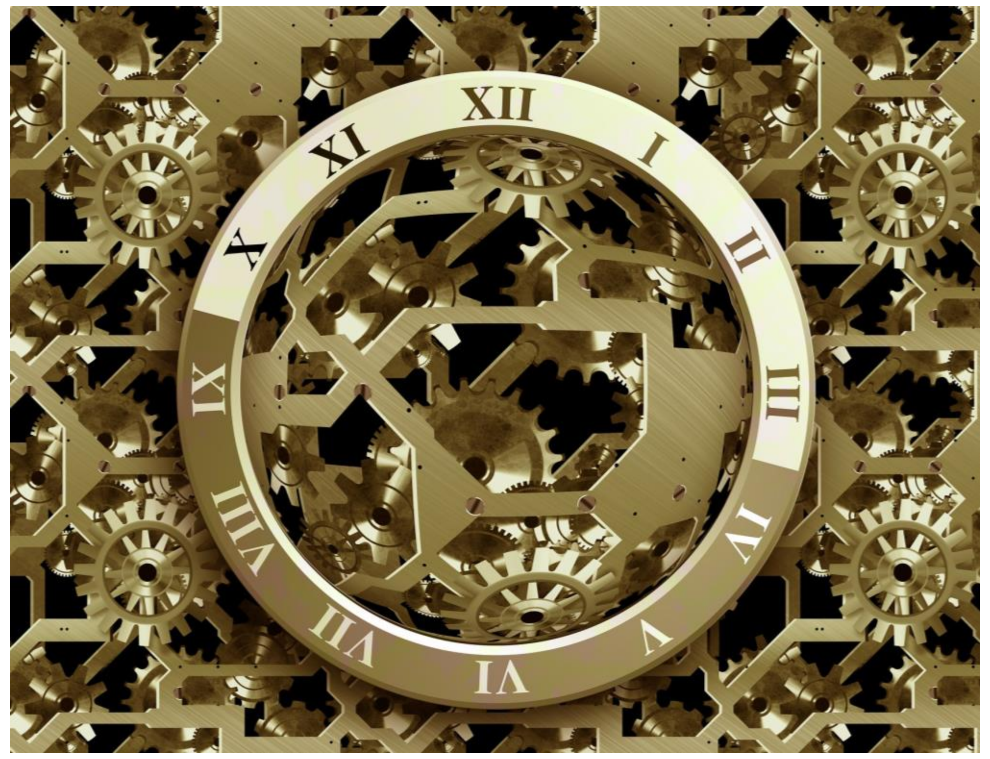
Delete b, K

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



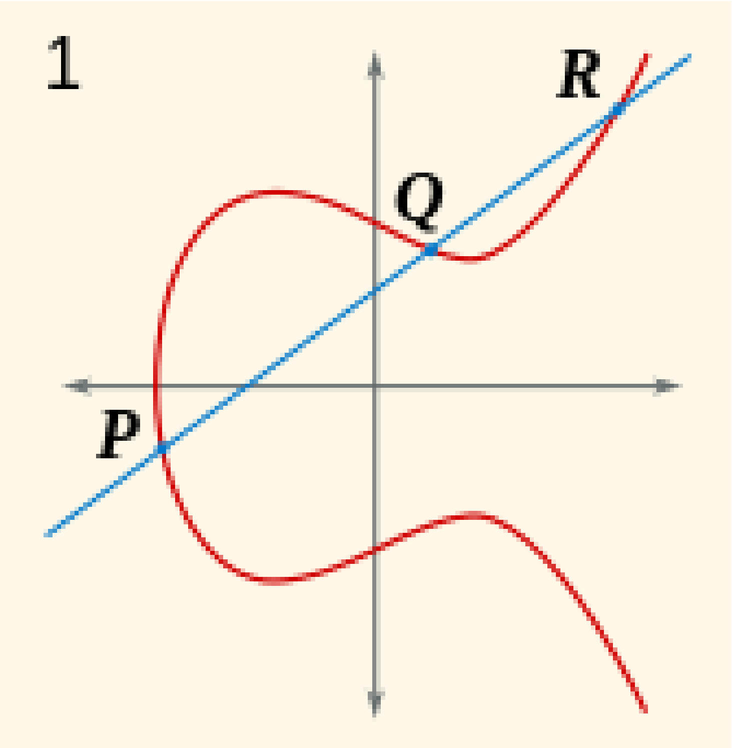
- Raise a constant to any power, e.g. $x \mapsto 3^x \pmod 7$

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

- Permutation, but hard* to invert

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

Elliptic curves



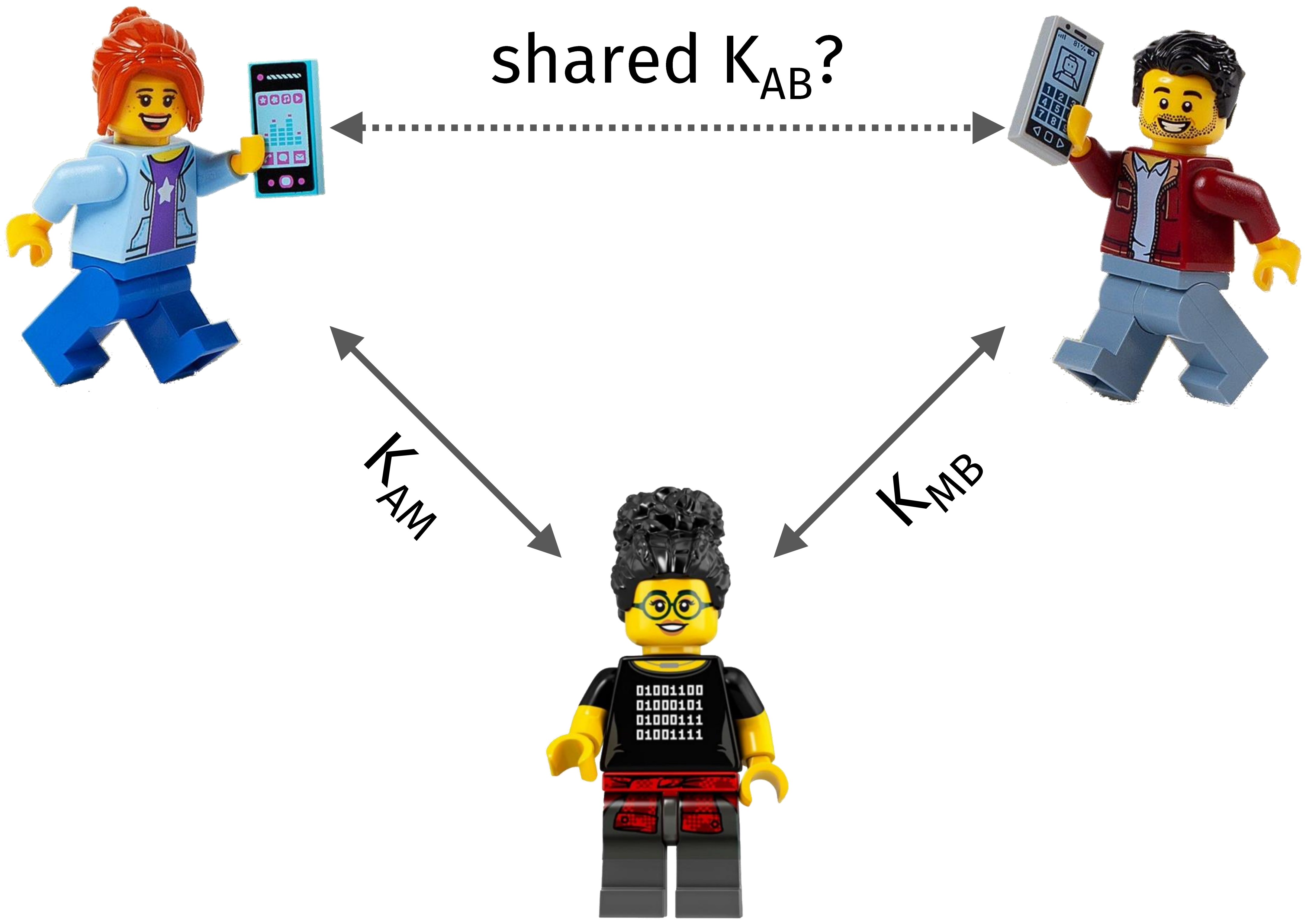
- Elliptic curve: a cubic equation $y^2 = x^3 + ax + b \pmod p$
- Consider set of points on this curve
- We can "multiply" points using the rule $P \cdot Q \cdot R = 1$

Technical Details

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

Diffie-Hellman key agreement (vs active Mallory)

Active attacker causes problems!



