

Public Key Infrastructure

An Overview of Asymmetric Key Encryption

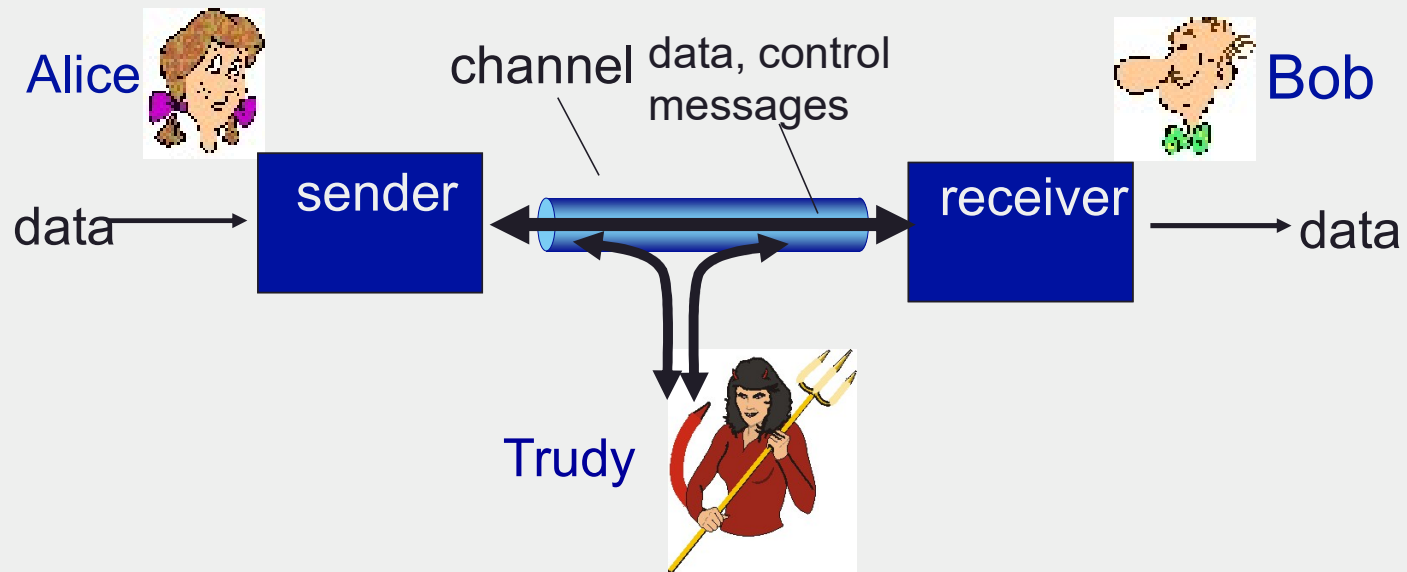
R. Venkateswaran

There are bad guys (and girls) out there!

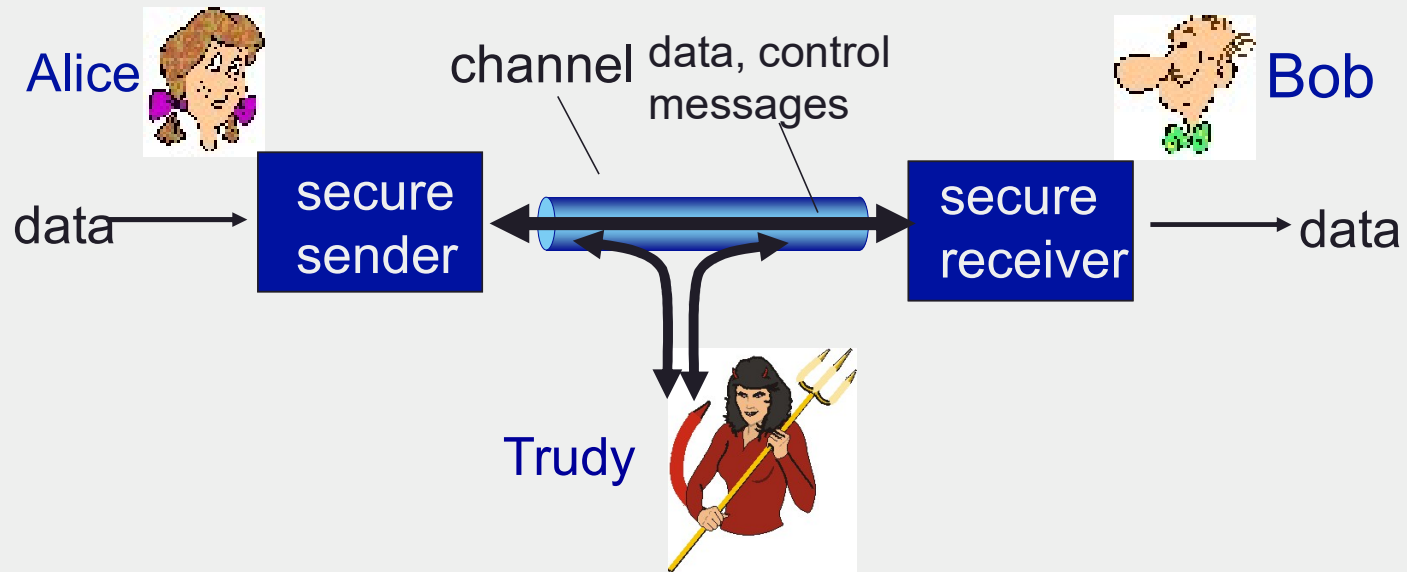
- Q: What can a “bad guy” do?
- A: A lot!
 - eavesdrop: intercept messages
 - actively insert messages into connection
 - impersonation: can fake (spoof) source address in packet (or any field in packet)
 - hijacking: “take over” ongoing connection by removing sender or receiver, inserting himself in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

Friends and enemies: Alice, Bob, Trudy

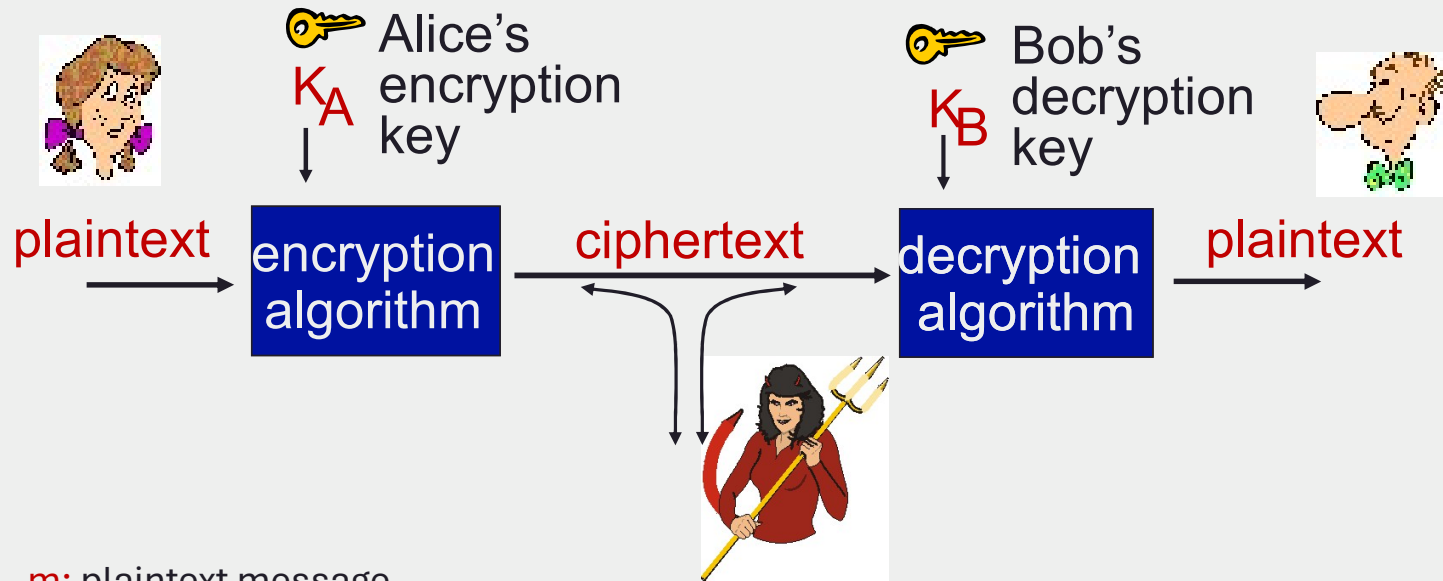
- well-known in network security world
- Alice & Bob want to communicate with each other
- Trudy (intruder) may intercept, delete, add messages



