COS433/Math 473: Cryptography

Mark Zhandry

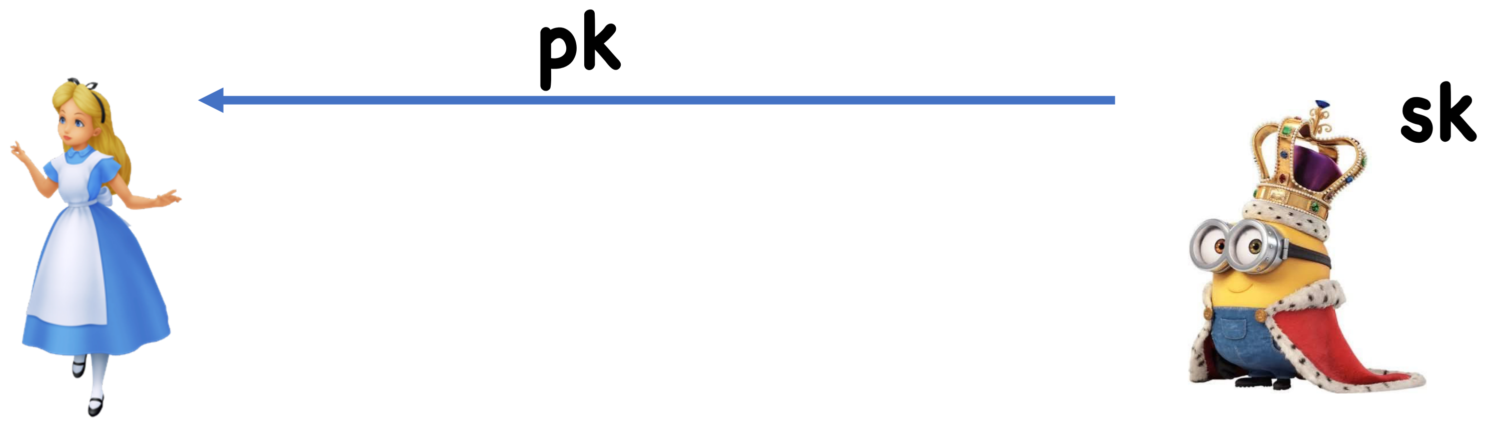
Princeton University

Spring 2017

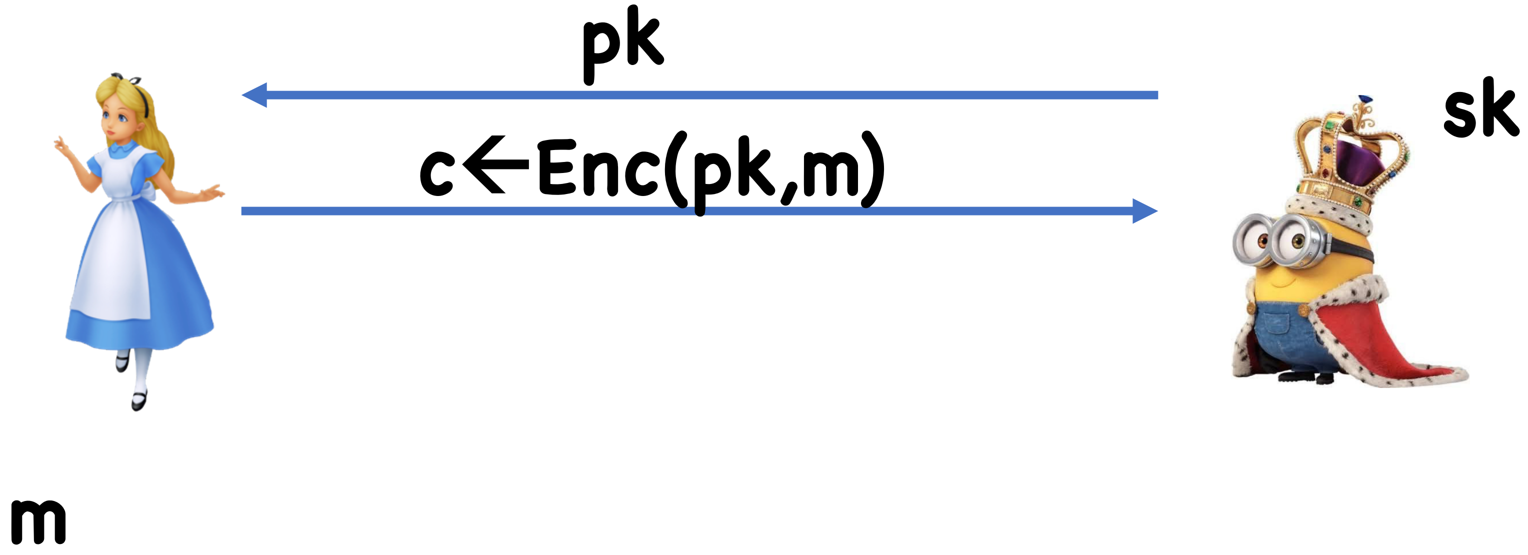
Previously

Exchanging keys and public key encryption