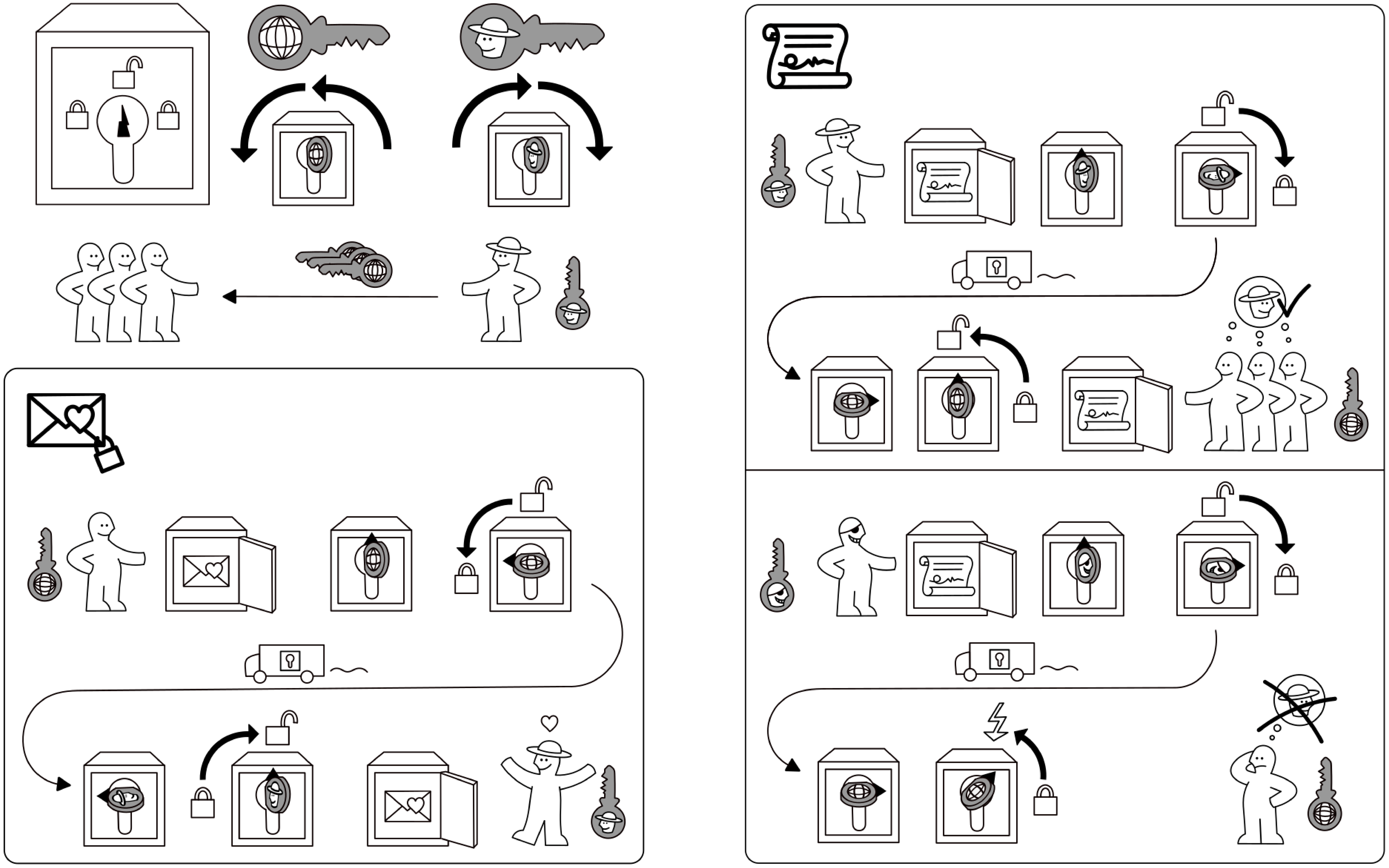
- Q: How do Alice and Bob verify they're talking with each other?
- A: Use a MAC?