Alice and Bob want to communicate **securely**



The language of cryptography

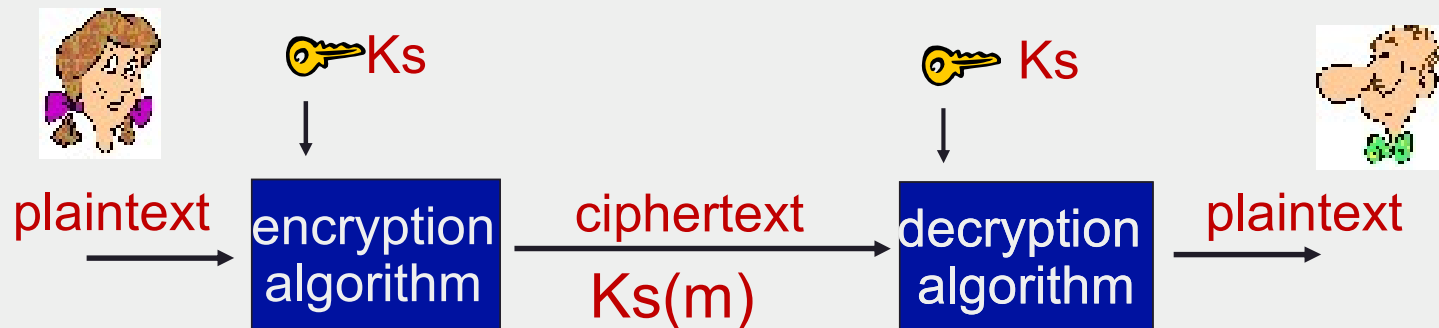


m : plaintext message

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

$m = K_B(K_A(m))$

Symmetric key cryptography

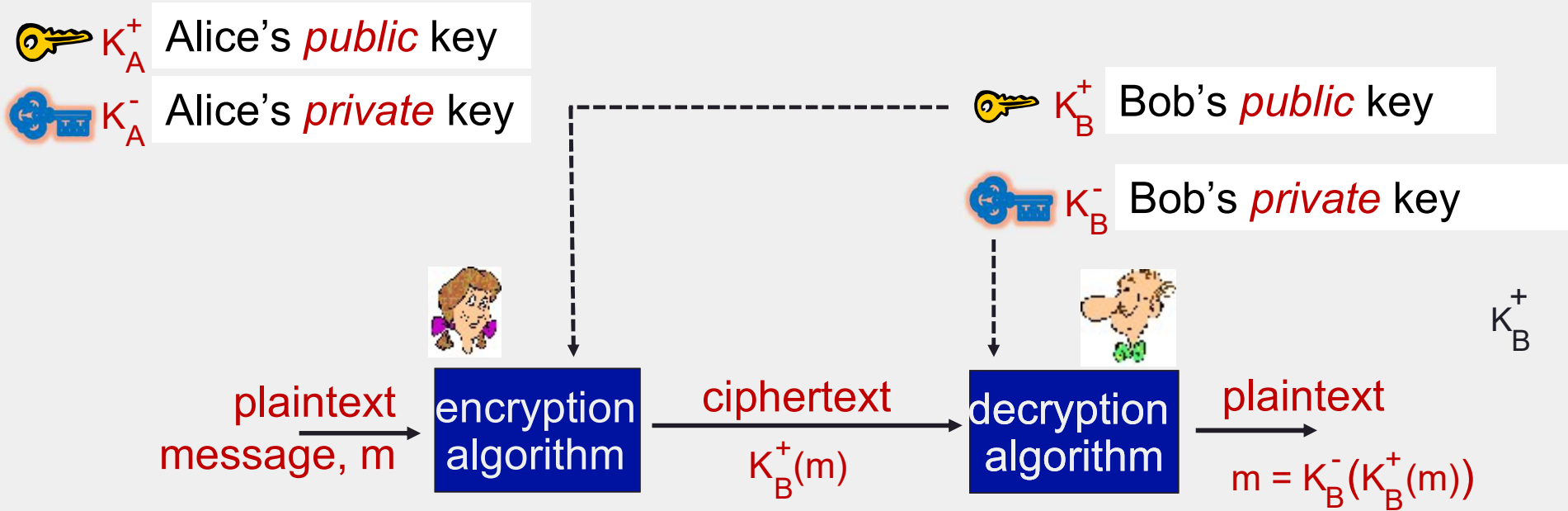


symmetric key crypto: Bob and Alice share same (symmetric) key K_s

K_s satisfies the following property : $K_s(K_s(m)) = m$

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

Public Key Cryptography



Public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

Public Key Infrastructure - Simple Analogy

The box has a lock with three positions:

Unlocked

Locked

Locked



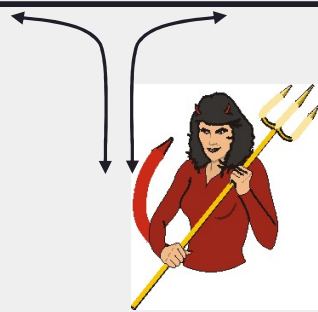
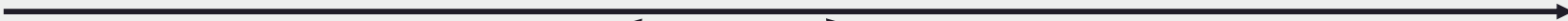
Golden Key – ONLY ONE COPY.
Turns **CLOCKWISE**



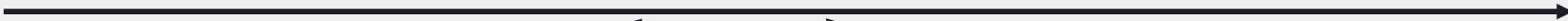
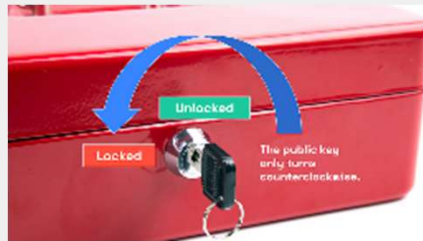
Regular Key – Many Copies.
Turns **ANTI-CLOCKWISE**



