Public Key Infrastructure

An Overview of Asymmetric Key Encryption

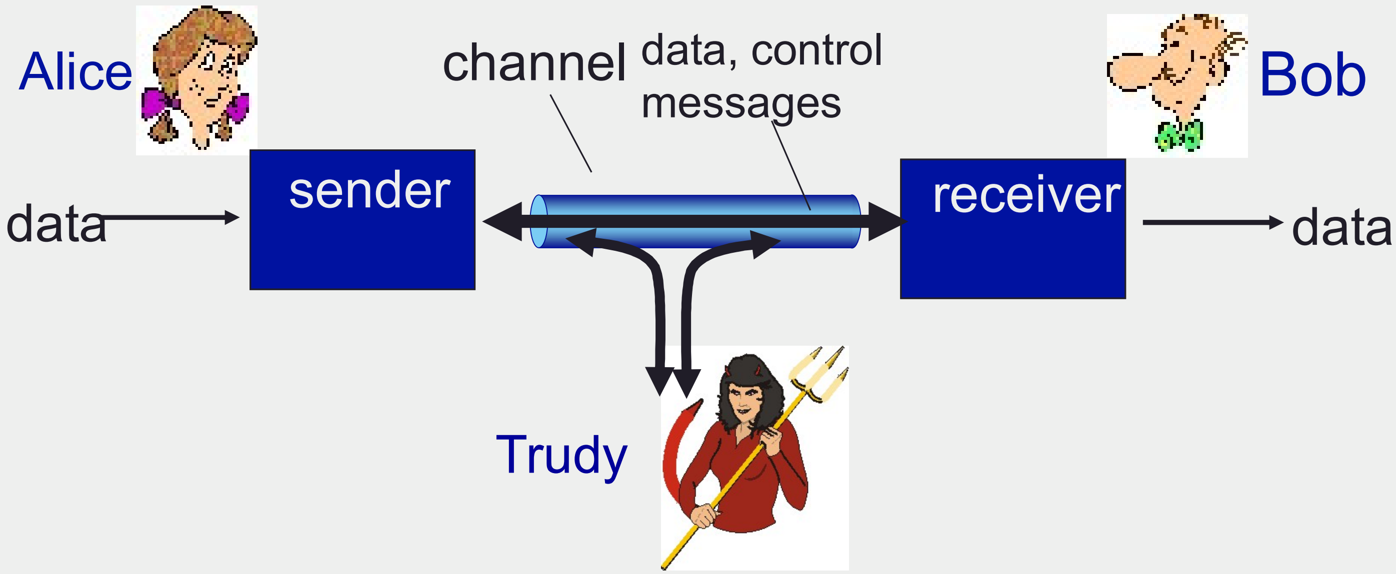
R. Venkateswaran

There are bad guys (and girls) out there!

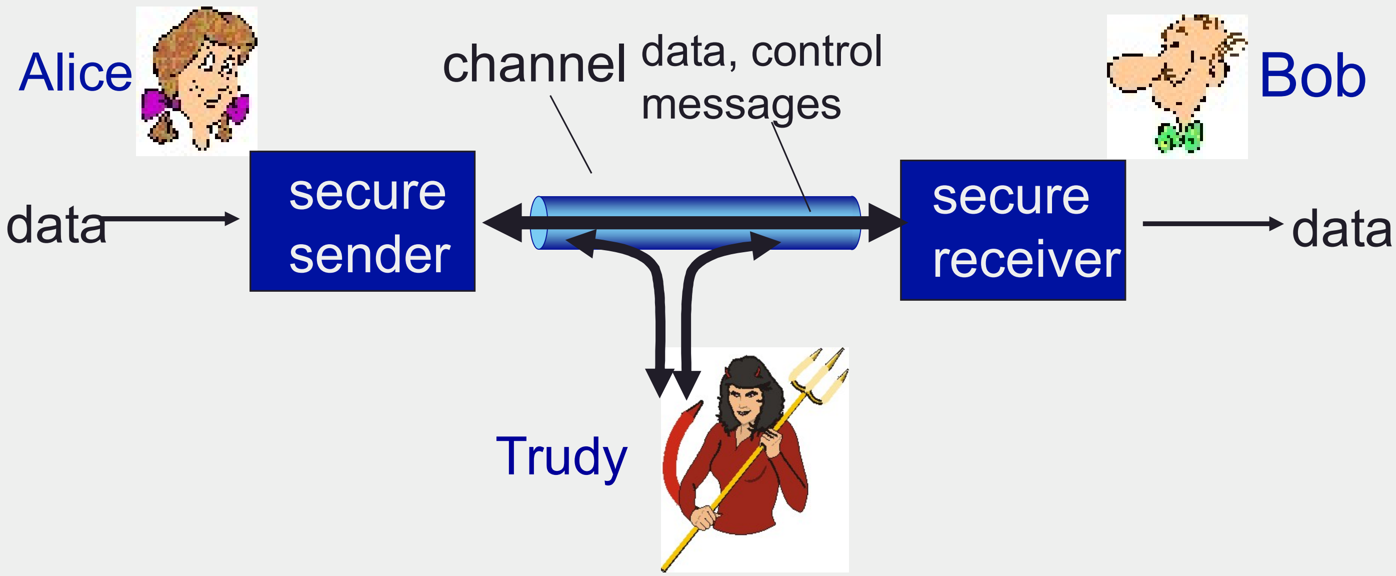
- Q: What can a “bad guy” do?
- A: A lot!
  - eavesdrop: intercept messages
  - actively insert messages into connection
  - impersonation: can fake (spoof) source address in packet (or any field in packet)
  - hijacking: “take over” ongoing connection by removing sender or receiver, inserting himself in place
  - denial of service: prevent service from being used by others (e.g., by overloading resources)

Friends and enemies: Alice, Bob, Trudy

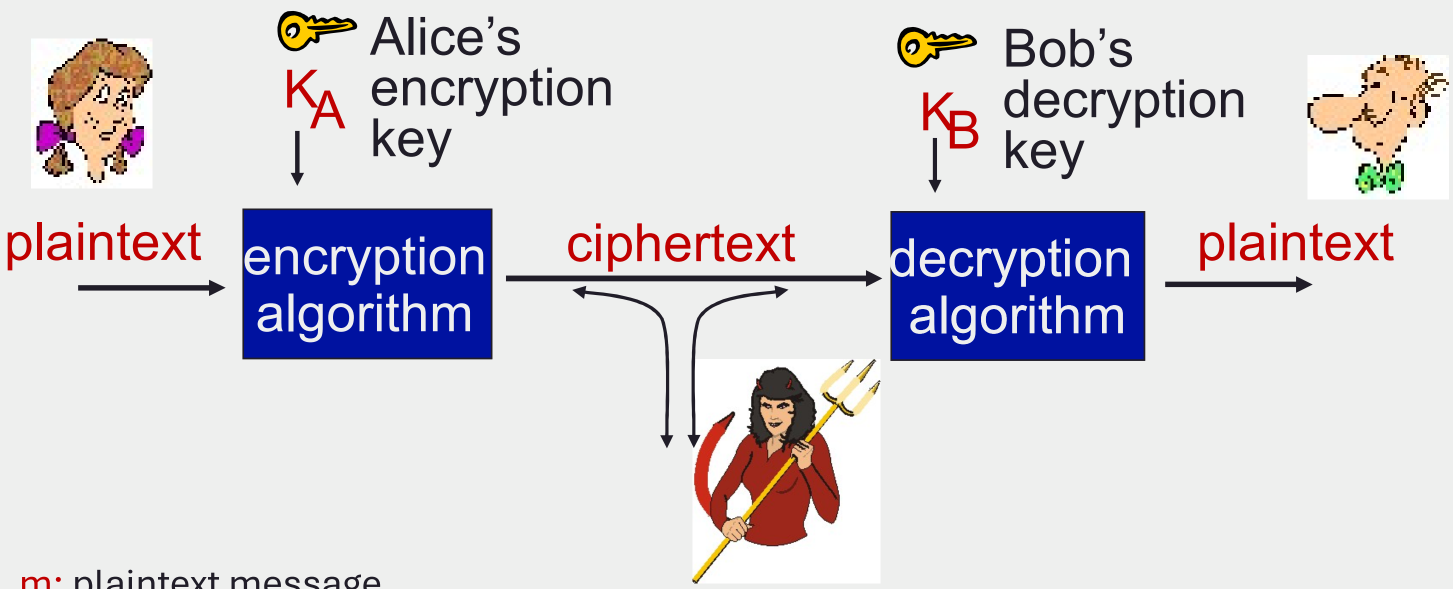
- well-known in network security world
- Alice & Bob want to communicate with each other
- Trudy (intruder) may intercept, delete, add messages