2. Public Key Cryptography

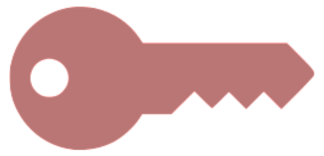
PUBLIC KEY KRÜPTO

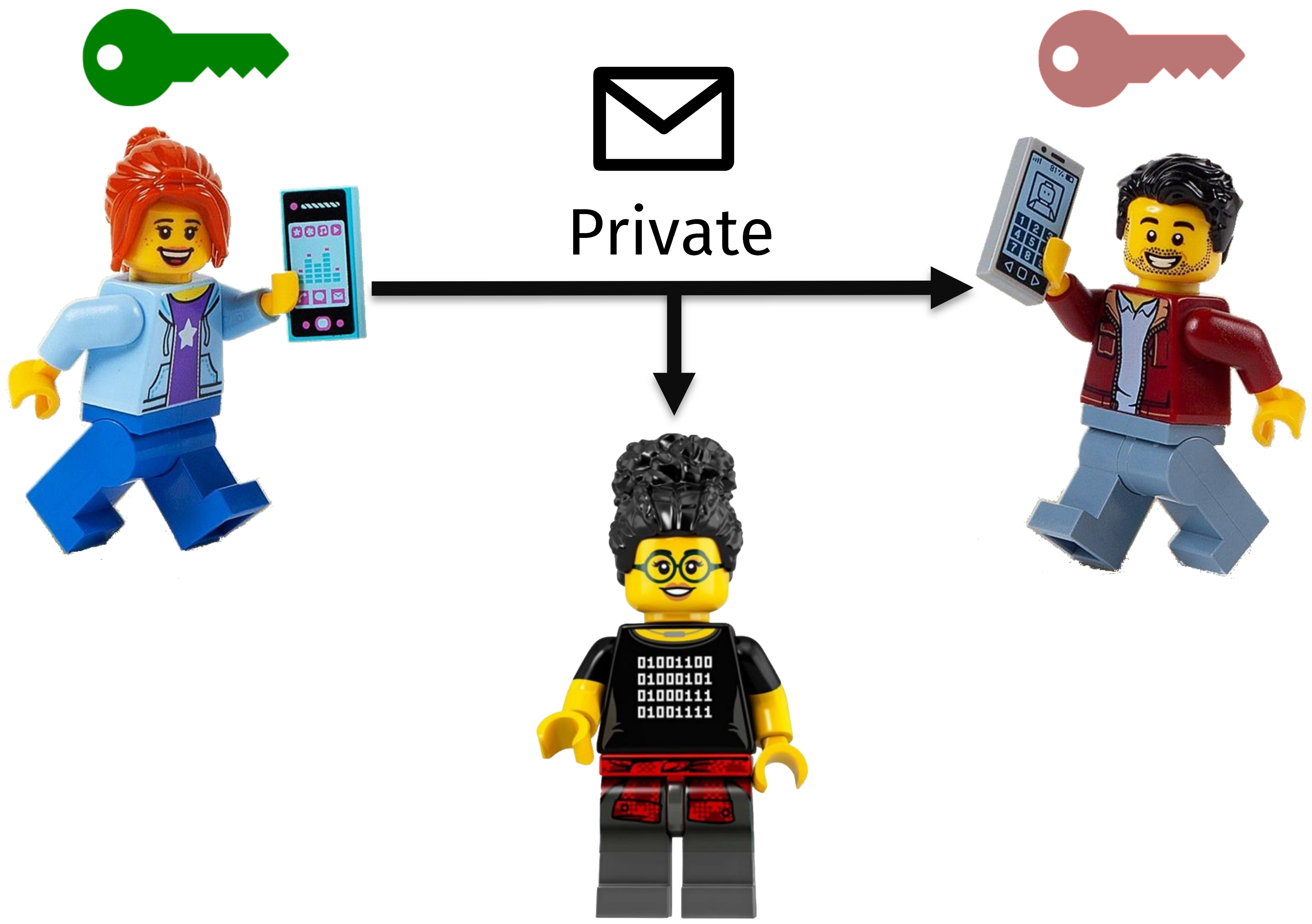
idea-instructions.com/public-key/

IDEA

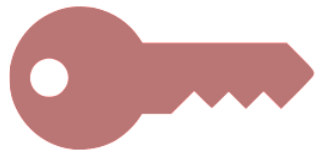


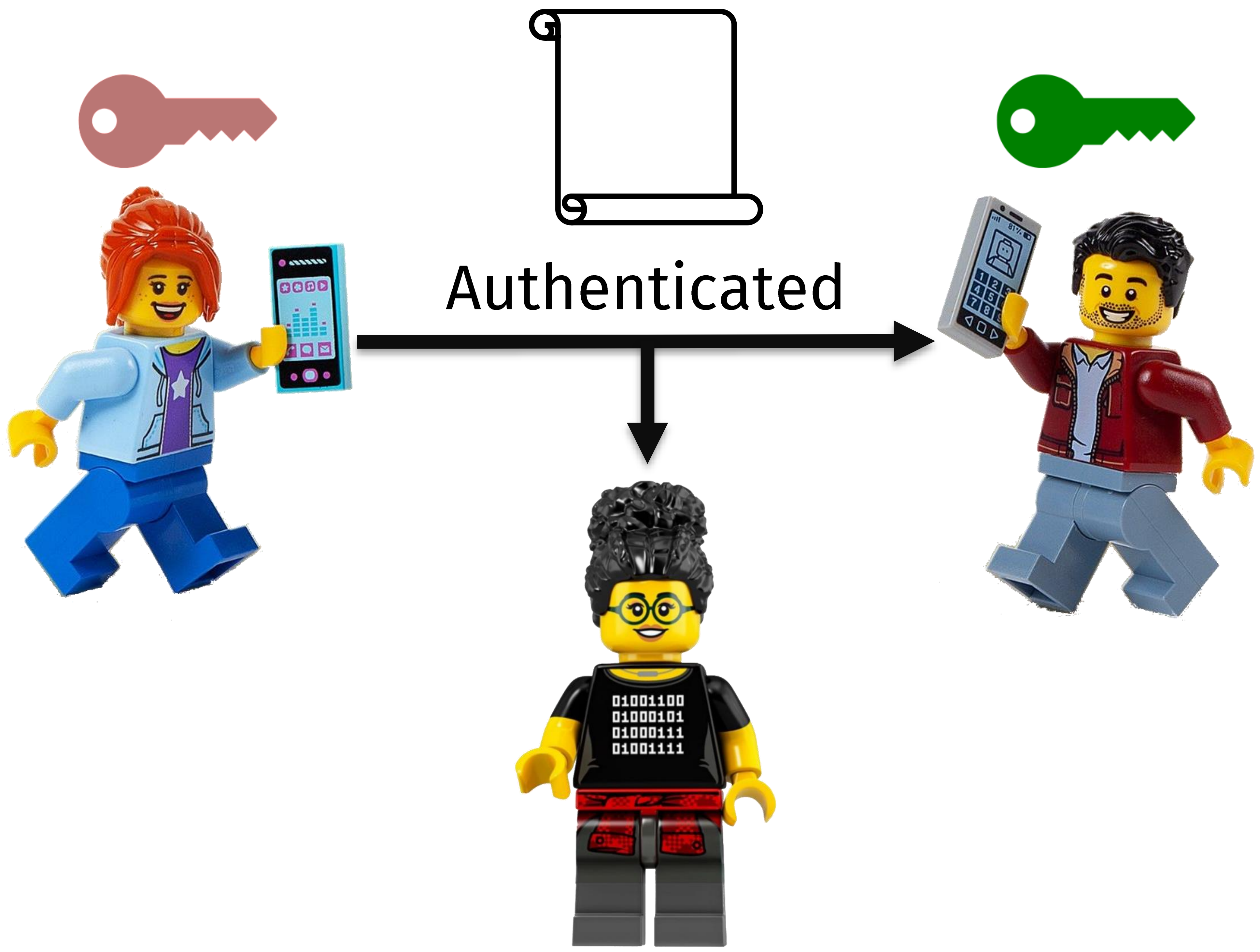
Public key encryption

- Operation
  - Anybody can send ciphertexts
  - Only Bob can decrypt + read
  - Security guarantee: CPA or CCA
- Impact if Mallory learns  ?
  - Problem: Eve reads msgs from *past*
  - Response: ??? (dangerous to use!)
- If necessary, use IES or KEM



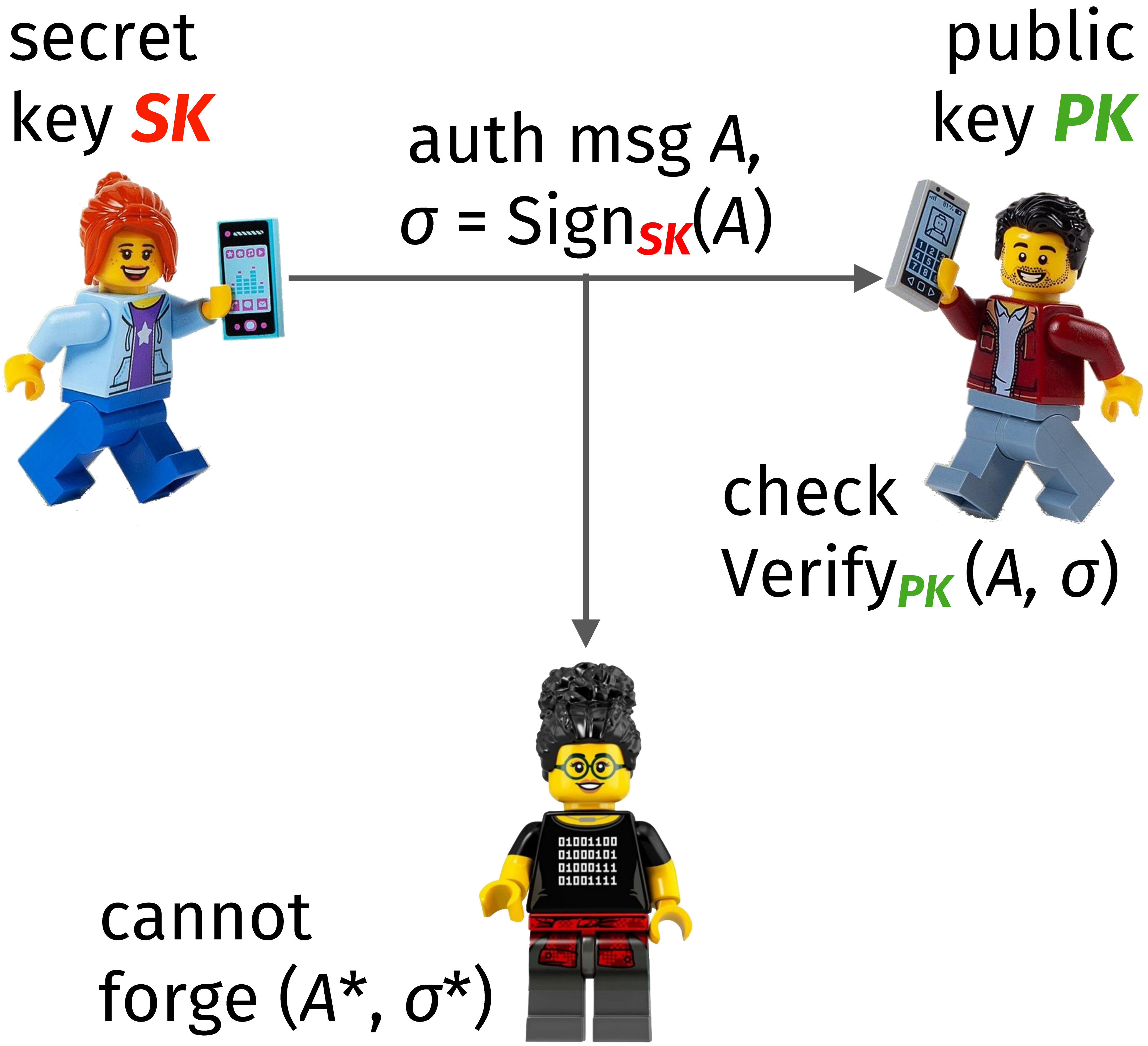
Public key digital signatures

- Operation
  - Only Alice can generate signatures
  - Anybody can verify
  - Security guarantee: EU-CMA
- Impact if Mallory learns ?
  - Problem: Eve forges msgs in *future*
  - Response: Alice can *revoke* her key