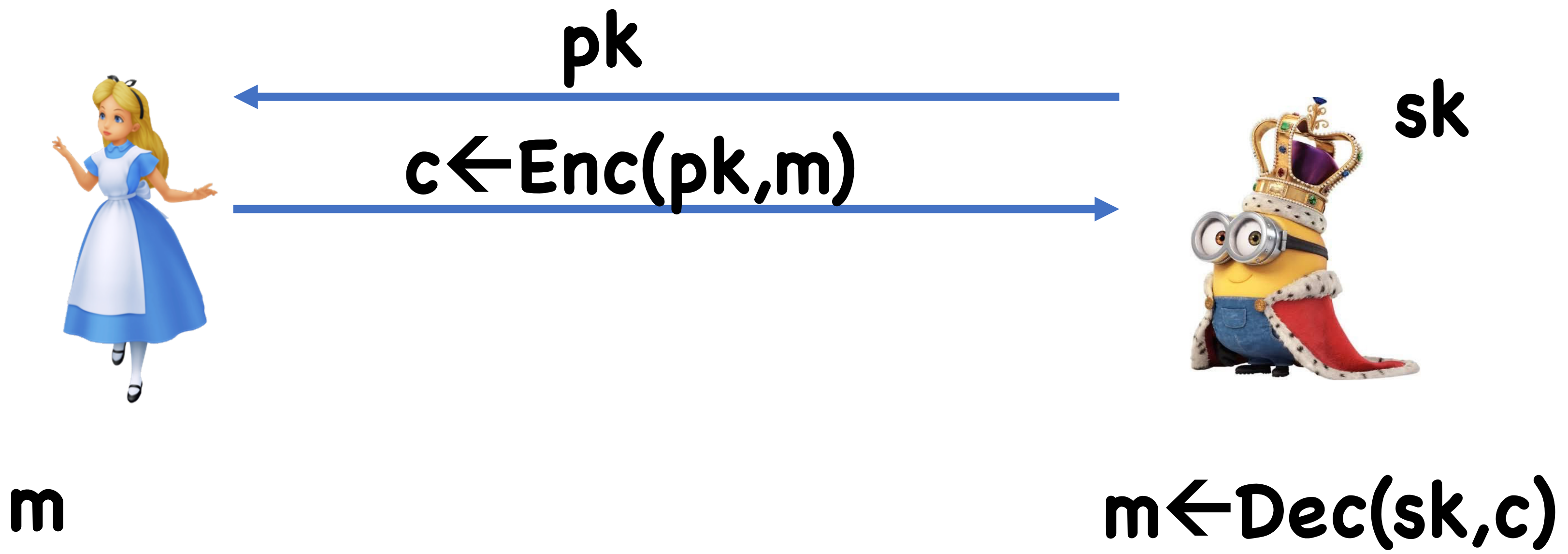
Public Key Encryption



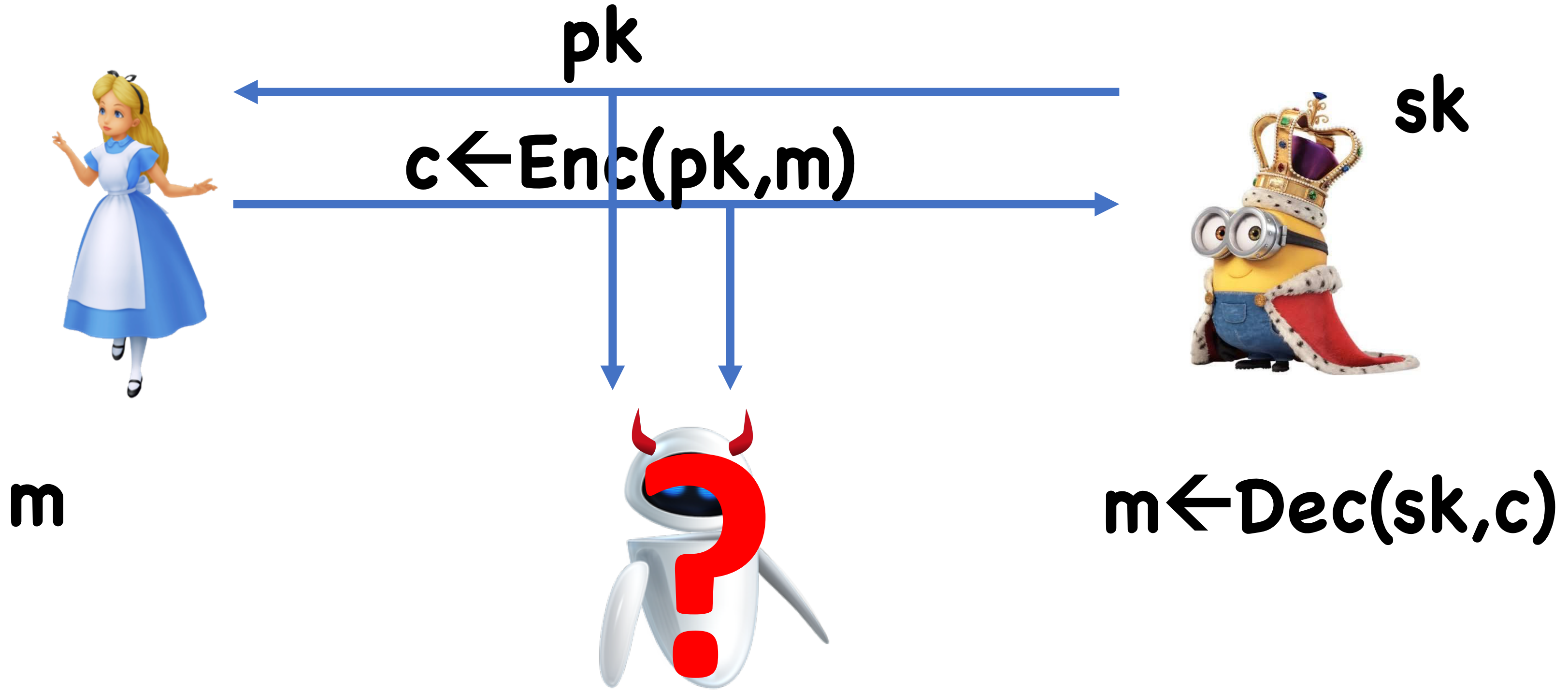
Public Key Encryption



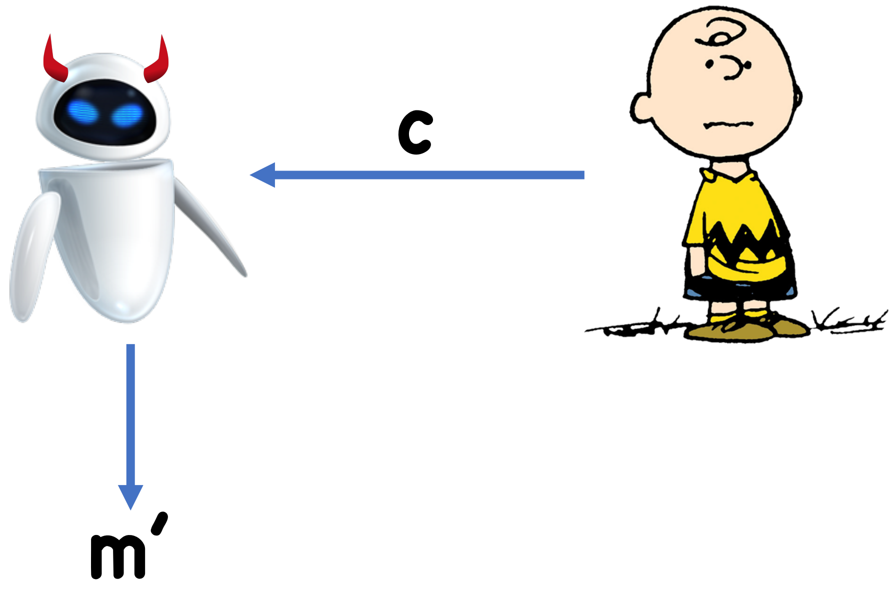
Public Key Encryption



Public Key Encryption