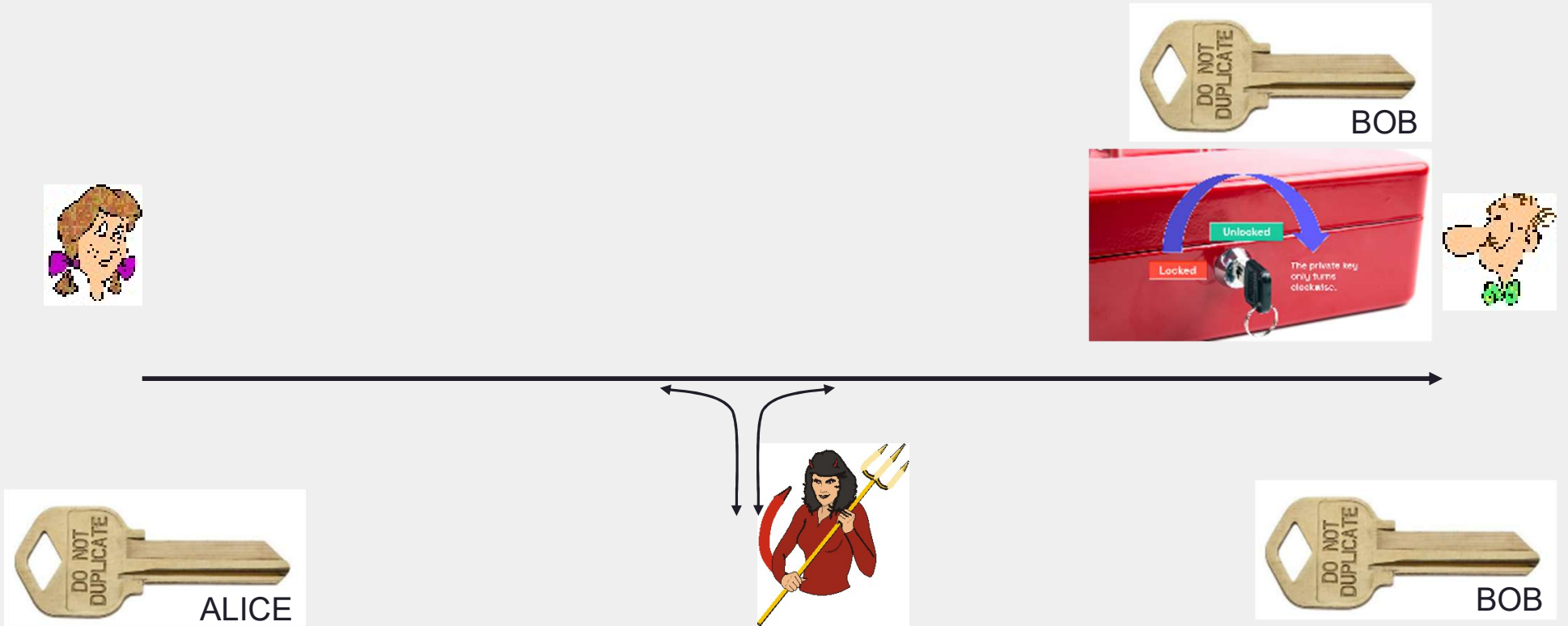
Alice sends a message to Bob **securely**



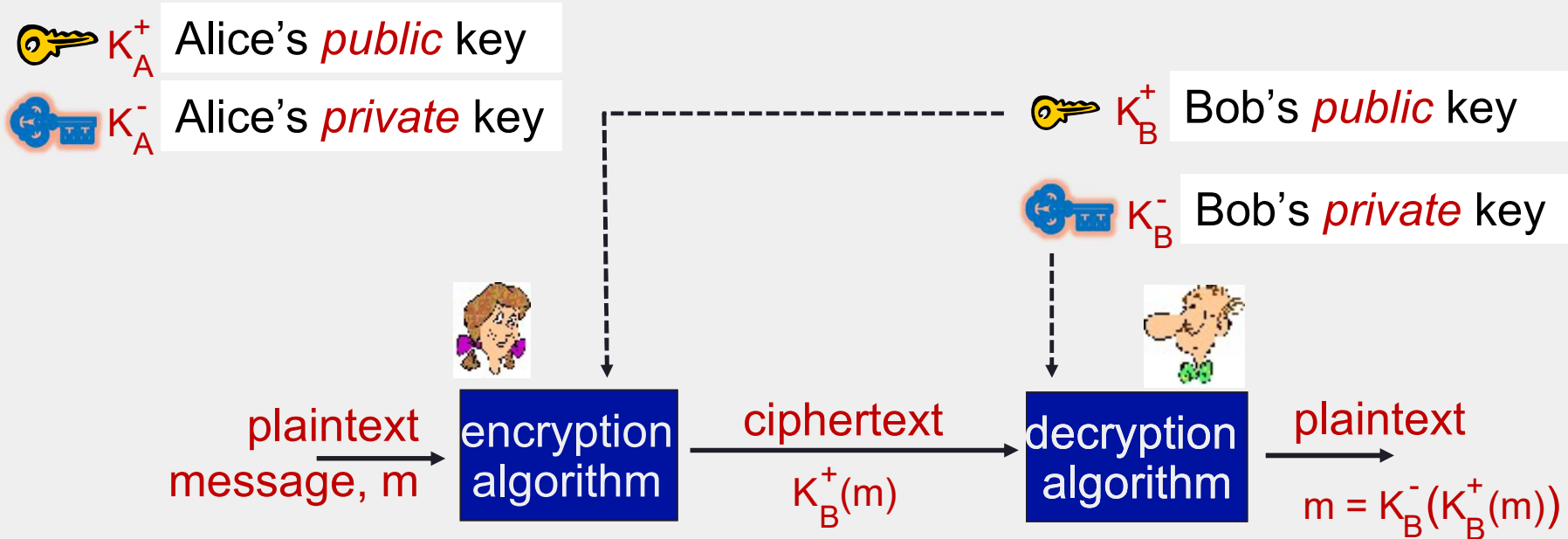
Alice sends a message to Bob **securely**



Alice sends a message to Bob **securely**



Public Key Cryptography



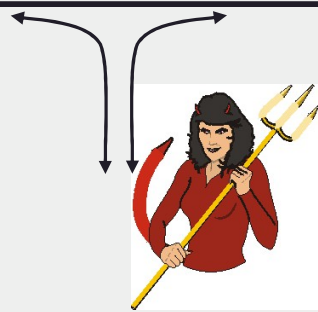
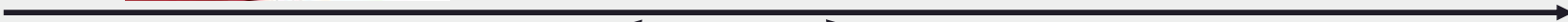
Property 1 $K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$ & $K_A^-(K_A^+(m)) = K_A^+(K_A^-(m)) = m$