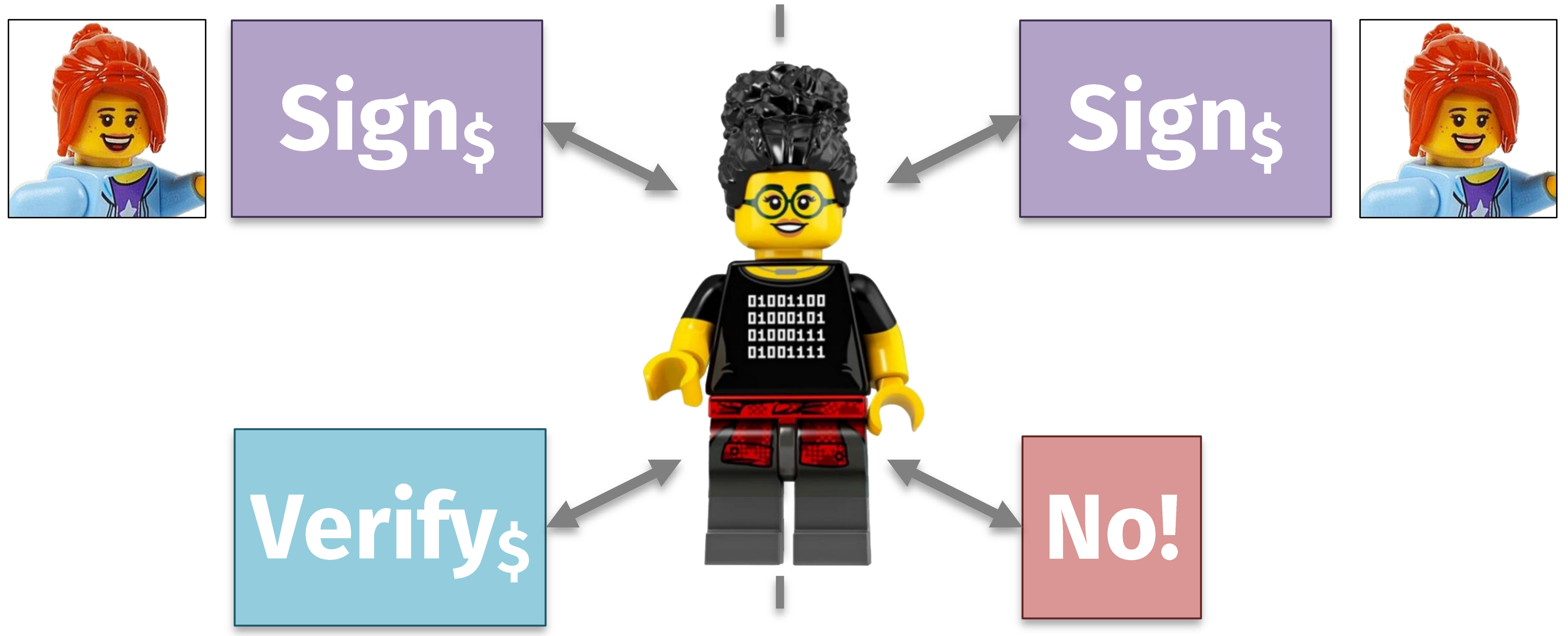


3. Public Key Digital Signatures

Digital signatures provide *public* authentication



Security: EU-CMA game (like MACs)

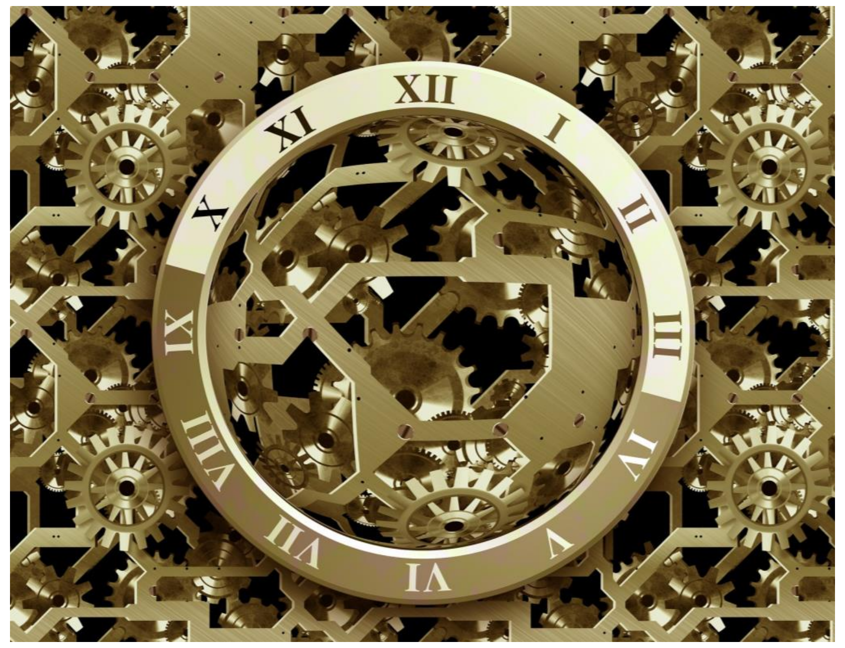


Property	MAC	Sign
Sender auth: Bob knows Alice sent *A*	✓	✓
Msg auth: Bob can detect tampering	✓	✓
Receiver auth: Bob knows *A* for him	✓	✗
Partial deniability: Alice can deny *A*	✓	✗

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

Technical Details

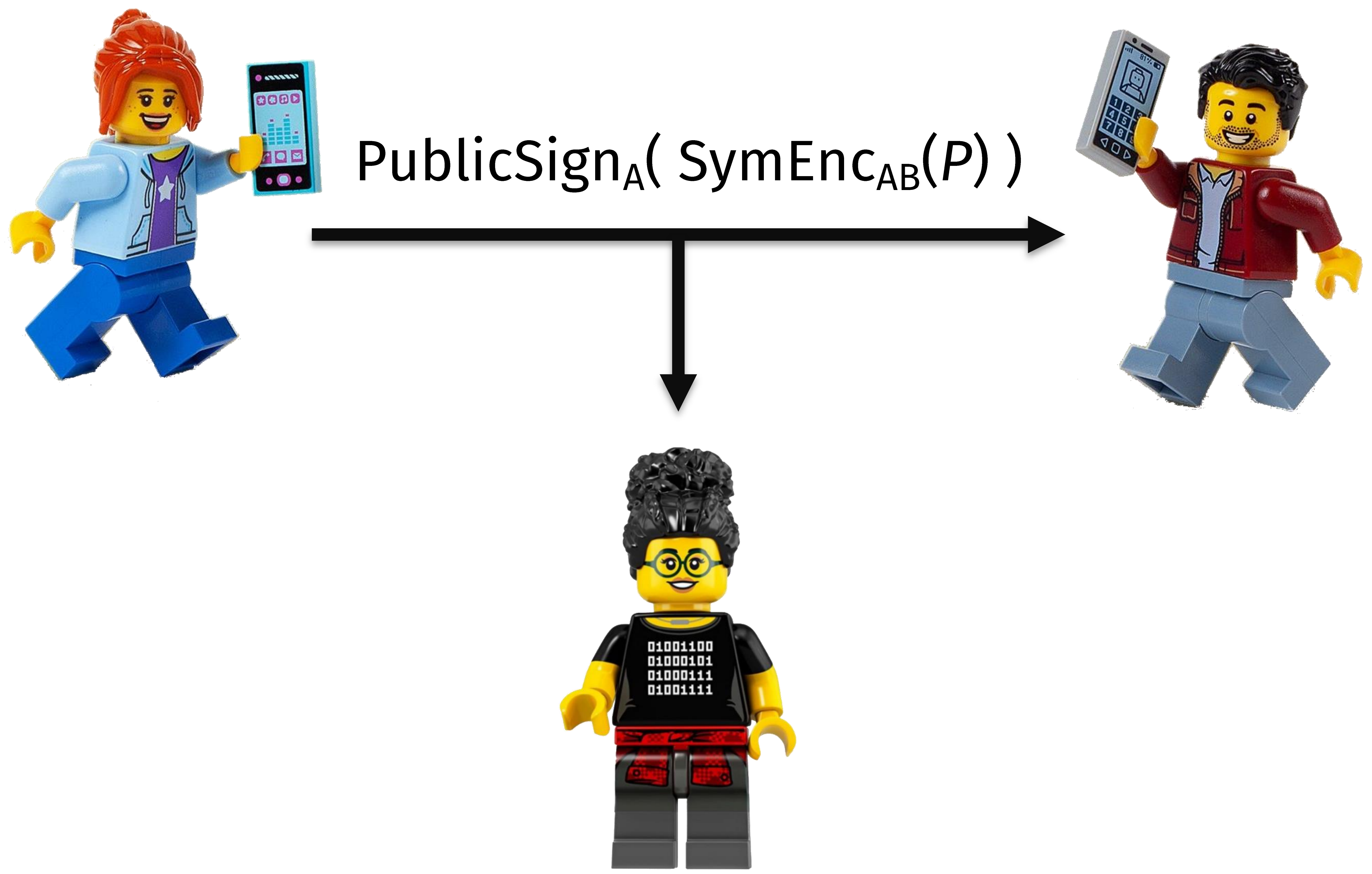
Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)