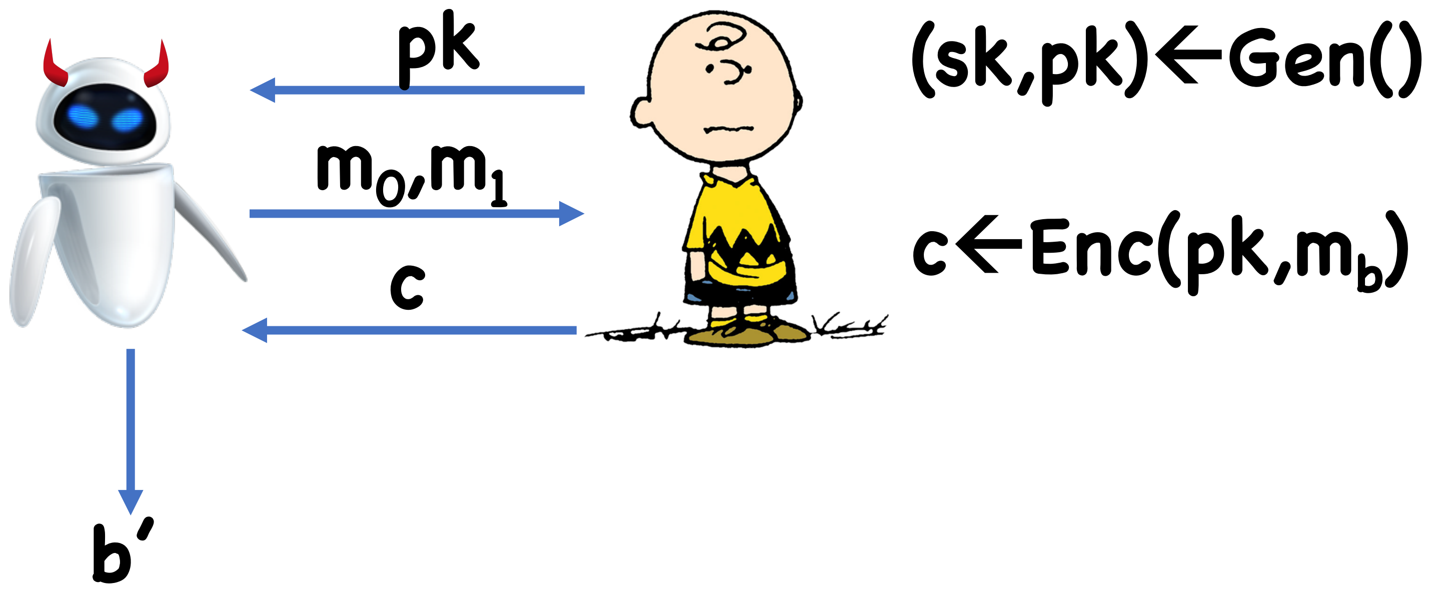


One-way Security

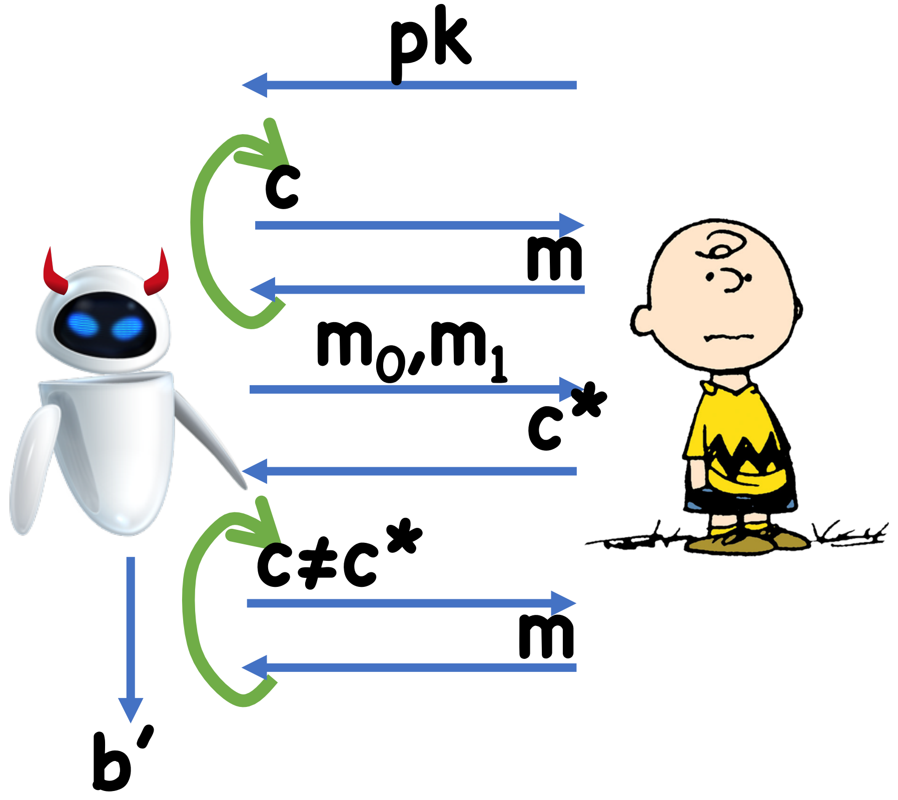


$(sk,pk) \leftarrow Gen()$
$m \leftarrow M$
$c \leftarrow Enc(pk,m)$

Semantic/CPA Security



CCA Security



(sk,pk)←Gen()