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

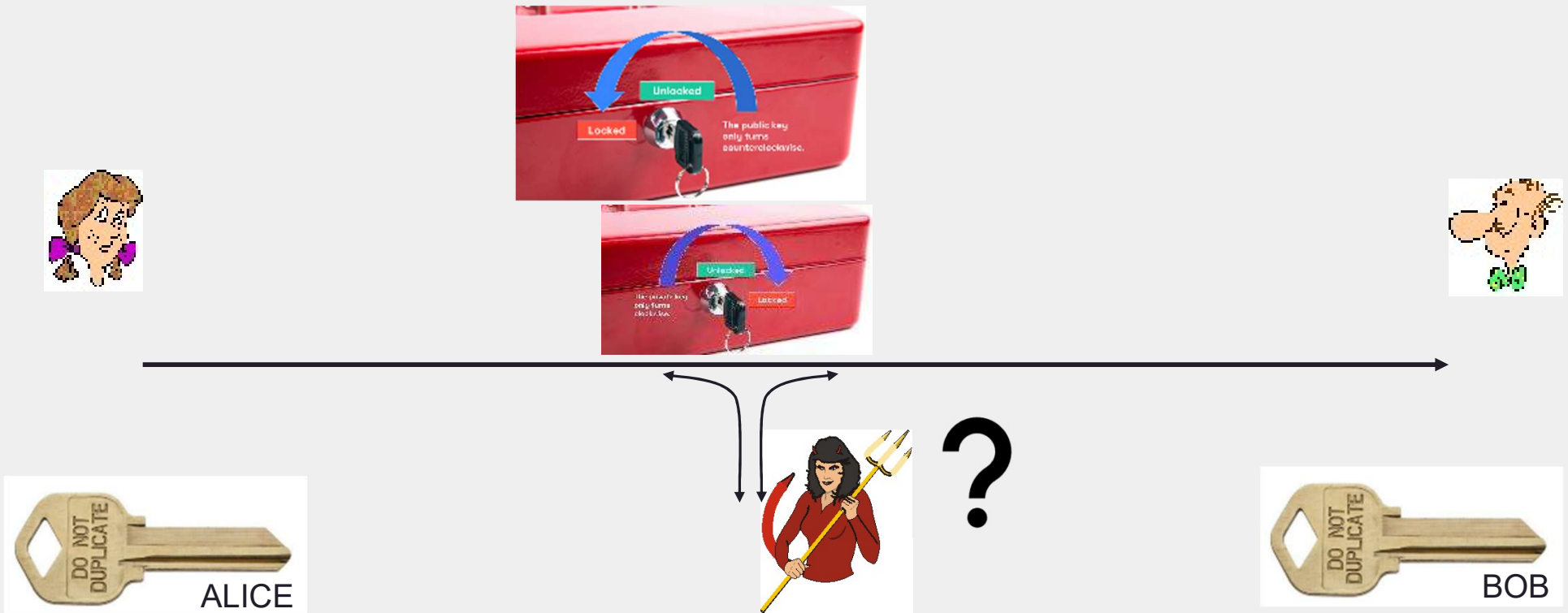
Trudy can intercept Alice's box, create her **own message** and pretend to be Alice



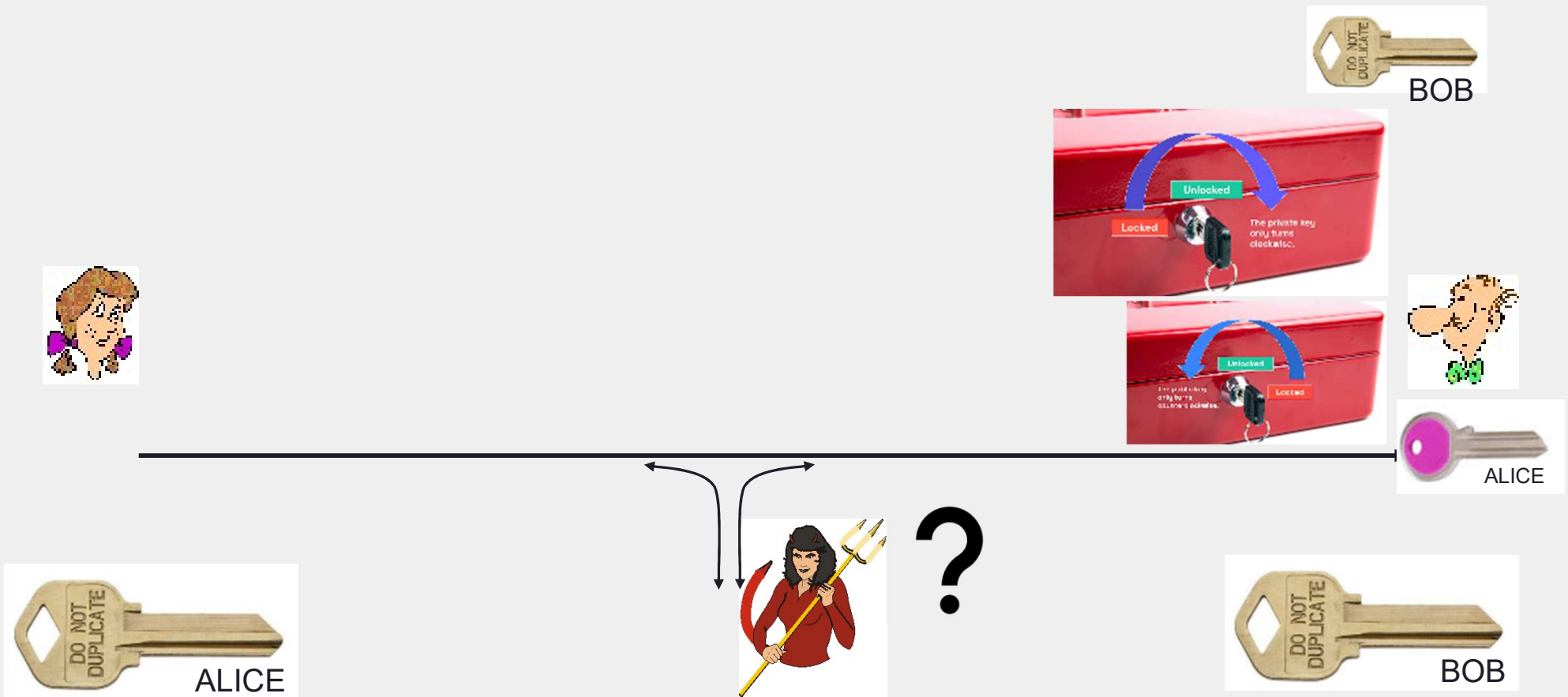
Alice sends a message to Bob **securely** with her **signature**



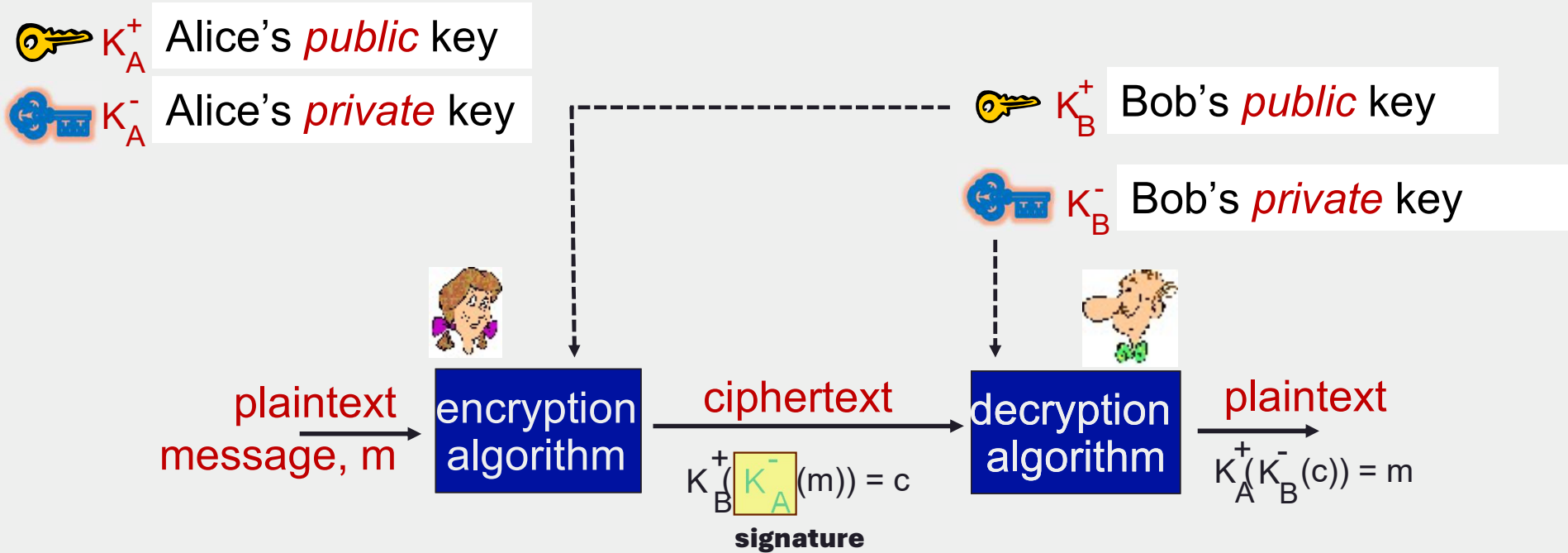
Alice sends a message to Bob **securely** with her **signature**