Technical Details

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

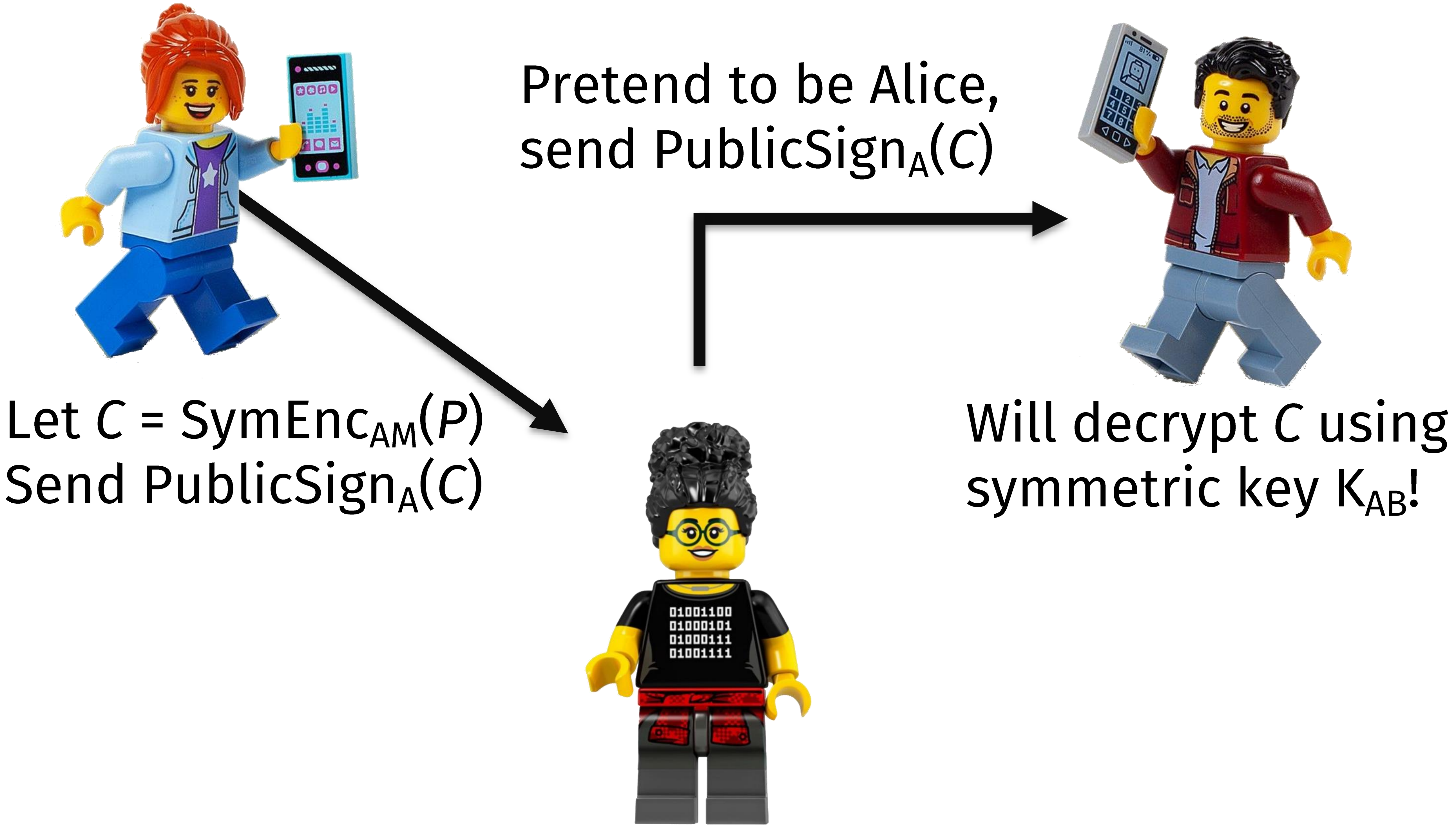
Combining symmetric encryption + public signatures

- In the symmetric case, we learned that Enc-then-MAC is best option
  - Intuition: Never expose the decryption key to an invalid message
- Does this technique work as well with public key signatures?



Combining symmetric encryption + public signatures

- In the symmetric case, we learned that Enc-then-MAC is best option
  - Intuition: Never expose the decryption key to an invalid message
- Does this technique work as well with public key signatures?
- Answer: No!
  - Issue: Mallory can receive ciphertexts from Alice, claim them as her own!
- Can lead to an oracle attack, as occurs with Apple's iMessage