c←Enc(pk,m_b)

Trapdoor Permutations

Domain $\mathbf{X}$

Gen(): outputs **(pk,sk)**
$F(pk, x \in X) = y \in X$
$F^{-1}(sk, y) = x$

Correctness:
$\Pr[\ F^{-1}(sk, F(pk, x)) = x : (pk,sk) \leftarrow Gen()\ ] = 1$

Correctness implies F, F^{-1} are deterministic, permutations

Relaxation: Injective Trapdoor Functions

Domain **X**, range **Y**

Gen(): outputs **(pk,sk)**
F(pk,x∈X) = y∈Y, deterministic
F^{-1}(sk,y) = x

Correctness:
Pr[F^{-1}(sk, F(pk, x)) = x : (pk,sk)←Gen()] = 1

Correctness implies **F** is injective

CCA Secure PKE from TDPs

$\mathrm{Gen}_{PKE}() = \mathrm{Gen}()$
$\mathrm{Enc}_{PKE}(pk, m)$:

- Choose random **r**
- Let **$c_0 \leftarrow F(pk,r)$**
- Let **$c_1 \leftarrow \mathrm{Enc}_{SKE}(H(r), m)$**
- Output **(c_0,c_1)**

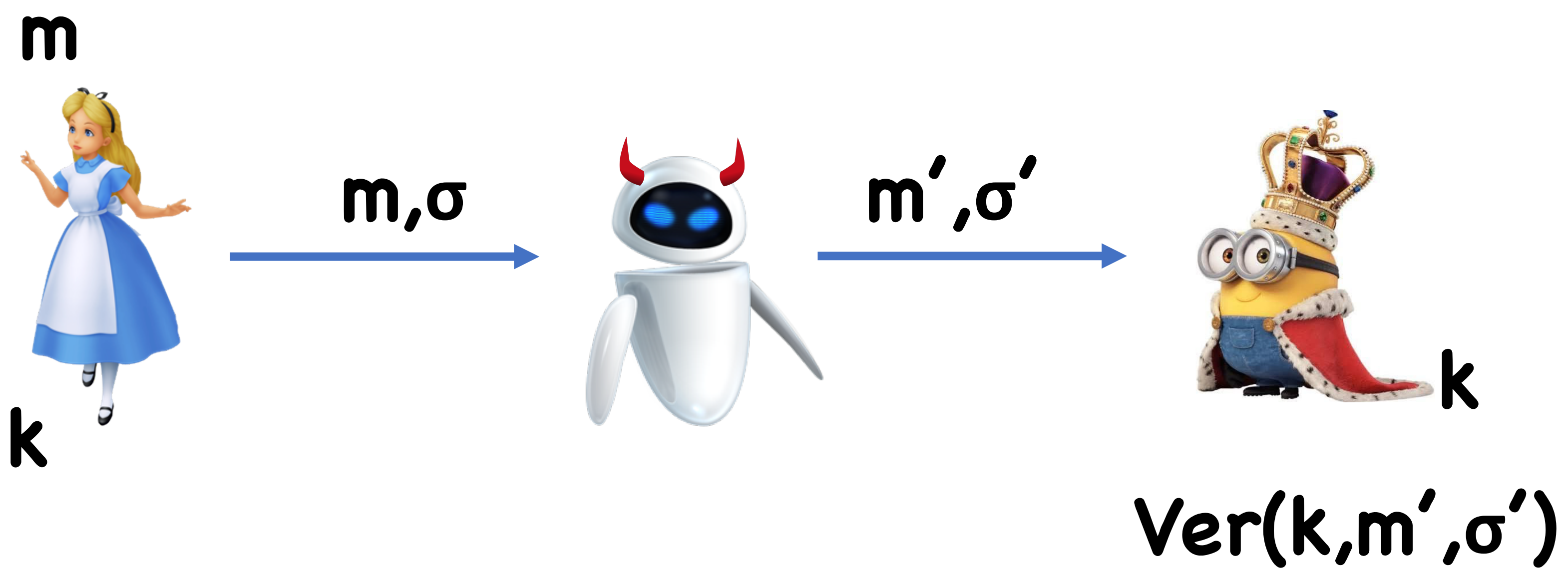
$\mathrm{Dec}_{PKE}(sk, (c_0, c_1))$:

- **Let $r \leftarrow F^{-1}(sk, c_0)$**
- **Let $m \leftarrow \mathrm{Dec}_{SKE}(H(r), c_1)$**

Today

Digital Signatures (aka public key MACs)