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



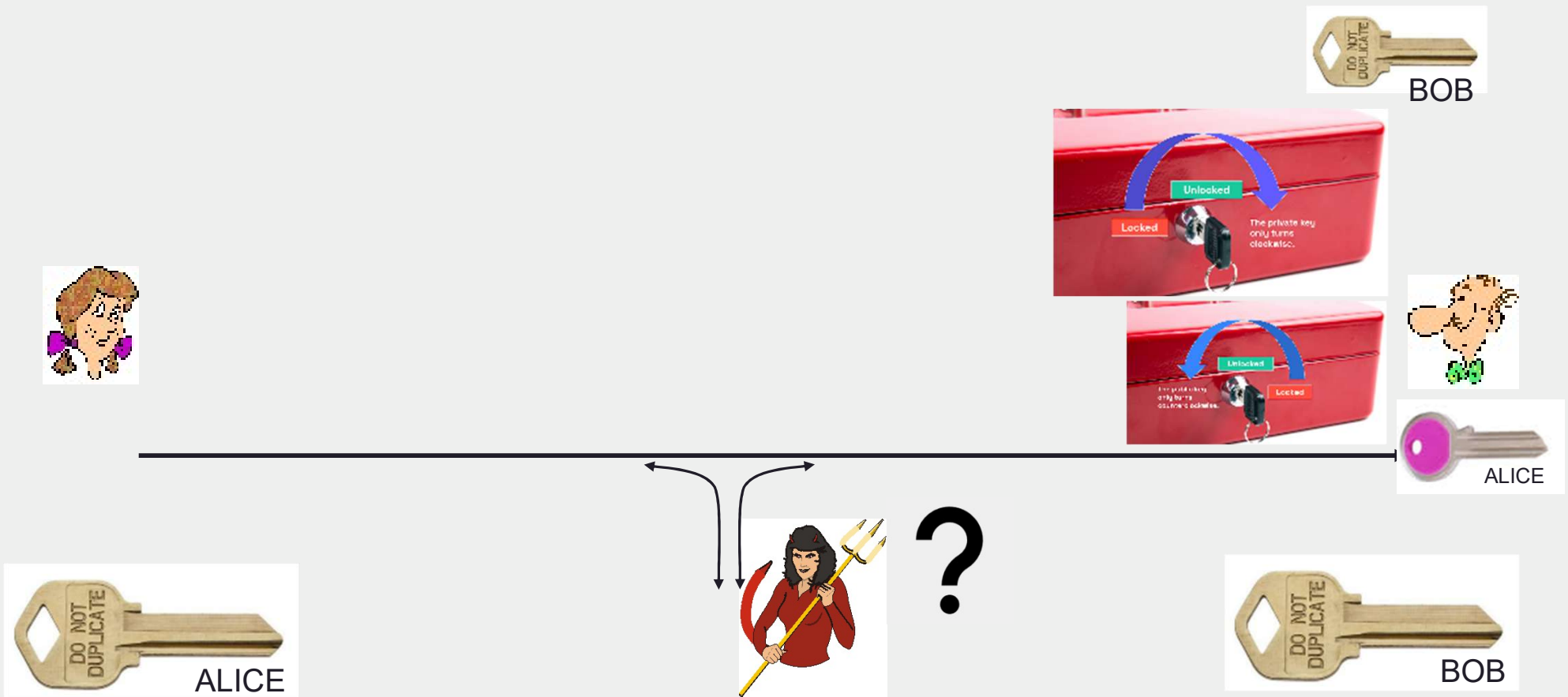
Public Key Cryptography



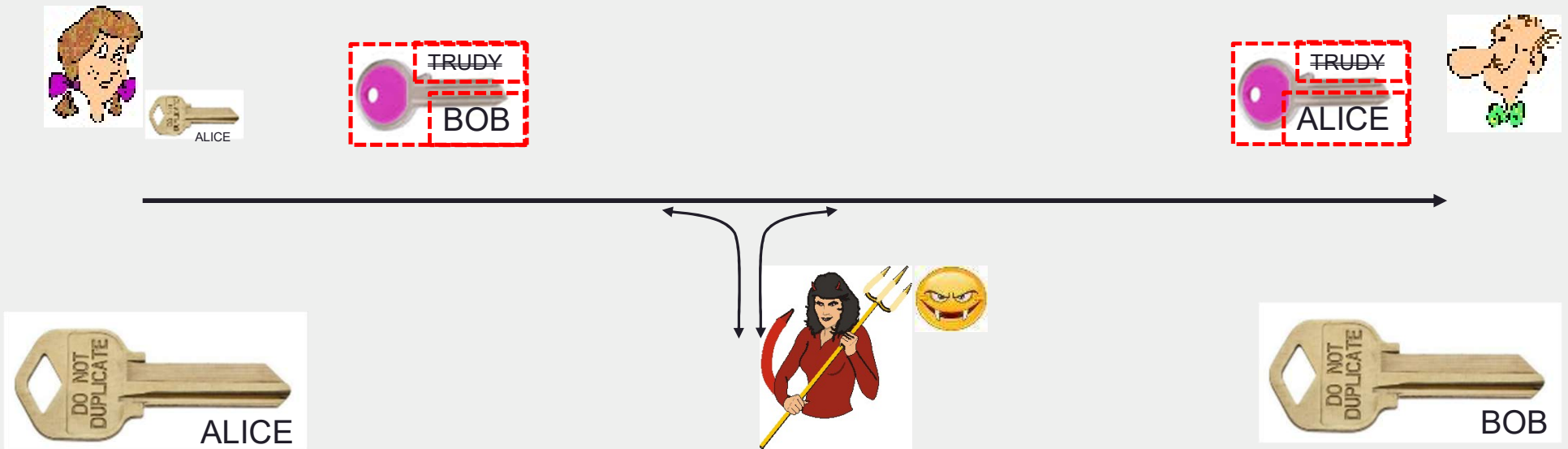
Property 1 $K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$ & $K_A^-(K_A^+(m)) = K_A^+(K_A^-(m)) = m$

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

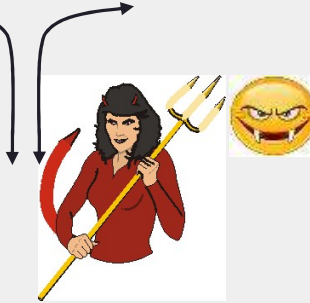
Alice sends a message to Bob **securely** with her **signature**