Alice and Bob want to communicate **securely**



The language of cryptography

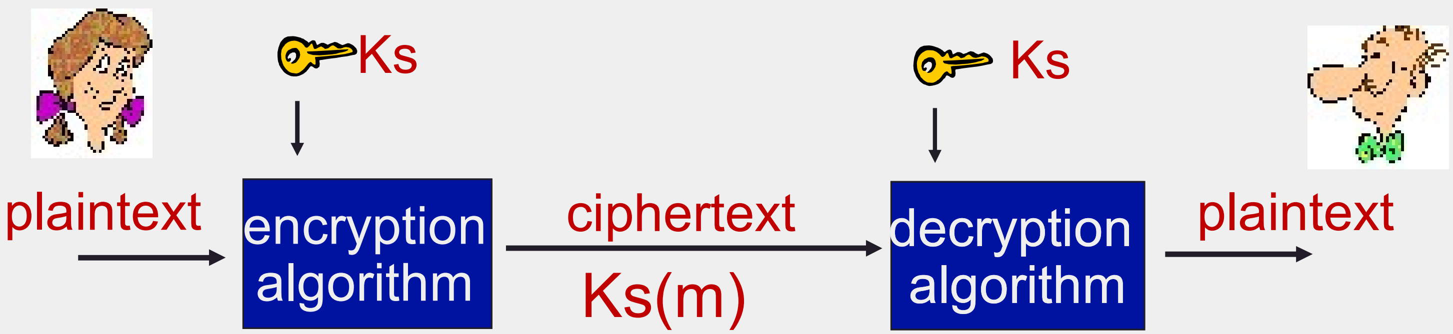


m: plaintext message

$K_A(m)$: ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Symmetric key cryptography

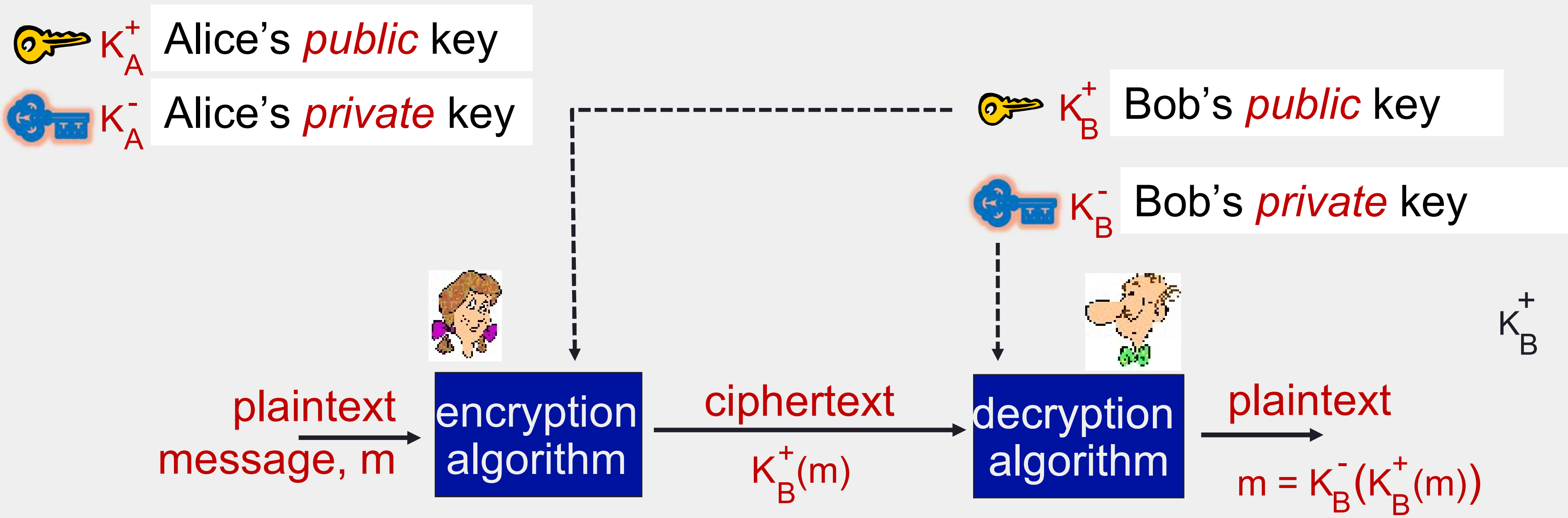


symmetric key crypto: Bob and Alice share same (symmetric) key Ks

Ks satisfies the following property : Ks(Ks(m)) = m

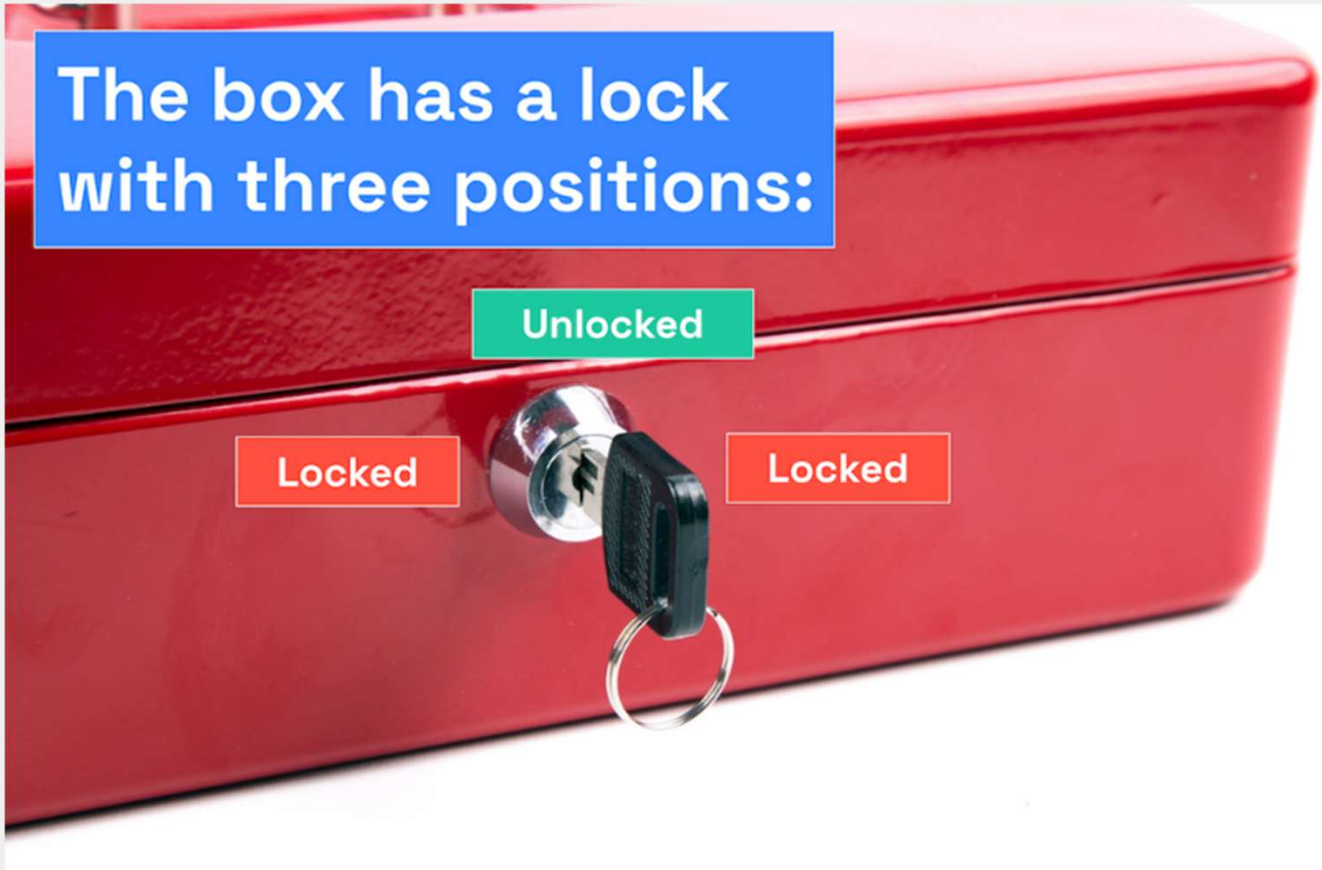
Challenge : How to ensure that both Alice and Bob have the same key