Authenticated key exchange

Choose a randomly
Compute $A = g^a$

Choose b randomly
Compute $B = g^b$

A, $\text{sign}_{\text{skA}}(A)$ →

← B, $\text{sign}_{\text{skB}}(B)$

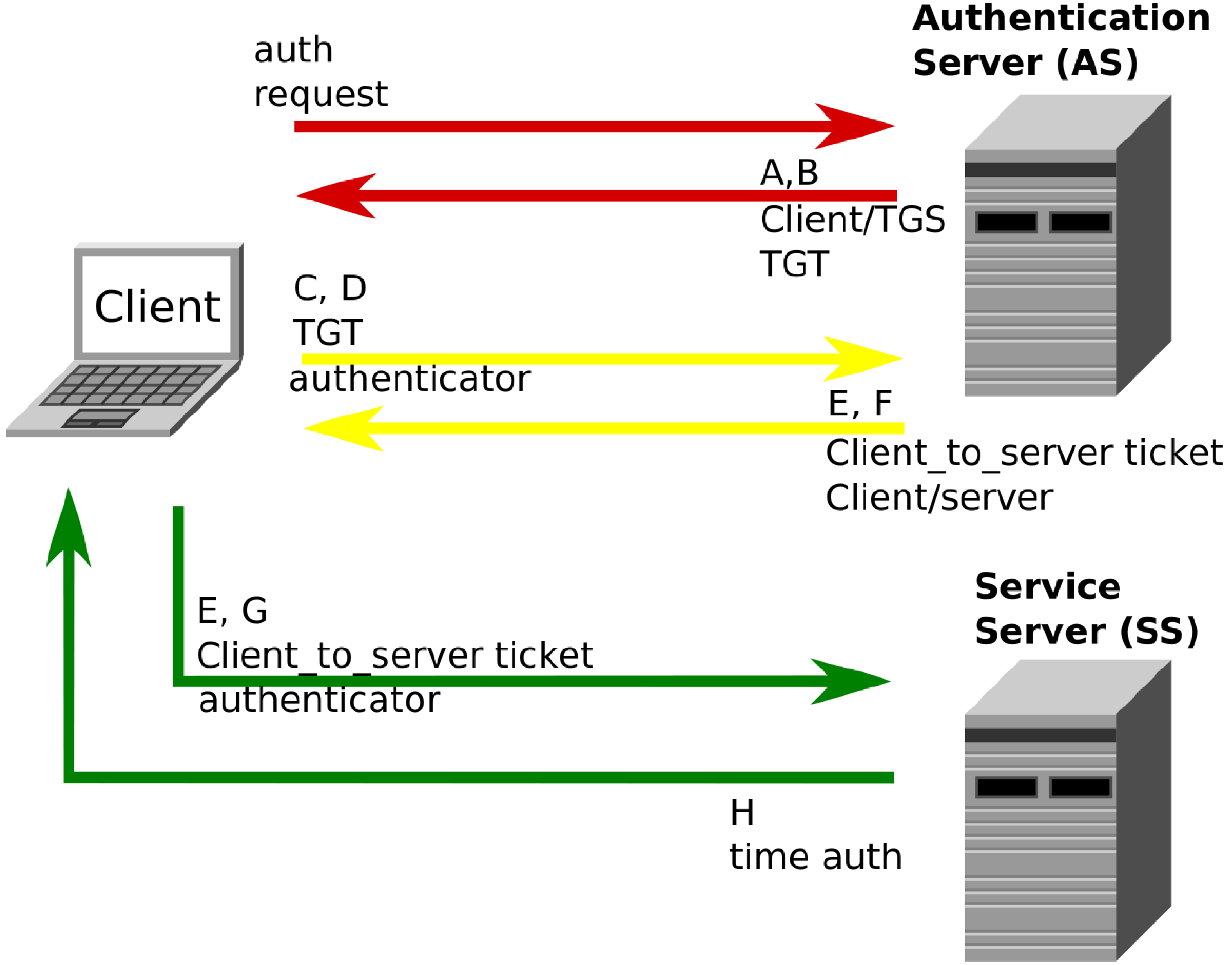
1. Alice and Bob sign their messages during Diffie-Hellman key exchange
2. Alice and Bob verify signature of each other's messages
3. Use shared key $A^b = B^a$ for (deniable) symmetric authenticated encryption

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

4. Digital Certificates & the Public Key Infrastructure (PKI)

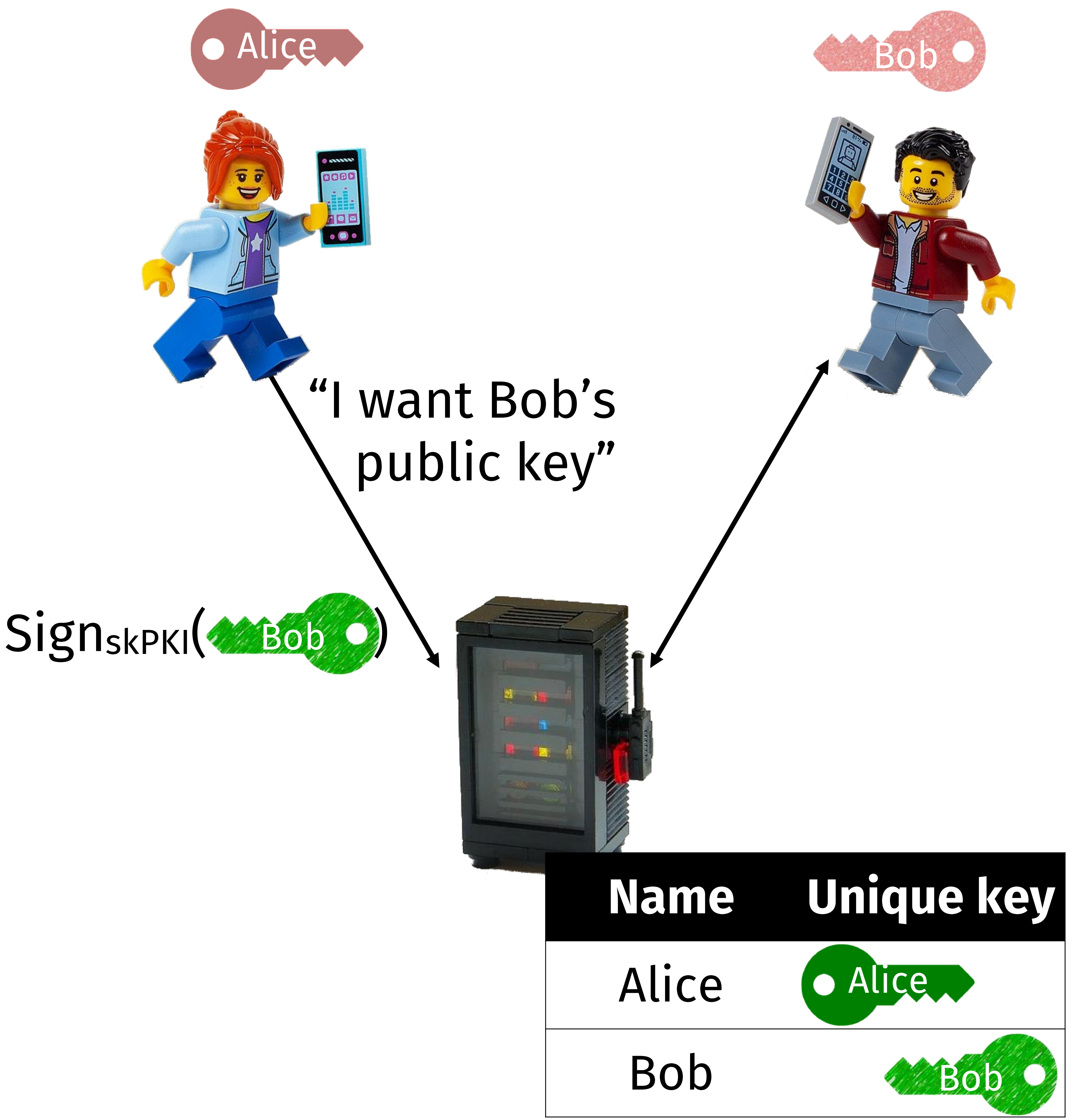
One option: ask a common friend to connect you

- Suppose that
  - Alice + Bob both trust server *S*
  - Alice + Bob have shared symmetric keys K_{AS}, K_{BS} with the server
- Then, server can create key K_{AB} and send it to them
  - Needham-Schoeder (1978)
  - Kerberos (late 1980s)
- Mostly used within a single enterprise, since server sees all