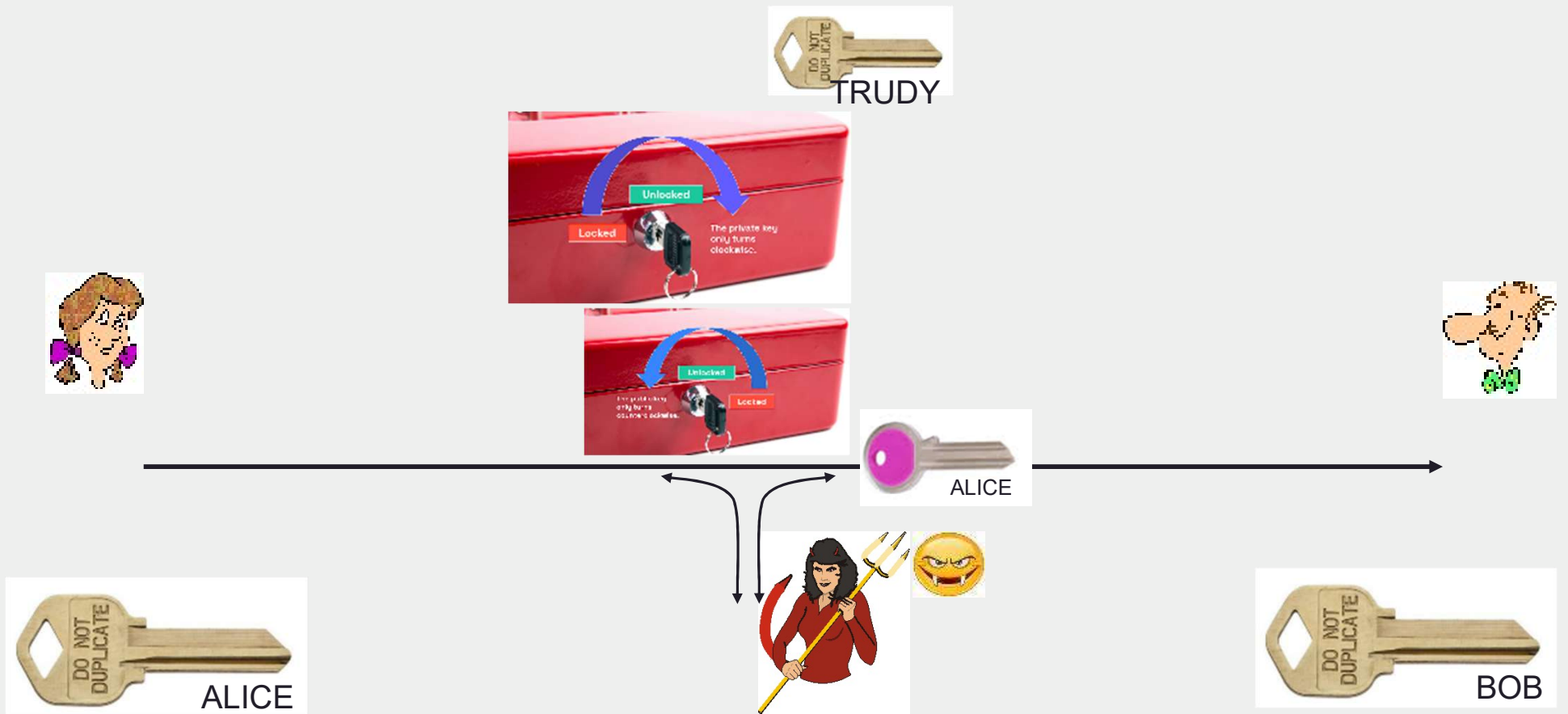
Trudy maliciously shares her Public key as Bob's and Alice's respectively



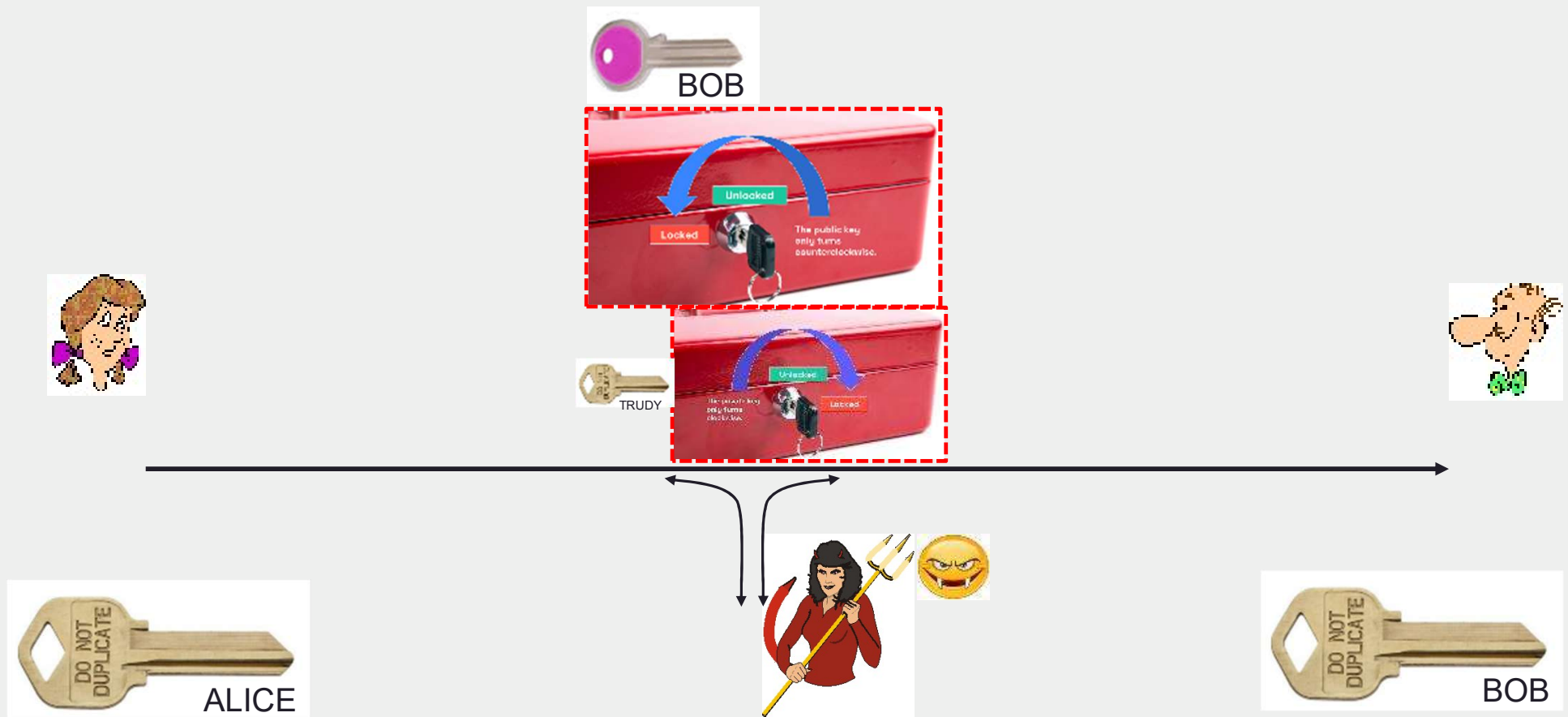
Alice sends a message to Bob securely with **Trudy's** signature (thinking it is **Bob's**)