Public Key Cryptography



*P*ublic key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

Public Key Infrastructure - Simple Analogy

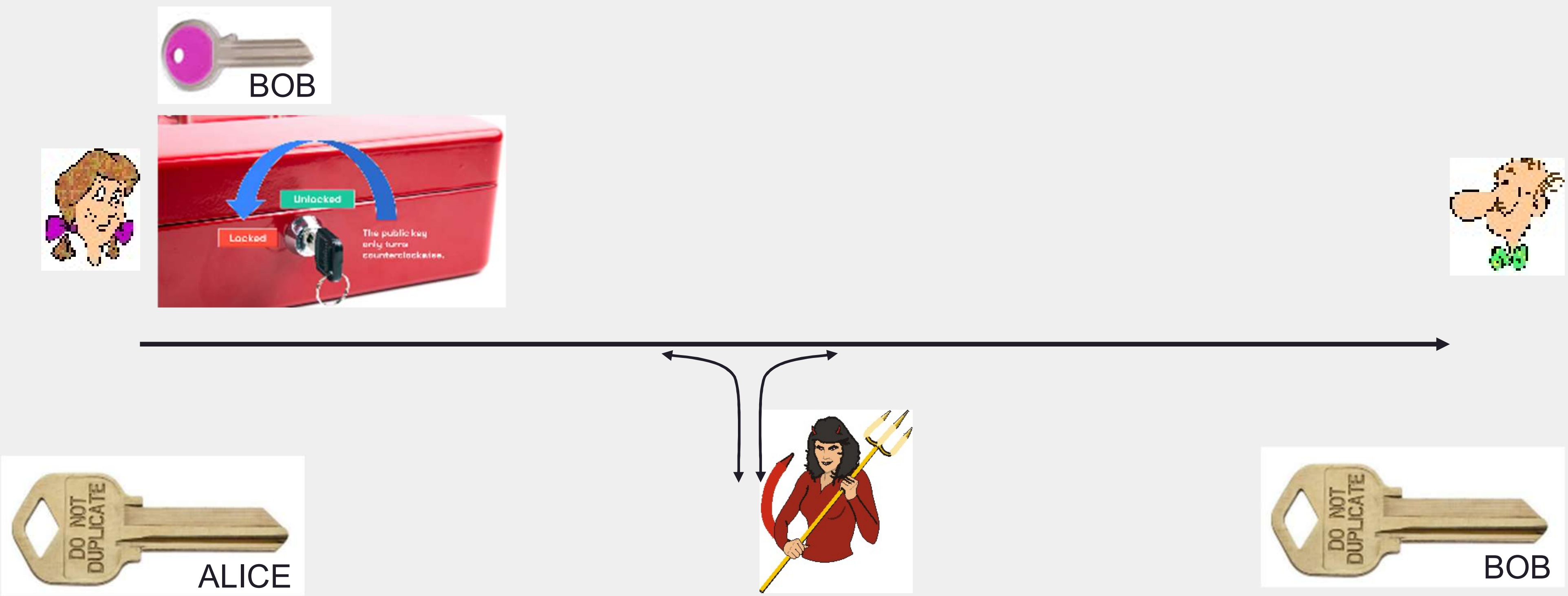


Golden Key – ONLY ONE COPY.
Turns **CLOCKWISE**


Regular Key – Many Copies.
Turns ANTI-**CLOCKWISE**

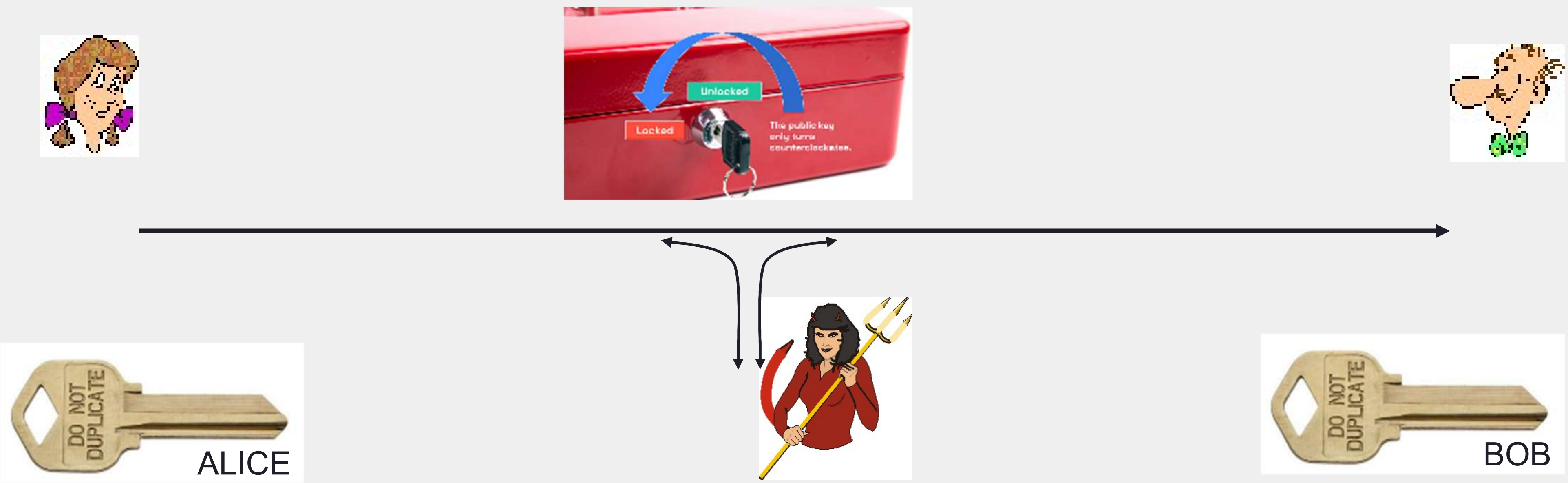

Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Alice sends a message to Bob **securely**



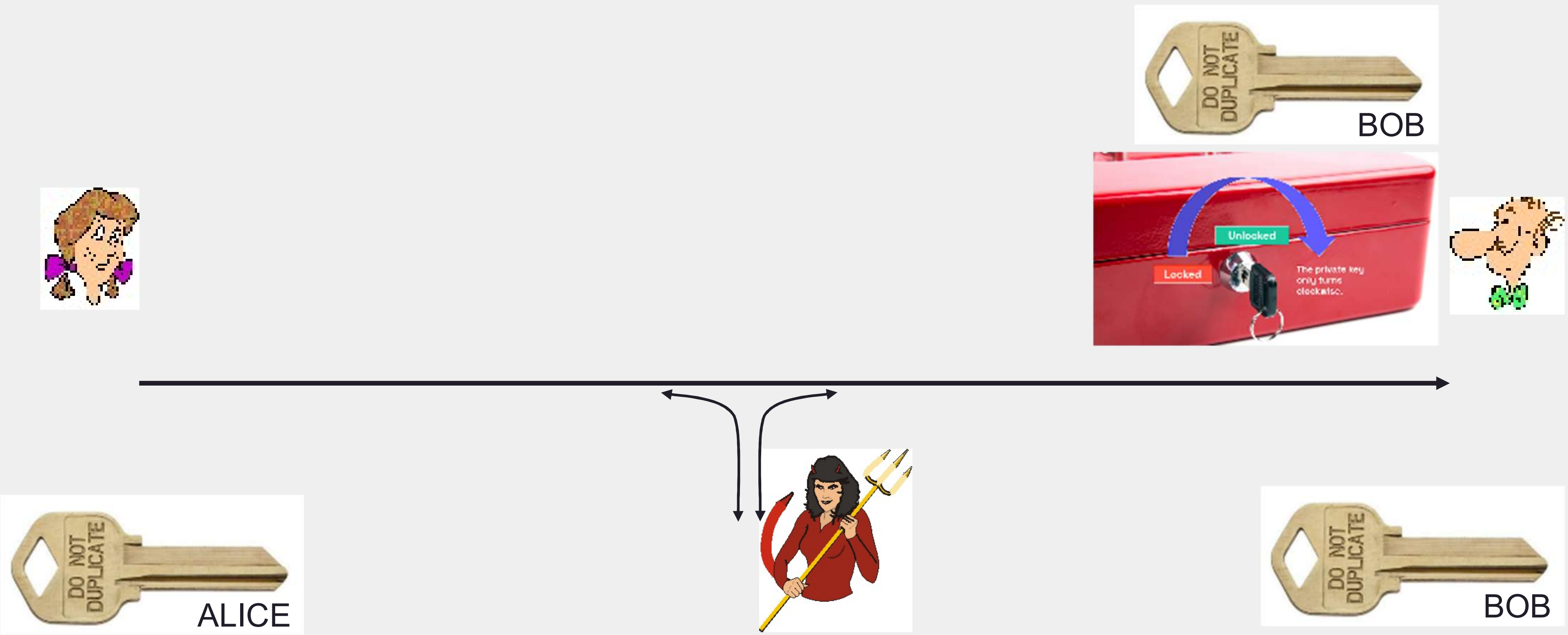
Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Alice sends a message to Bob **securely**



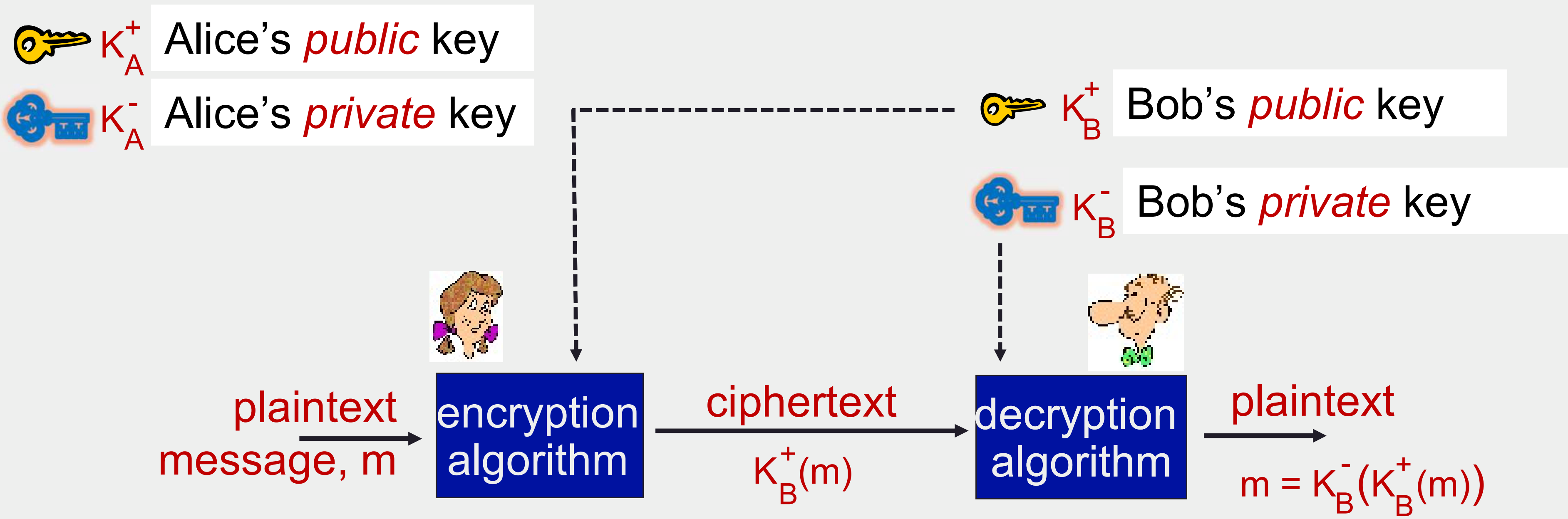
Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Alice sends a message to Bob **securely**



Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

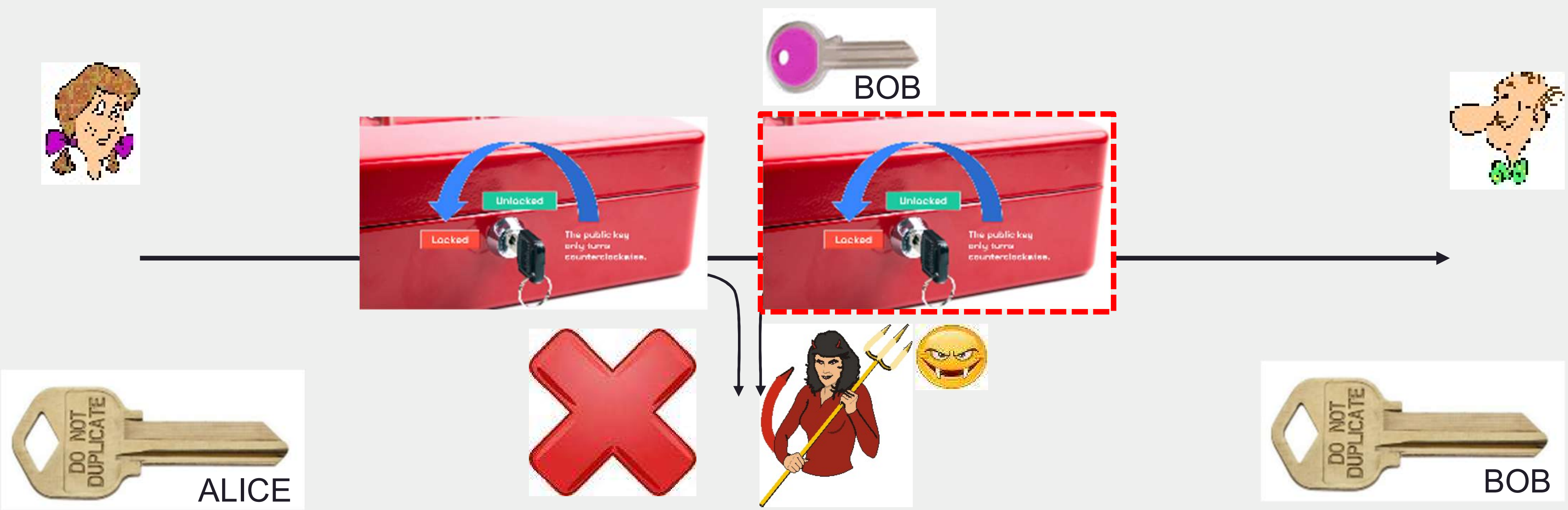
Public Key Cryptography



Property 1 $K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$ & $K_A^-(K_A^+(m)) = K_A^+(K_A^-(m)) = m$

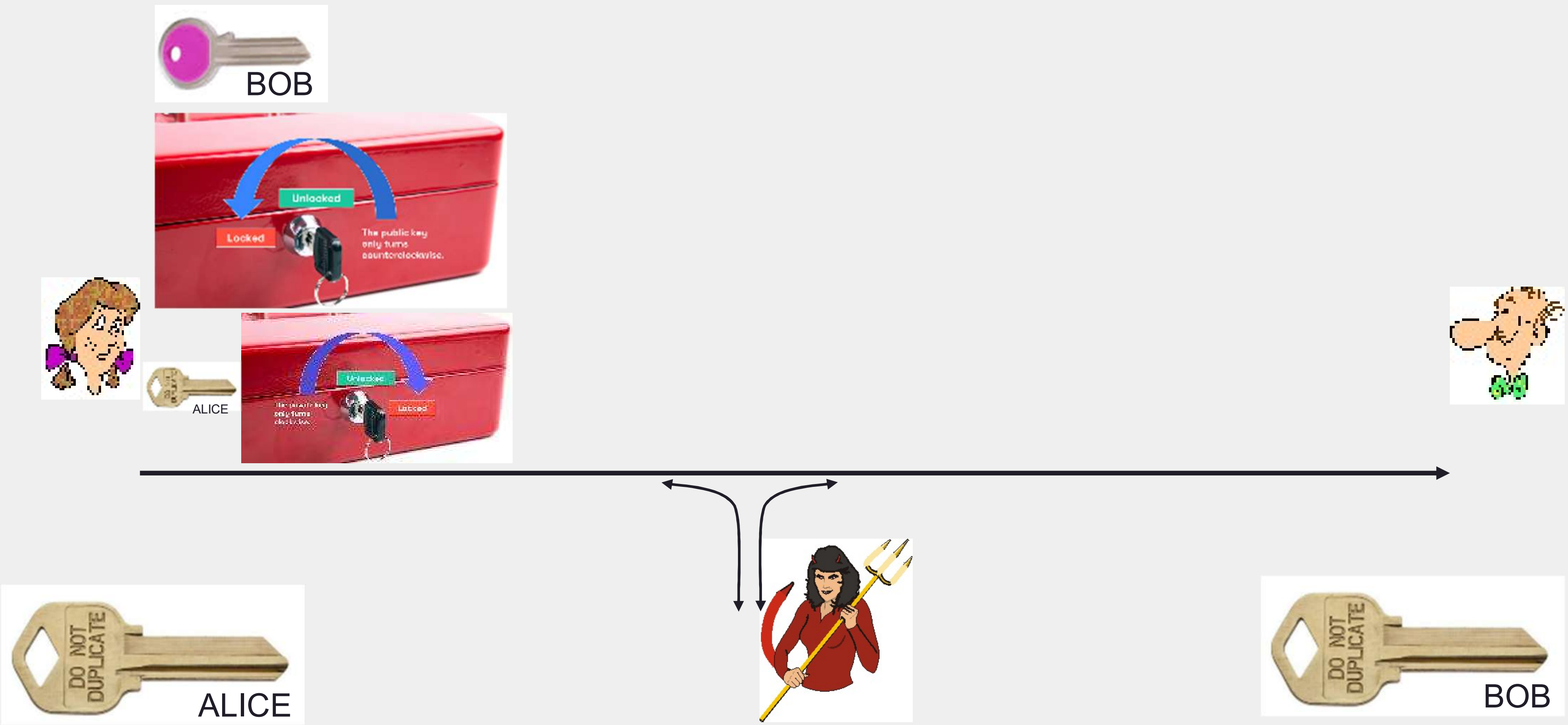
Property 2 – Given K_B^+, it must be computationally hard to compute K_B^-

Trudy can intercept Alice’s box, create her own message and pretend to be Alice



Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Alice sends a message to Bob **securely** with her **signature**