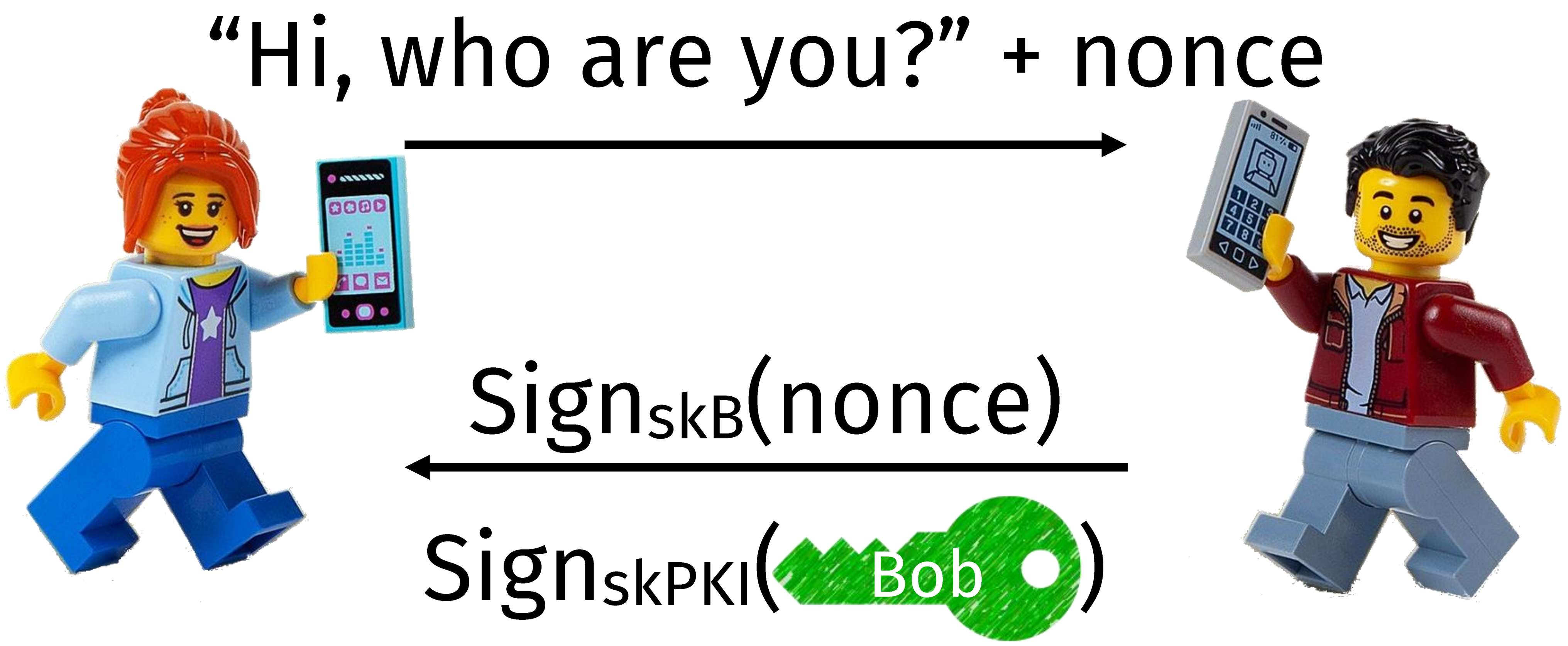
Public key infrastructure

- A *certificate authority* stores all public keys (like a phone book)
    - Server does *not* learn private keys
- Anyone can query the authority to learn someone else's key
- CA signs responses so that everybody knows they are legit
- Alice knows the CA's public key because it is included in her OS



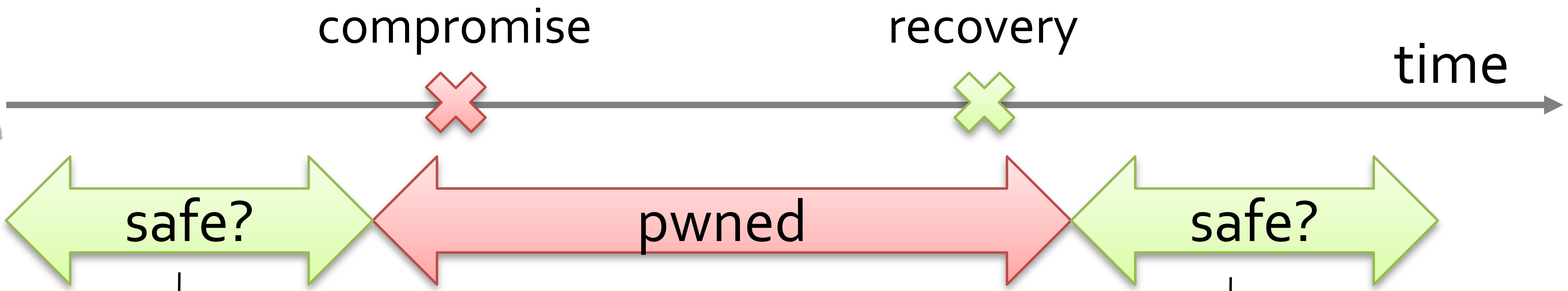
PKI improved

- Alice talks with Bob, not CA
- Bob adds the CA's attestation that signing key belongs to him
- (Shown: simplified version of the TLS handshake)



Name	Unique key
Alice	Alice
Bob	Bob

What if Bob's secret key *SK* is compromised?

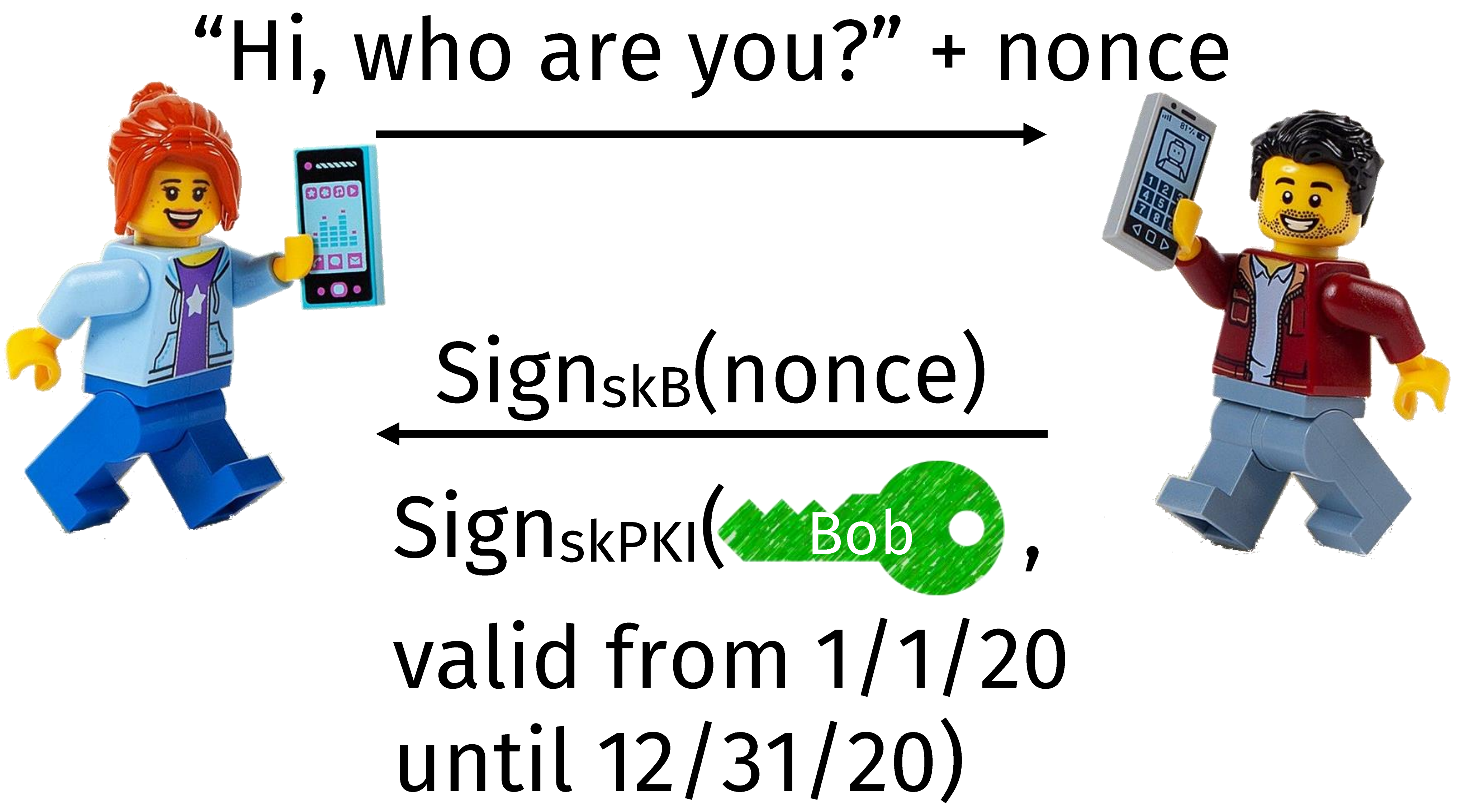


<u>Forward (pre-compromise) secrecy</u>
Yes! Unless Mallory has a time machine, signatures Alice verified before a breach must be valid.

<u>Backward (post-recovery) secrecy</u>
No. If Mallory has Bob's secret key, she can sign messages and Alice will believe they are from Bob.