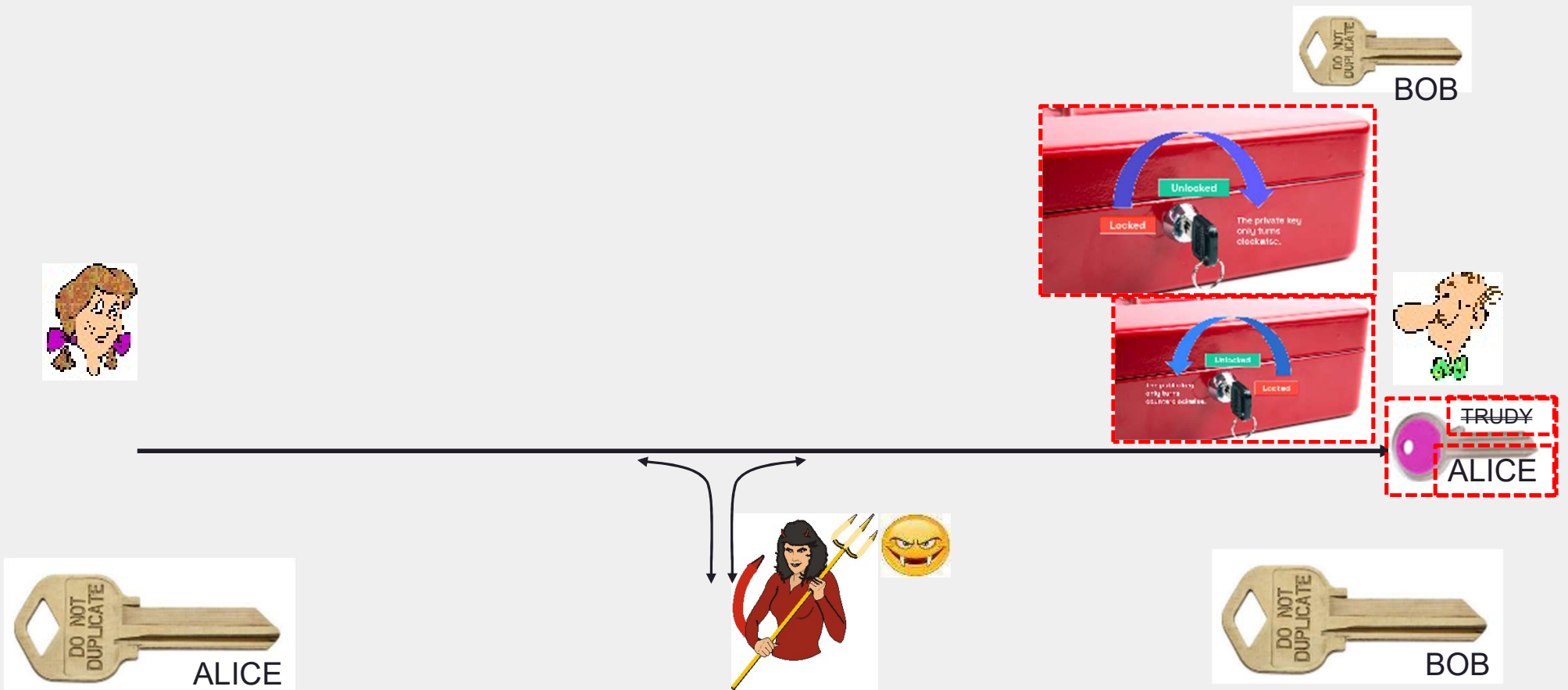
Trudy intercepts the message – and can access it using her Private key and Alice’s public key



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

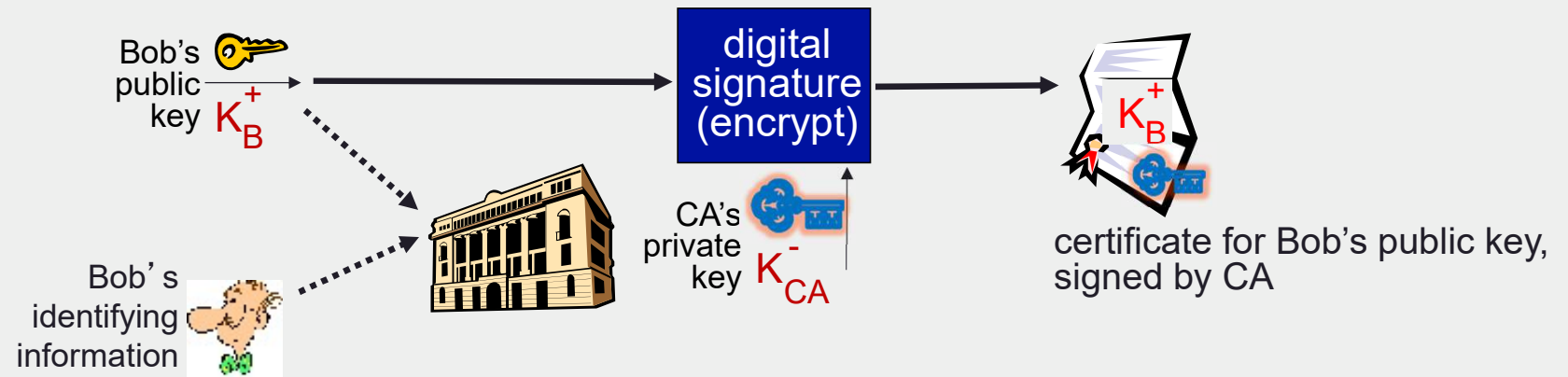


Bob uses his private key and **Trudy's** public key (thinking it is **Alice's**)



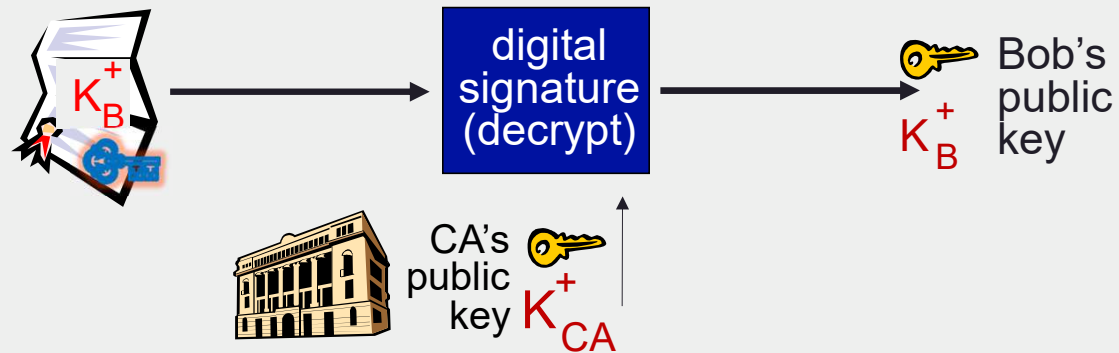
Public key Certification Authorities (CA)

- 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”