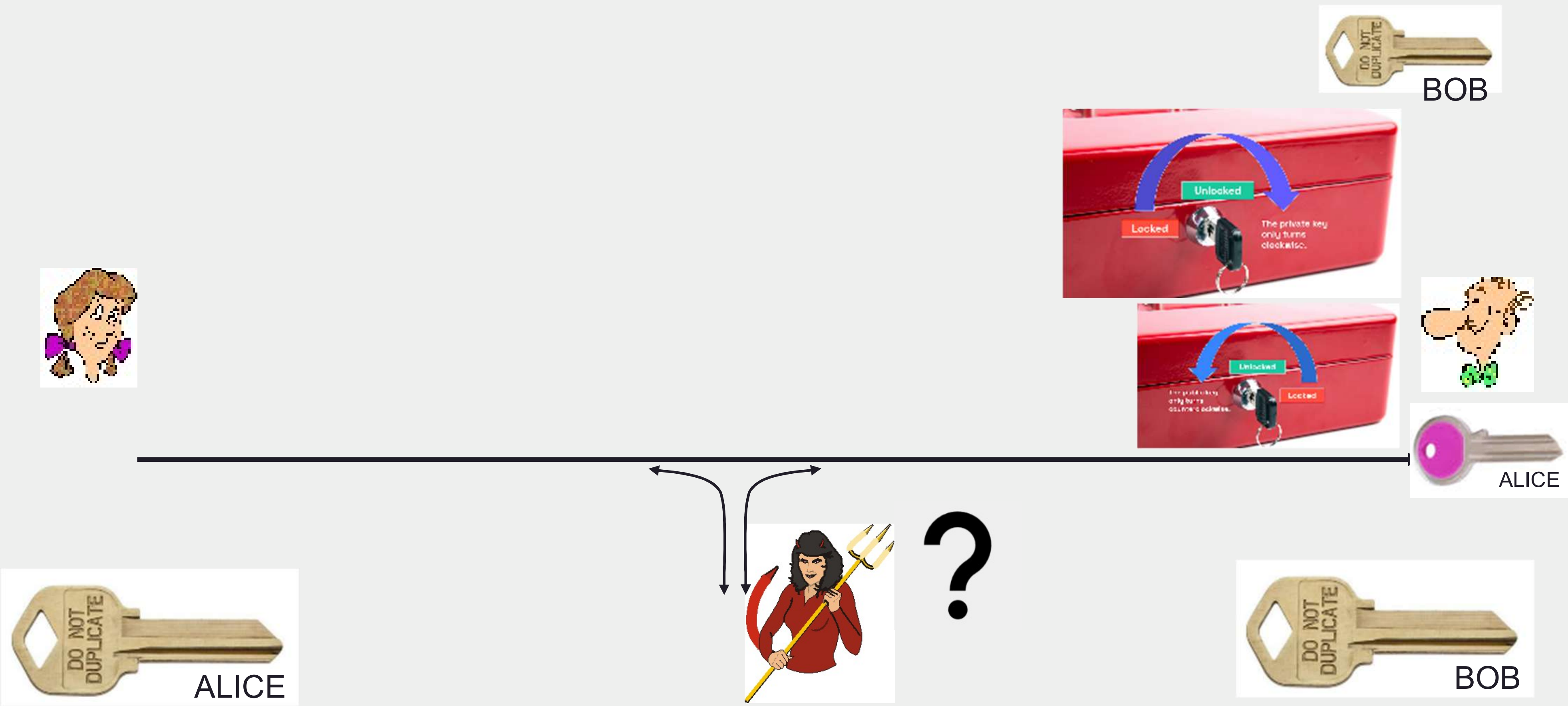


Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Alice sends a message to Bob **securely** with her **signature**

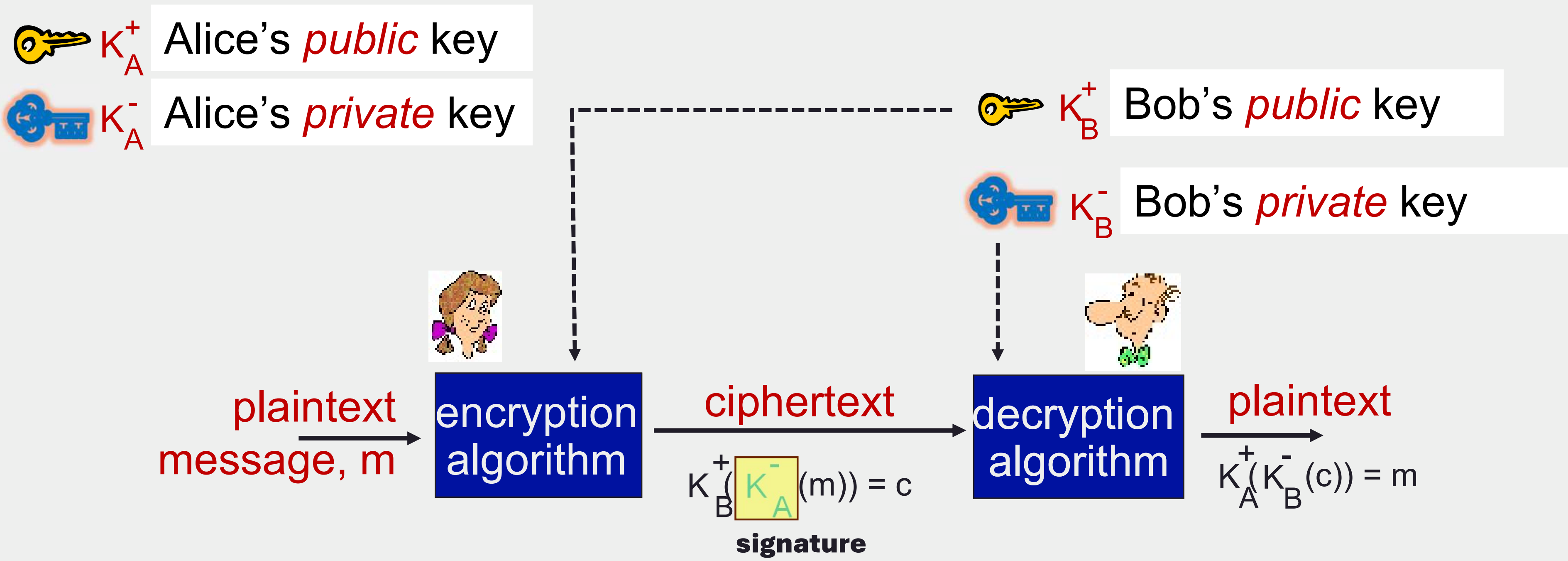
Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Alice sends a message to Bob **securely** with her **signature**



Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

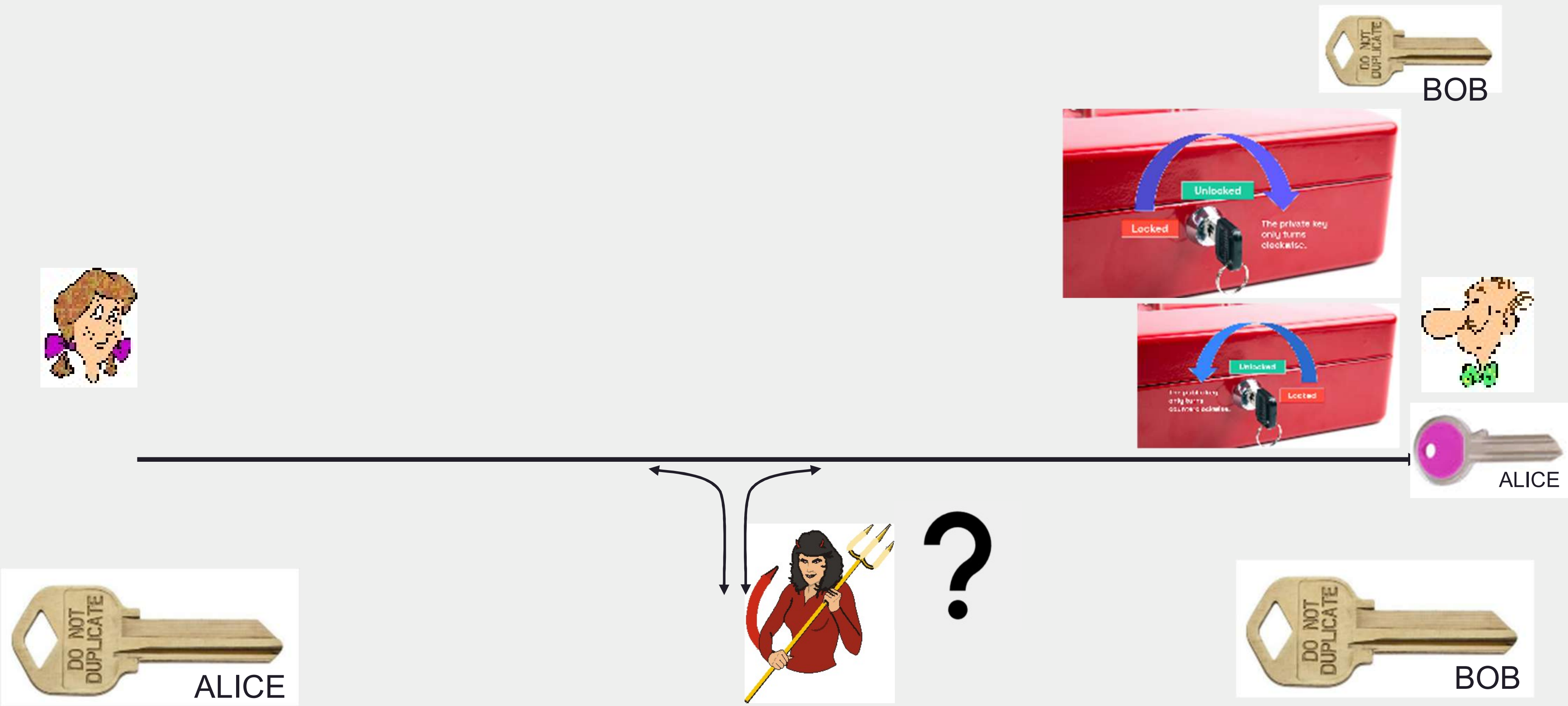
Public Key Cryptography



Property 1 $K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$ & $K_A^-(K_A^+(m)) = K_A^+(K_A^-(m)) = m$