Message Authentication Codes



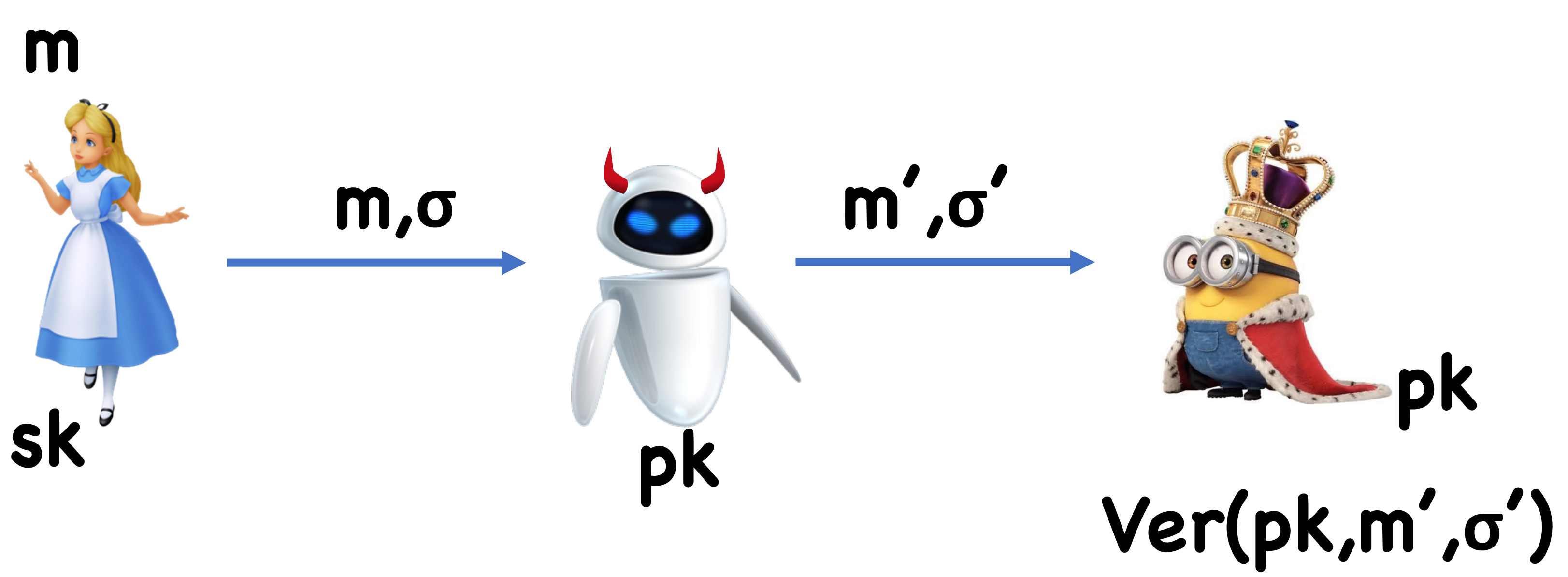
Goal: If Eve changed **m**, Bob should reject

Problem

What if Alice and Bob have never met before to exchange key **k**?

Want: a public key version of MACs where Bob can verify without having Alice's secret key

Message Integrity in Public Key Setting



Goal: If Eve changed **m**, Bob should reject

Digital Signatures

Algorithms:

- **Gen() → (sk,pk)**
- **Sign(sk,m) → σ**
- **Ver(pk,m,σ) → 0/1**

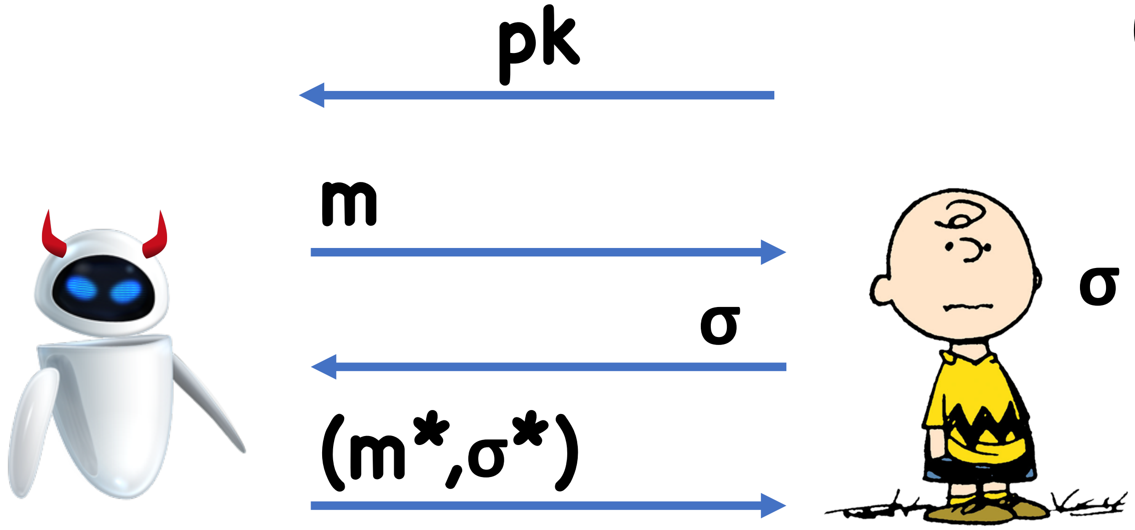
Correctness:

Pr[Ver(pk,m,Sign(sk,m))=1: (sk,pk)←Gen()] = 1

Security Notions?

Much the same as MACs, except adversary gets verification key

1-time Security For MACs



(sk,pk)←Gen()

σ ← Sign(sk,m)