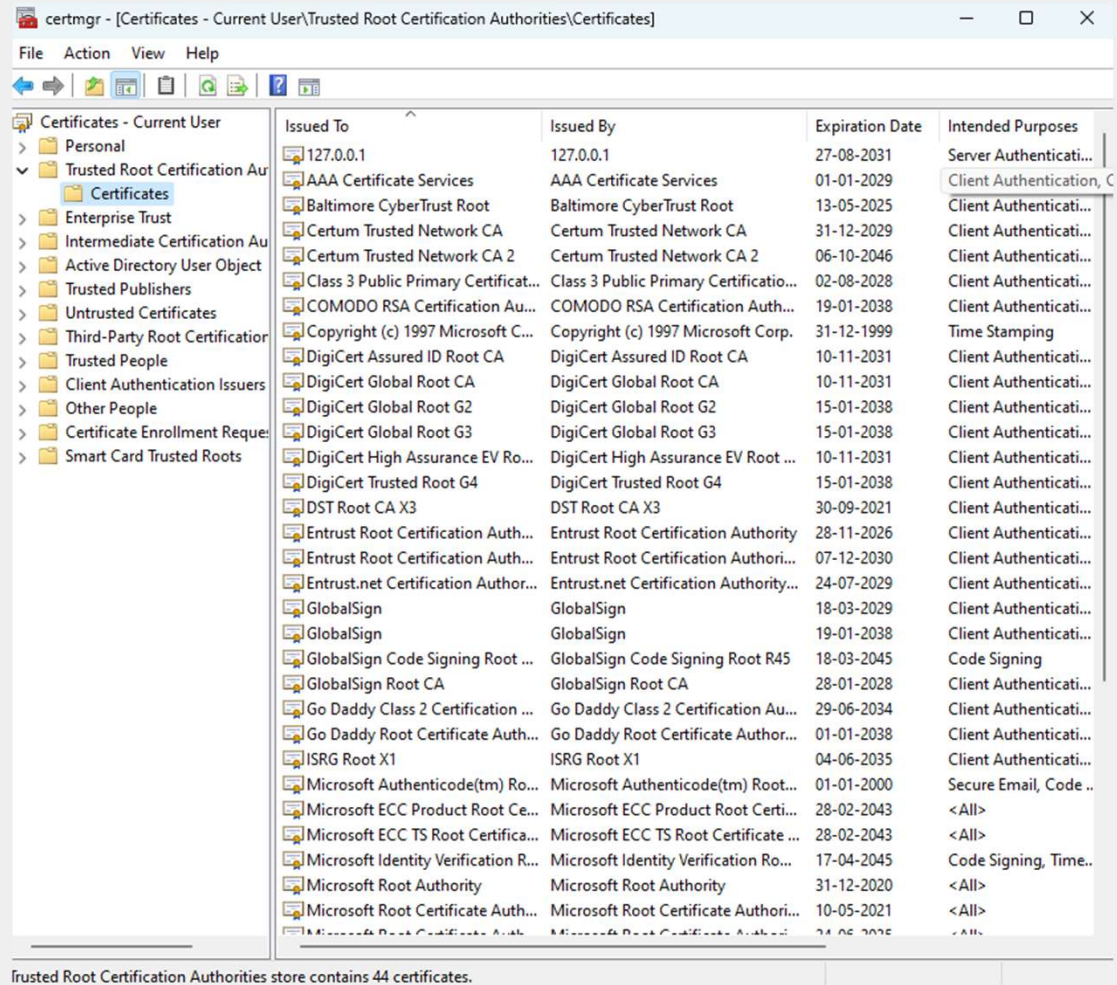


Public key Certification Authorities (CA)

- 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



The screenshot displays the Windows Certificate Manager (certmgr) window, specifically the 'Certificates - Current User\Trusted Root Certification Authorities\Certificates' view. The window shows a list of 44 certificates issued by various Certification Authorities (CAs). The list is organized into columns: Issued To, Issued By, Expiration Date, and Intended Purposes. The 'Intended Purposes' column is expanded, showing a list of purposes for each certificate, such as 'Server Authentication', 'Client Authentication', 'Time Stamping', 'Code Signing', and 'Secure Email, Code Signing'.

Issued To	Issued By	Expiration Date	Intended Purposes
127.0.0.1	127.0.0.1	27-08-2031	Server Authentication...
AAA Certificate Services	AAA Certificate Services	01-01-2029	Client Authentication...
Baltimore CyberTrust Root	Baltimore CyberTrust Root	13-05-2025	Client Authentication...
Certum Trusted Network CA	Certum Trusted Network CA	31-12-2029	Client Authentication...
Certum Trusted Network CA 2	Certum Trusted Network CA 2	06-10-2046	Client Authentication...
Class 3 Public Primary Certificat...	Class 3 Public Primary Certificatio...	02-08-2028	Client Authentication...
COMODO RSA Certification Auth...	COMODO RSA Certification Auth...	19-01-2038	Client Authentication...
Copyright (c) 1997 Microsoft C...	Copyright (c) 1997 Microsoft Corp.	31-12-1999	Time Stamping
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA	10-11-2031	Client Authentication...
DigiCert Global Root CA	DigiCert Global Root CA	10-11-2031	Client Authentication...
DigiCert Global Root G2	DigiCert Global Root G2	15-01-2038	Client Authentication...
DigiCert Global Root G3	DigiCert Global Root G3	15-01-2038	Client Authentication...
DigiCert High Assurance EV Ro...	DigiCert High Assurance EV Root ...	10-11-2031	Client Authentication...
DigiCert Trusted Root G4	DigiCert Trusted Root G4	15-01-2038	Client Authentication...
DST Root CA X3	DST Root CA X3	30-09-2021	Client Authentication...
Entrust Root Certification Auth...	Entrust Root Certification Authority	28-11-2026	Client Authentication...
Entrust Root Certification Auth...	Entrust Root Certification Authori...	07-12-2030	Client Authentication...
Entrust.net Certification Author...	Entrust.net Certification Authority...	24-07-2029	Client Authentication...
GlobalSign	GlobalSign	18-03-2029	Client Authentication...
GlobalSign	GlobalSign	19-01-2038	Client Authentication...
GlobalSign Code Signing Root ...	GlobalSign Code Signing Root R45	18-03-2045	Code Signing
GlobalSign Root CA	GlobalSign Root CA	28-01-2028	Client Authentication...
Go Daddy Class 2 Certification ...	Go Daddy Class 2 Certification Au...	29-06-2034	Client Authentication...
Go Daddy Root Certificate Author...	Go Daddy Root Certificate Author...	01-01-2038	Client Authentication...
ISRG Root X1	ISRG Root X1	04-06-2035	Client Authentication...
Microsoft Authenticode(tm) Ro...	Microsoft Authenticode(tm) Root...	01-01-2000	Secure Email, Code ..
Microsoft ECC Product Root Ce...	Microsoft ECC Product Root Certi...	28-02-2043	<All>
Microsoft ECC TS Root Certifica...	Microsoft ECC TS Root Certificate ...	28-02-2043	<All>
Microsoft Identity Verification R...	Microsoft Identity Verification Ro...	17-04-2045	Code Signing, Time..
Microsoft Root Authority	Microsoft Root Authority	31-12-2020	<All>
Microsoft Root Certificate Auth...	Microsoft Root Certificate Authori...	10-05-2021	<All>

Trusted Root Certification Authorities store contains 44 certificates.

RSA Details

Recap: Modulo Arithmetic

- $x \bmod n$ = remainder of x when divide by n
- Modulo Arithmetic facts:
 - $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$
 - $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$
 - $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$
- 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 ($\neq e$) such that $ed-1$ is exactly divisible by z . (in other words: $ed \bmod z = 1$).
5. *public* key is (n, e) . *private* key is (n, d) .

K_B^+

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,

$$c^d \bmod n = (m^e \bmod n)^d \bmod n$$

$$= m^{ed} \bmod n$$

$$= m^{(ed \bmod z)} \bmod n$$

$$= m^1 \bmod n$$

$$= m$$

RSA: another important property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$$

use public key
first, followed by
private key

use private key
first, followed by
public key

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