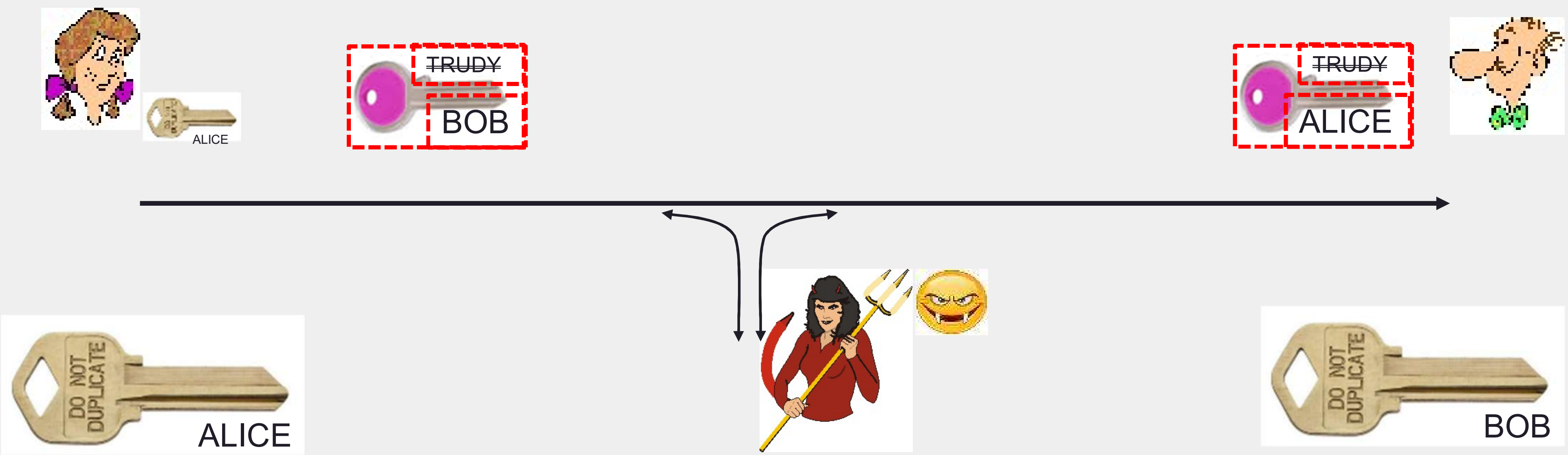
Property 2 – Given K_B^+, it must be computationally hard to compute K_B^-

Alice sends a message to Bob **securely** with her **signature**



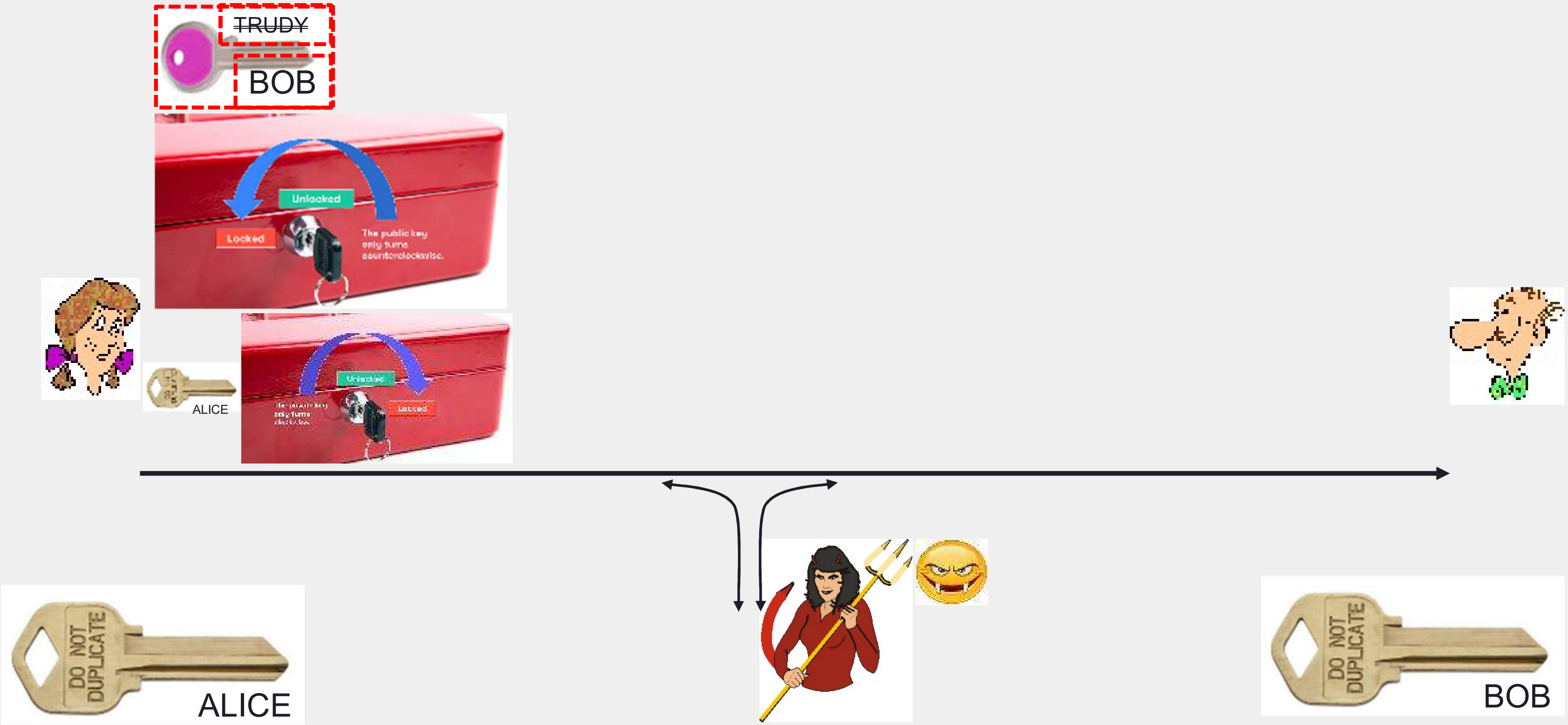
Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Trudy maliciously shares her Public key as Bob's and Alice's respectively



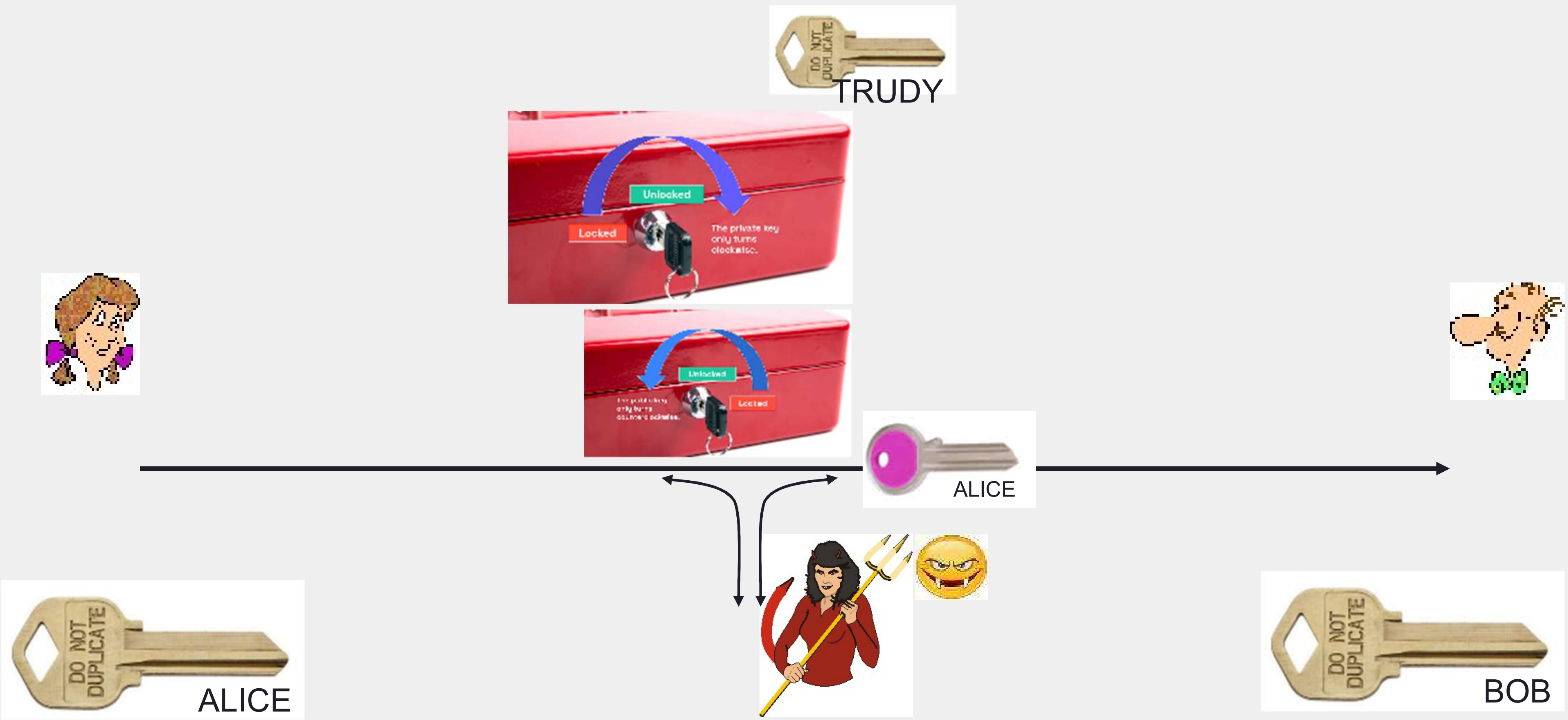
Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Alice sends a message to Bob securely with **Trudy's** signature (thinking it is **Bob's**)