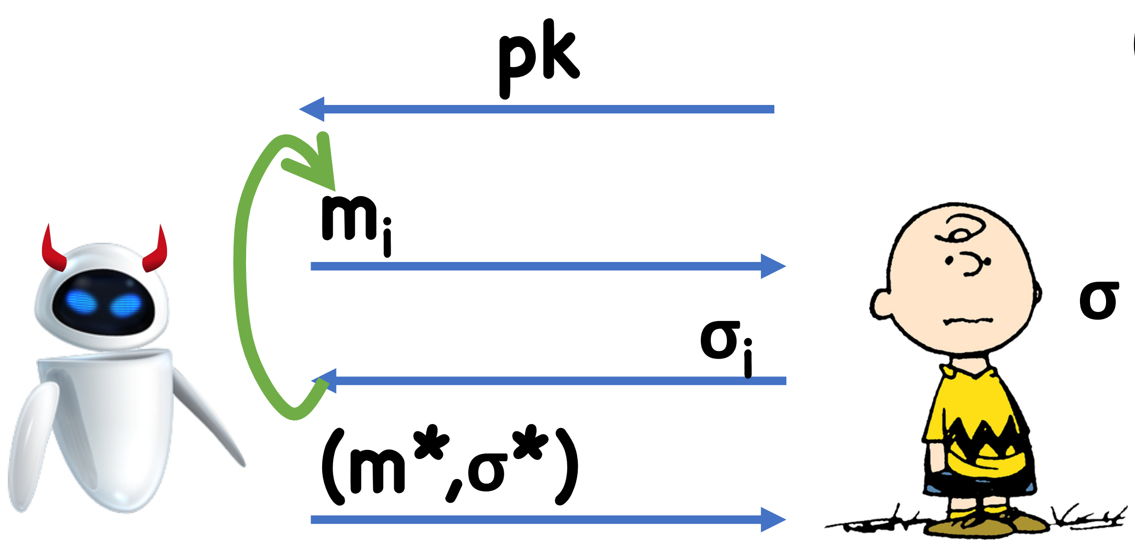
Output 1 iff:

- m*≠m
- Ver(pk,m*,σ*) = 1

1CMA-Adv( ) = Pr[outputs 1]

Unbounded Use Signatures

$(sk,pk) \leftarrow \mathrm{Gen}()$



$\sigma \leftarrow \mathrm{Sign}(sk,m)$

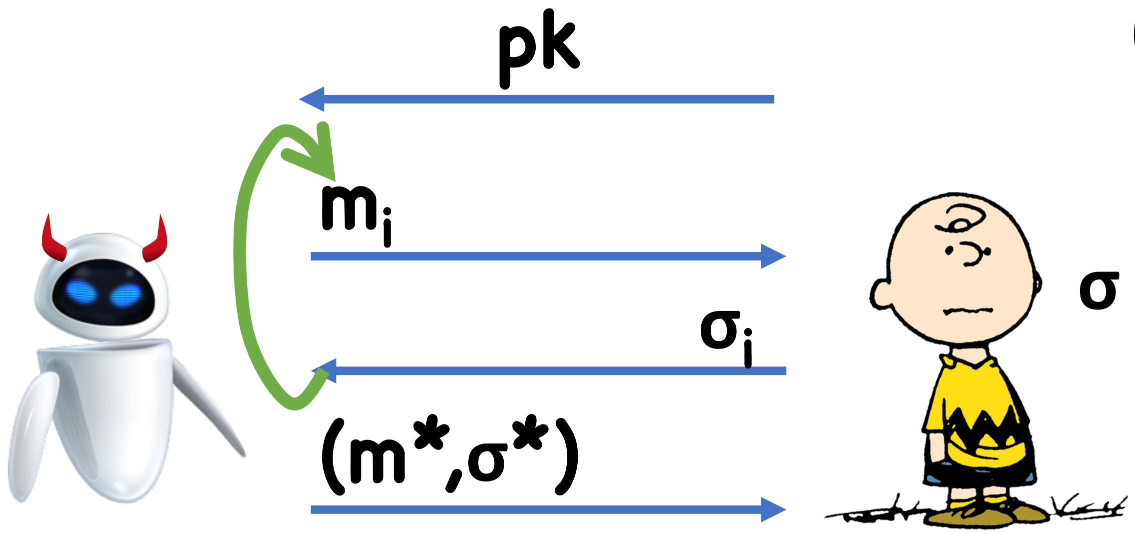
Output 1 iff:

- $m^* \notin \{m_1,\dots\}$
- $\mathrm{Ver}(pk,m^*,\sigma^*) = 1$

CMA-Adv( ) = Pr[outputs 1]

Strong Security

$(sk,pk) \leftarrow Gen()$



$\sigma \leftarrow Sign(sk,m)$

Output 1 iff:

- $(m^*, \sigma^*) \notin \{(m_1,\sigma_1)\ldots\}$
- $Ver(pk,m^*,\sigma^*) = 1$

CMA-Adv( ) = Pr[outputs 1]

Signatures from TDPs?

$\mathrm{Gen}_{\mathrm{Sig}}() = \mathrm{Gen}()$

$\mathrm{Sign}(sk,m) = F^{-1}(sk,m)$

$\mathrm{Ver}(pk,m,\sigma)$: $F(pk, \sigma) == m$

Signatures from TDPs

$\mathsf{Gen}_{\mathsf{Sig}}() = \mathsf{Gen}()$

$\mathsf{Sign}(\mathsf{sk},\mathsf{m}) = \mathsf{F}^{-1}(\mathsf{sk},\ \mathsf{H}(\mathsf{m})\ )$

$\mathsf{Ver}(\mathsf{pk},\mathsf{m},\sigma)\text{:}\ \ \mathsf{F}(\mathsf{pk},\ \sigma) == \mathsf{H}(\mathsf{m})$

Theorem: If $(\mathsf{Gen},\mathsf{F},\mathsf{F}^{-1})$ is a secure TDP, and H is modeled as a random oracle, then $(\mathsf{Gen}_{\mathsf{Sig}},\mathsf{Sign},\mathsf{Ver})$ is (strongly) CMA-secure

Proof Idea

Consider hypothetical adversary. Let (m^*,σ^*) be the forgery

Easy case: suppose adversary tries to always forge on the same message m^*

Notice: $F(pk,\sigma^*) = H(m^*)$

Therefore, adversary is inverting F on a random point, namely $y^* = H(m^*)$

Proof Idea

Consider hypothetical adversary. Let (m^*,σ^*) be the forgery

In general, adversary can choose m^* so that maybe $H(m^*)$ is easy to invert

However, adversary only sees a polynomial number of outputs of H, each output randomly chosen

- If adversary succeeds, such easy-to-invert outputs occur non-negligibly often

Proof Idea

Consider hypothetical adversary. Let **(m^*, σ^*)** be the forgery

Finishing touches:

- Adversary has signing oracle that may help him invert
- To remedy, can simulate **$H(m)=F(pk, H'(m))$**
- Now we can answer signing queries using **$H'(m)$**

Signatures from Injective TDFs?