Backward security technique #1: Cert expiration

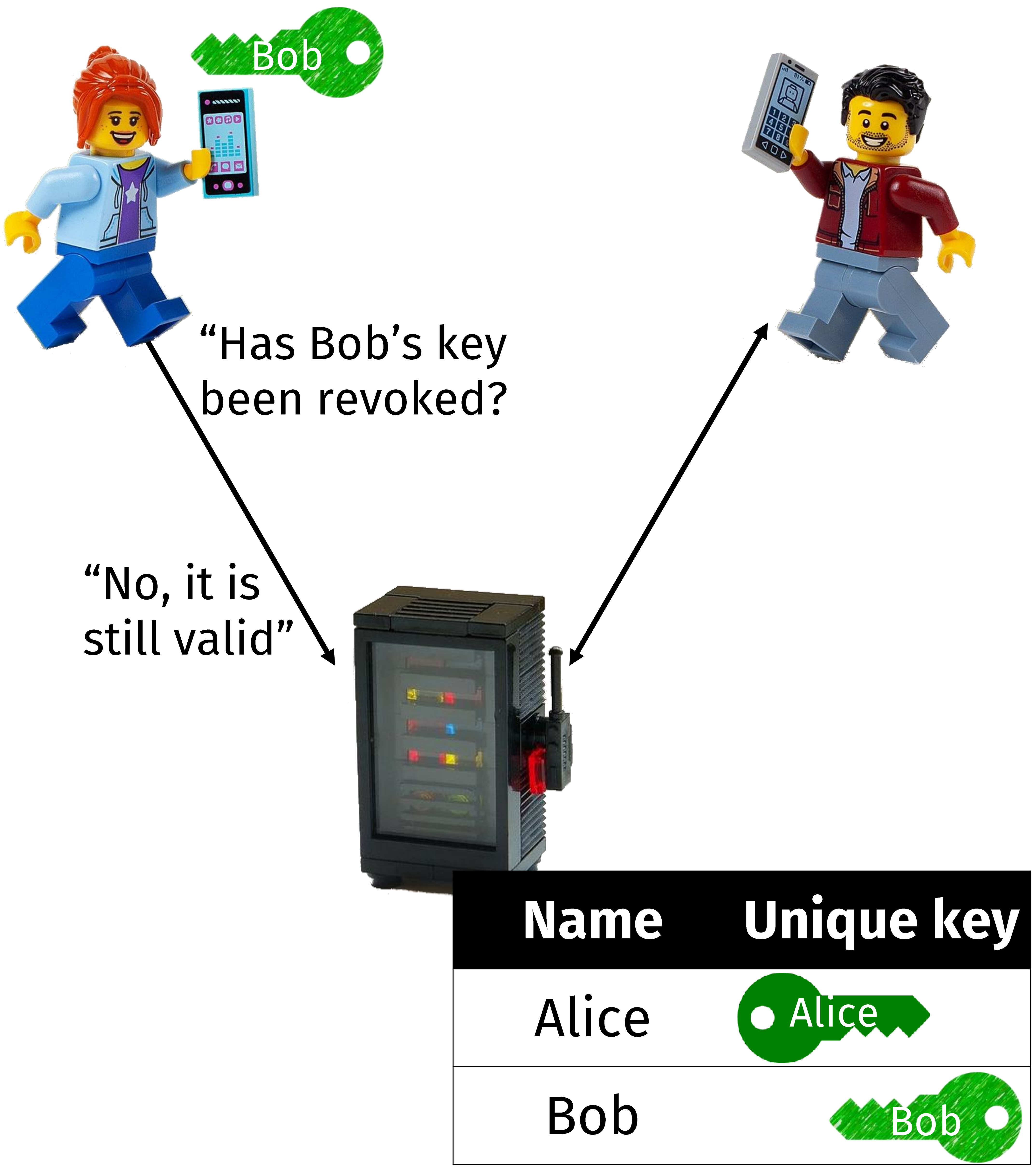
- Alice should only trust Bob's key for a limited time
- The CA's attestation includes this time range
- Afterward, Bob must register a new public key



Name	Unique key
Alice	Alice
Bob	Bob

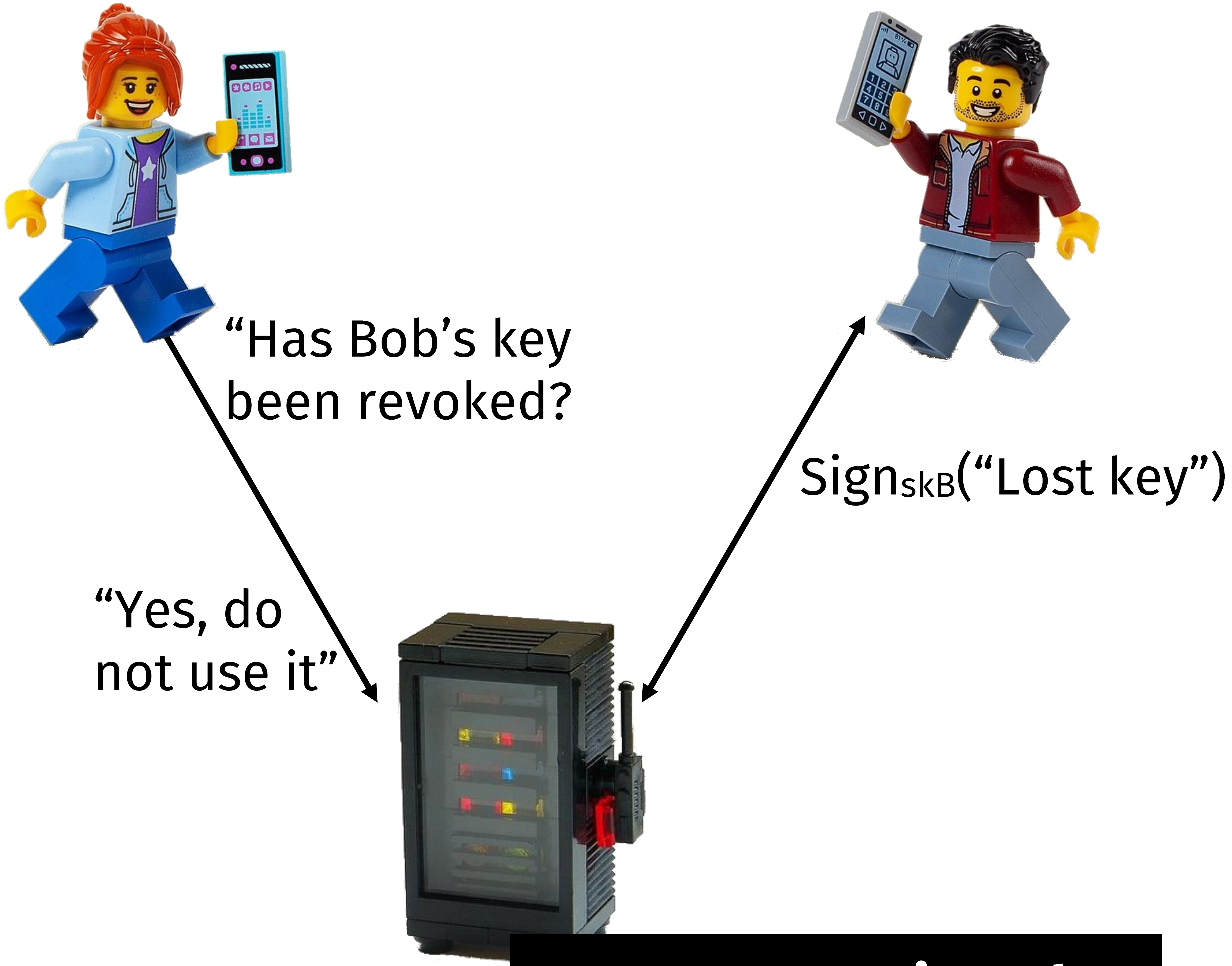
Backward security technique #2: Key revocation

- CA binds public key to a name
- If you lose control of your public key, you should tell the CA to break this binding
- Every CA maintains a certificate revocation list that anyone can query



Backward security technique #2: Key revocation

- CA binds public key to a name
- If you lose control of your public key, you should tell the CA to break this binding
- Every CA maintains a certificate revocation list that anyone can query



Name	Unique key
Alice	Alice