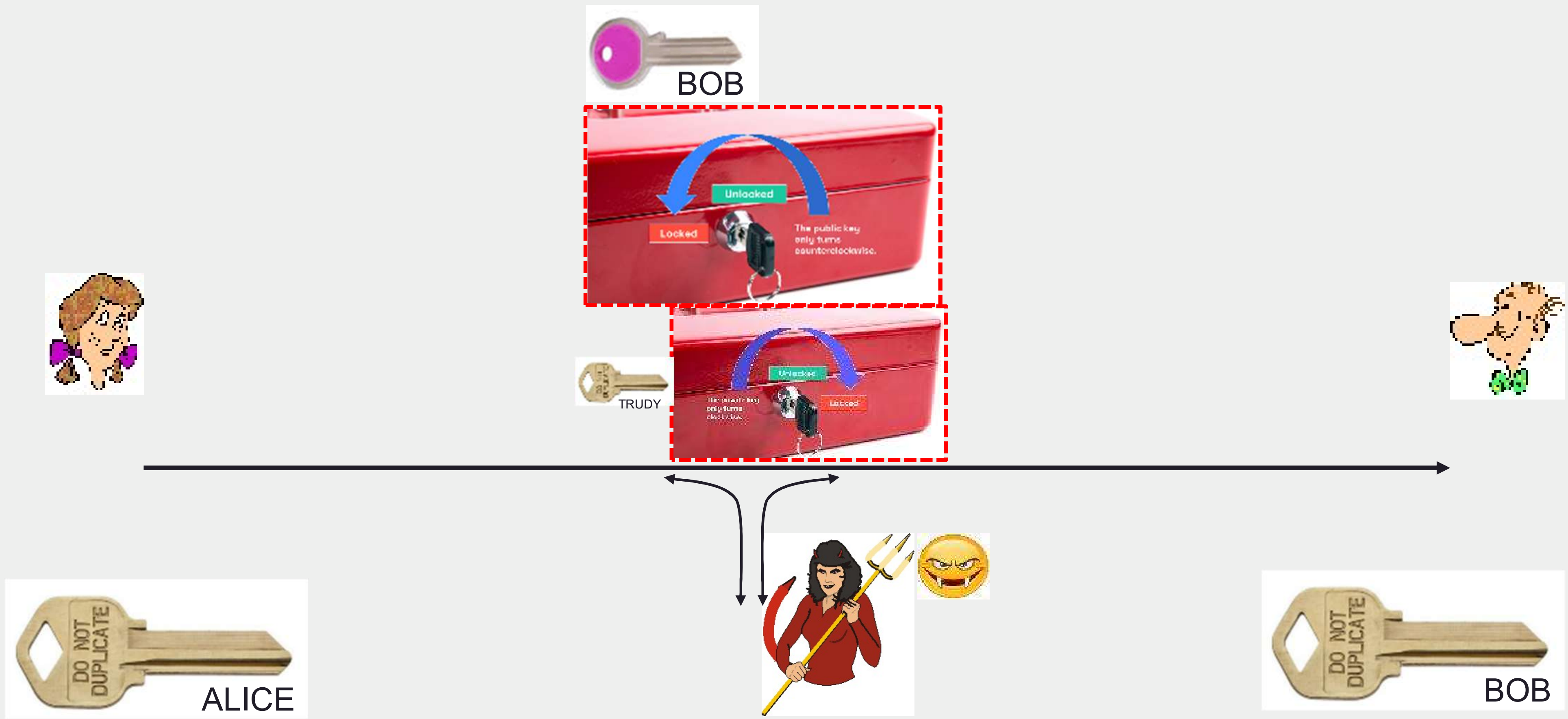
Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Trudy intercepts the message – and can access it using her Private key and Alice's public key



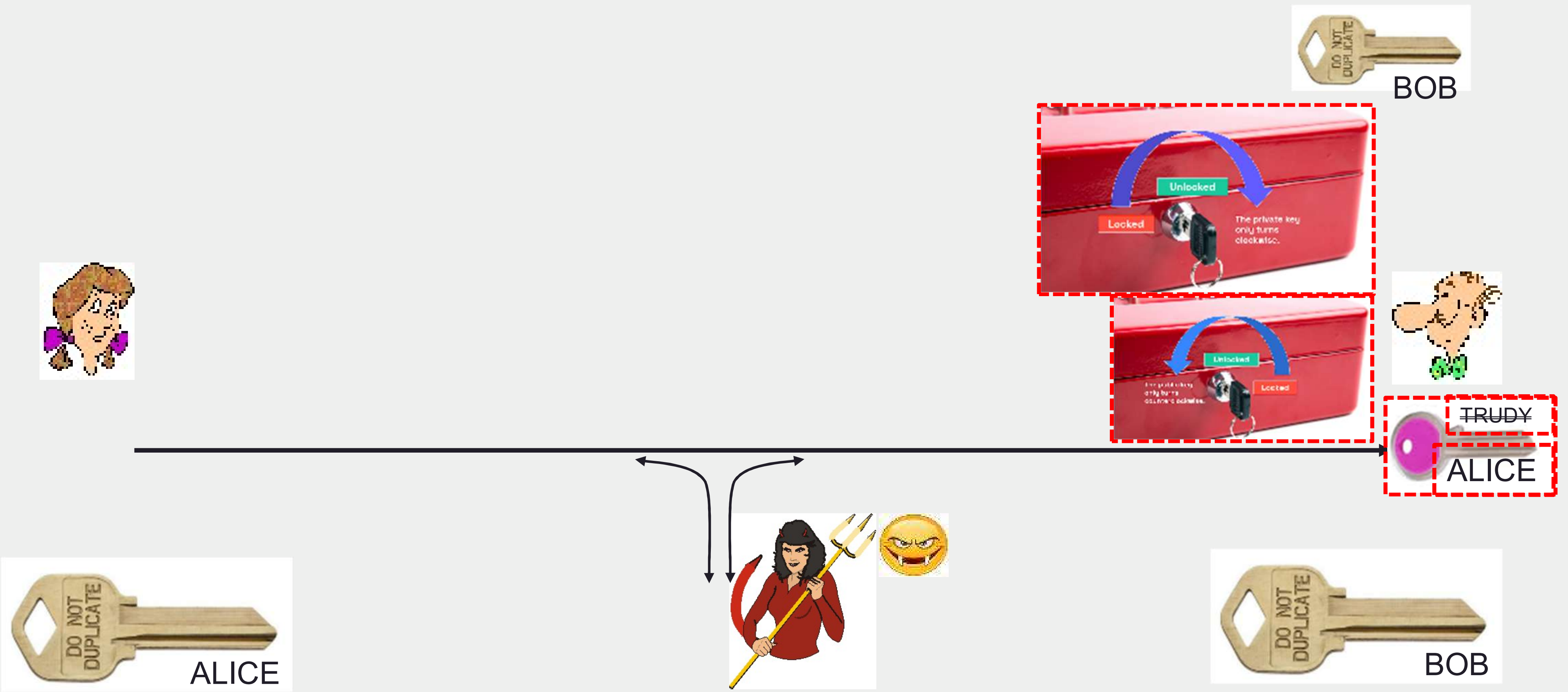
Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

Trudy sends malicious message to Bob securely with **Trudy's** signature, pretending to be **Alice**



Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

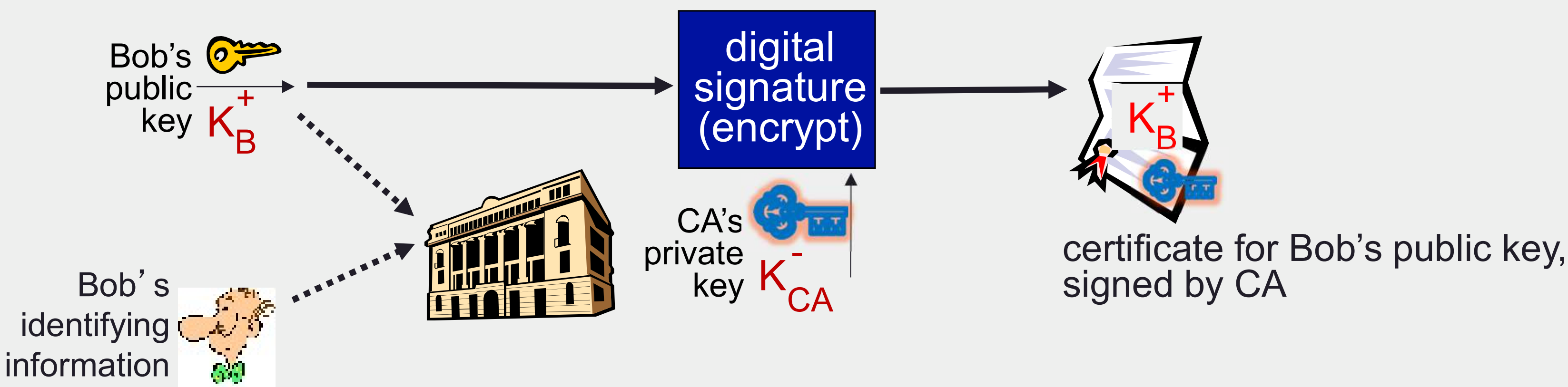
Bob uses his private key and Trudy's public key (thinking it is Alice's)



Source: https://auth0.com/blog/how-to-explain-public-key-cryptography-digital-signatures-to-anyone/

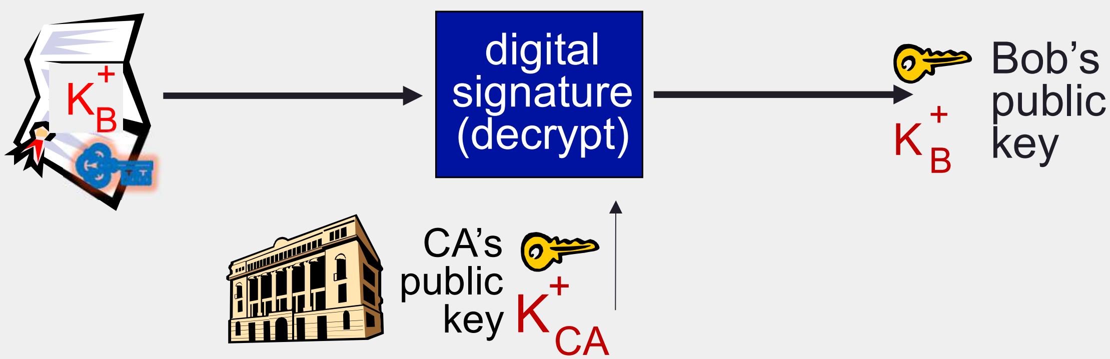
Public key Certification Authorities (CA)

- Certification authority (CA): binds public key to particular entity, E
- Entity (person, website, router) registers its public key with CE provides “proof of identity” to CA
  - CA creates certificate binding identity E to E’s public key
  - certificate containing E’s public key digitally signed by CA: CA says “this is E’s public key”

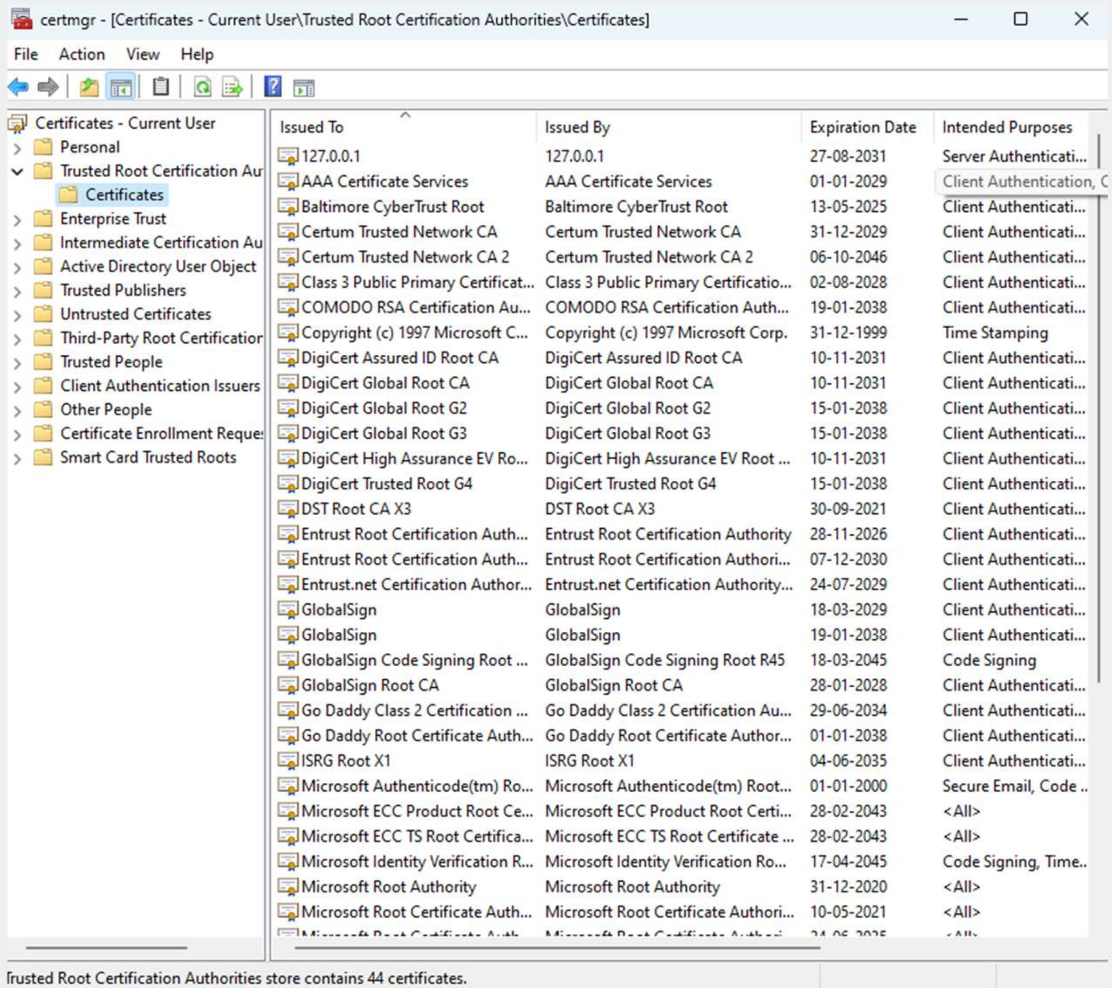


Public key Certification Authorities (CA)

- when Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere)
  - apply CA's public key to Bob's certificate, get Bob's public key



Certificate Manager on Windows – Public Keys of several Certification Authorities



RSA Details

Recap: Modulo Arithmetic

- x mod n = remainder of x when divide by n
- Modulo Arithmetic facts:
  - [(a mod n) + (b mod n)] mod n = (a+b) mod n
  - [(a mod n) - (b mod n)] mod n = (a-b) mod n
  - [(a mod n) * (b mod n)] mod n = (a*b) mod n
- thus
  - $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: a = 14, n = 10, d = 2:

  LHS: $(a \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  RHS: $a^d \bmod n = 14^2 \bmod 10 = 196 \bmod 10 = 6$

RSA: getting ready

- Message: sequence of bits
- Each bit pattern can be uniquely represented by an integer number
- Encrypting a message is equivalent to encrypting a number
- example:
  - m = **10010001**. This message is uniquely represented by the decimal number **145**.
- To encrypt m,
  - We encrypt the corresponding number (eg 145), which gives a new number (the ciphertext)

RSA: Creating public/private key pair

1. Choose two large prime numbers *p, q*. (e.g., 1024 bits each)

2. Compute $n = pq$, $z = (p-1)(q-1)$

3. Choose a **small** *e* (with $e<n$) that has no common factors with z (*e, z* are “relatively prime”).

4. choose *d* (≠ **e**) such that $ed-1$ is exactly divisible by *z*. (in other words: $ed \bmod z = 1$).

5. *public* key is $\underbrace{(n,e)}_{K_B^+}$. *private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

1. Given (n,e) and (n,d) as computed above

2. To encrypt message m ($<n$), compute ciphertext c

$$c = m^e \bmod n$$

3. To decrypt received ciphertext, c, compute

$$m = c^d \bmod n$$

Magic happens! $m = (m^e \bmod n)^d \bmod n$

RSA Example: Bob creates public/private key pair

1. Choose two large prime numbers p, q. (***p = 5, q = 7***)

2. Compute $n = pq$, $z = (p-1)(q-1)$ --- ***n = 35, z = 24***

3. Choose e (with $e<n$) that has no common factors with z (e, z are "relatively prime"). ***e = 5***

4. choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$). ***d = 29***

5. *public* key is (n,e). *private* key is (n,d).
   (35,5) ***(35,29)***

To encrypt a message ***m = 12***, Alice uses Bob's Public Key ***(35, 5)*** and computes

$$c = m^e \bmod n = 12^5 \bmod 35 = 248832 \bmod 35 = 17$$

To decrypt the ciphertext ***c = 17***, Bob uses his Private Key ***(35, 29)*** and computes

$$c = c^d \bmod n = 17^{29} \bmod 35 = 481968572106750915091411825223071697 \bmod 35 = 12$$

Why does RSA work?

- We must show that for any cipher *c*, $c^d \bmod n = m$, where cipher $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where n= pq and z = (p-1)(q-1)
- thus,

$$
\begin{aligned}
c^d \bmod n &= (m^e \bmod n)^d \bmod n \\
&= m^{ed} \bmod n \\
&= m^{(ed \bmod z)} \bmod n \\
&= m^1 \bmod n \\
&= m
\end{aligned}
$$

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}} = m$$

result is the same!

$$
\begin{aligned}
(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\
&= m^{de} \bmod n \\
&= (m^d \bmod n)^e \bmod n
\end{aligned}
$$

Why is RSA secure?

- Suppose you know Bob's public key ***(n,e)***. How hard is it to determine ***d***?
- Essentially need to find factors of ***n*** without knowing the two factors ***p*** and ***q***
- Finding Prime Factors of a large number is **computationally HARD**

Questions and Discussions

Email: venkateswaranr.comp@coeptech.ac.in