What goes wrong?

Basic Rabin Signatures

Gen_{Sig}(): let **p,q** be random large primes
sk = (p,q), pk = N = pq

Sign(sk,m): Solve equation **σ^2 = H(m) mod N** using factors **p,q**
- Output **σ**

Ver(pk,m,σ): σ^2 mod N == H(m)

Problems

H(m) might not be a quadratic residue

Can only sign roughly ¼ of messages

Suppose adversary makes multiple signing queries on the same message

- Receives $\sigma_1, \sigma_2, \ldots$ such that $\sigma_i^2 \bmod N = H(m)$
- After enough tries, may get all 4 roots of **H(m)**
- Suppose $\sigma_1 \neq \pm\sigma_2 \bmod N$
- Then $\mathrm{GCD}(\sigma_1 - \sigma_2, N)$ will give a factor

One Solution

$\mathbf{Gen_{Sig}}$(): let **p,q** be primes, **a,b,c** s.t.

- **a** is a non-residue **mod p** and **q**,
- **b** is a residue **mod p** but not **q**,
- **c** is a residue **mod q** but not **p**

sk = (p,q,a,b,c), pk = (N = pq, a,b,c)

Sign(sk,m):

- Solve equation $\sigma^2 \in \{1,a,b,c\}\times H(m) \bmod N$
- Output σ

Ver(pk,m,σ): $\sigma^2 \bmod N \in \{1,a,b,c\}\times H(m)$

One Solution

Exactly one of $\{1,a,b,c\} \times H(m)$ is a residue $\bmod N$

$\Rightarrow$ Solution guaranteed to be found

Still have problem that multiple queries on same message will give different roots

One Solution

Possibilities:

- Have signer remember all messages signed

- Use a PRF to choose root deterministically

One Solution

Gen_{Sig}(): let **p,q** be primes, **a,b,c** …

sk = (p,q,a,b,c), pk = (N = pq, a,b,c)

Sign(sk,m):

- Solve equation **σ^2 = {1,a,b,c}×H(m) mod N,** where root is chosen according to **PRF(k, m)**
- Output **σ**

Ver(pk,m,σ): σ^2 mod N == {1,a,b,c}×H(m)

General Transformation

Let **(Gen,Sign,Ver)** be a (randomized) signature scheme that is secure as long as the adversary never queries the same message twice

Gen'(): run **(sk,pk)←Gen()**, let **k** a random PRF key
sk' = (sk,k), pk' = pk

Sign'(sk',m): Output **σ←Sign(sk,m; PRF(k,m))**
Ver' = Ver

Theorem: If **(Gen,Sign,Ver)** is secure when no message is queried twice, then **(Gen',Sign',Ver')** is CMA-secure

Proof Sketch:

- Can assume wlog that adversary never queries the same message twice
  - Would have received the same answer anyway
- Because using PRF, signatures look like they used fresh randomness

One Solution

Possibilities:

- Have signer remember all messages signed

- Use a PRF to choose root deterministically

- Choose root that is itself a quadratic residue
  (if **-1** is not a residue mod **p,q**,
  there will be exactly one)

Another Solution

$\mathbf{Gen}_{\mathbf{Sig}}$(): let **p,q** be random large primes
sk = (p,q), pk = N = pq

Sign(sk,m): Repeat until successful:

- Choose random $u \leftarrow \{0,1\}^\lambda$
- Solve equation $\sigma^2 = H(m,u) \bmod N$
- Output (u,σ)

Ver(pk,m,(u,σ)): $\sigma^2 \bmod N == H(m,u)$

Another Solution

In expectation, after 4 tries will have success

(Whp) Only ever get a single root of a given **H(m,u)**

Theorem: If factoring is hard and **H** is modeled as a random oracle, then Rabin signatures are (weakly) CMA secure

Another Solution

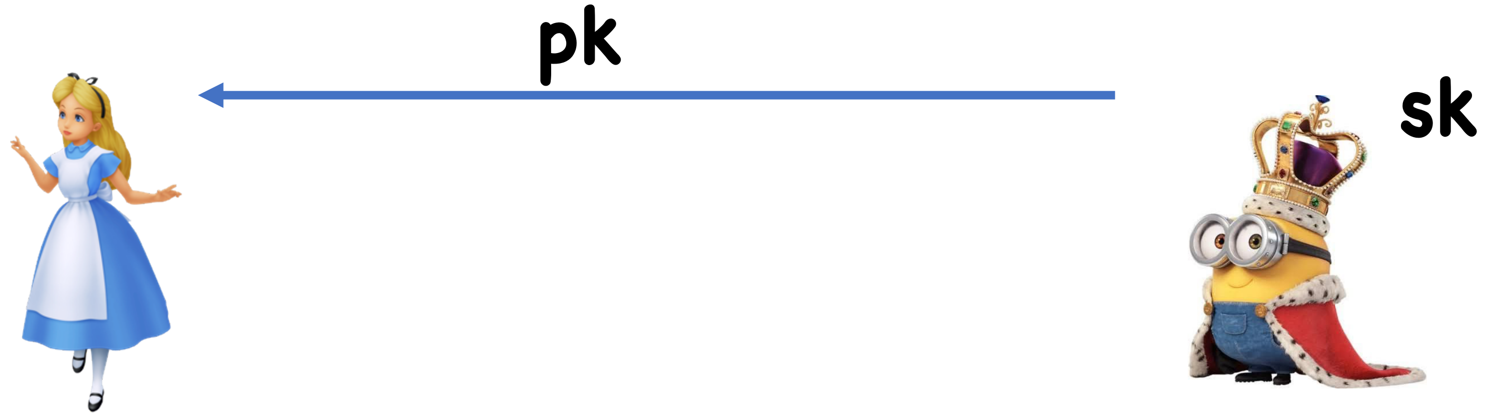
Sign(sk,m): Repeat until successful:

- Choose random $\mathbf{u} \leftarrow \{0,1\}^\lambda$
- Solve equation $\sigma^2 = H(m,u) \bmod N$ using factors p,q, where $\sigma<(N-1)/2$
- Output (u,σ)

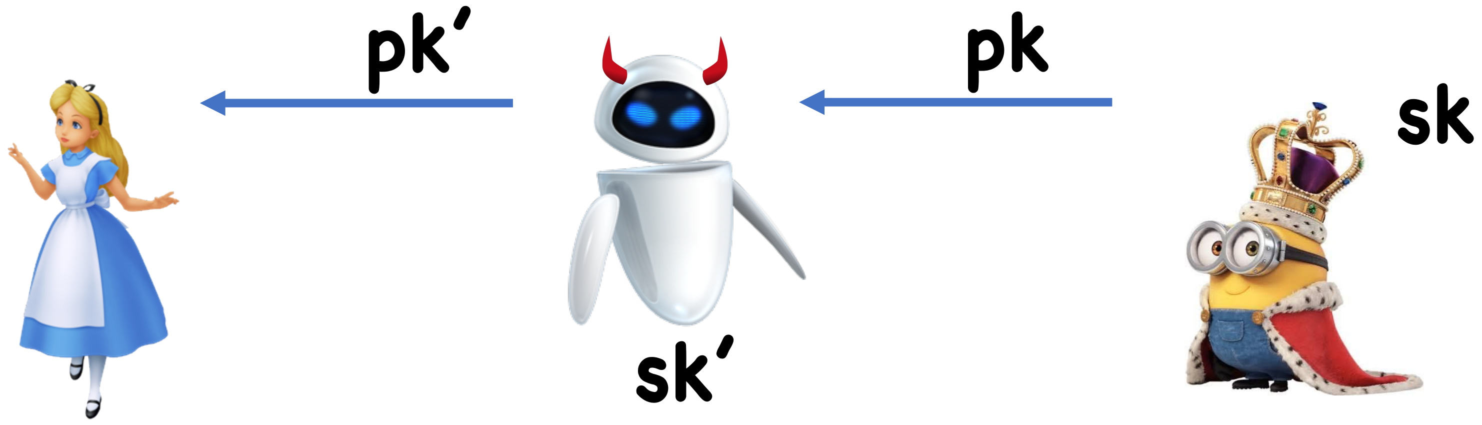
Ver(pk,m,(u,σ)): $\sigma^2 \bmod N == H(m,u) \wedge \sigma<(N-1)/2$

Theorem: If factoring is hard and **H** is modeled as a random oracle, then Rabin signatures are strongly CMA secure

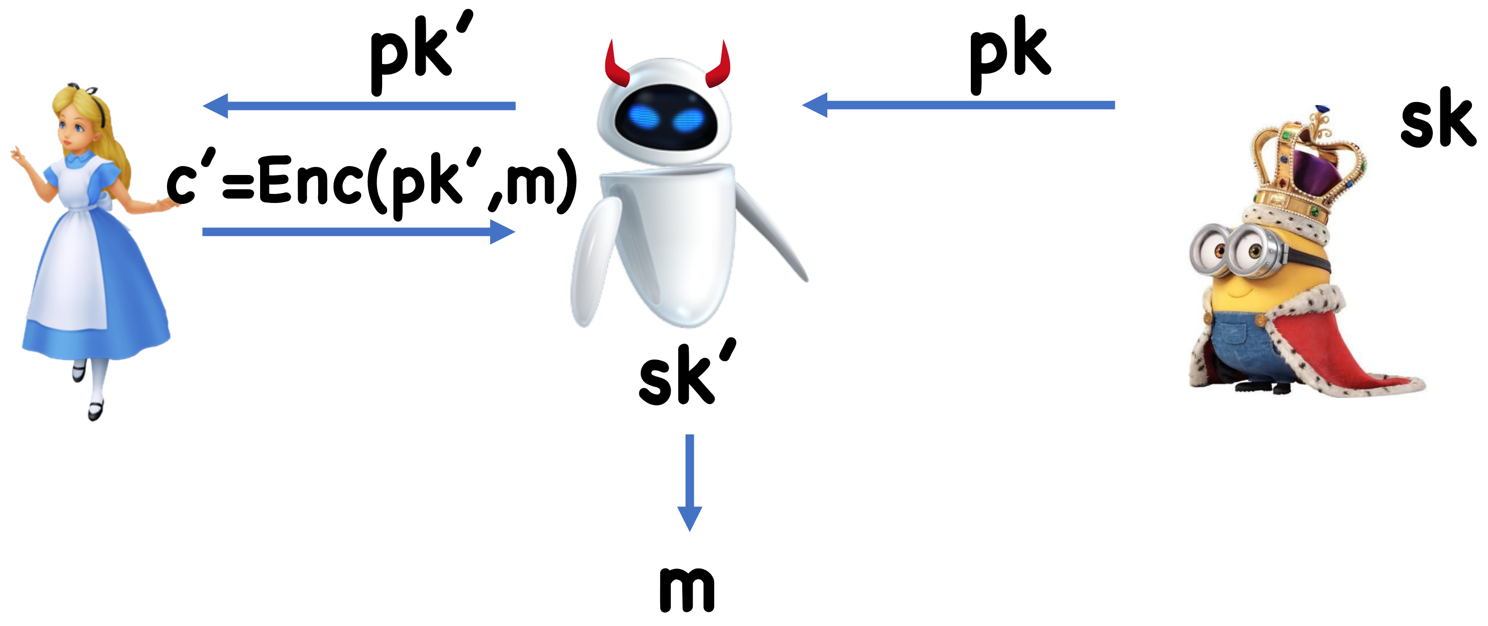
Digital Signatures and the Public Key Infrastructure



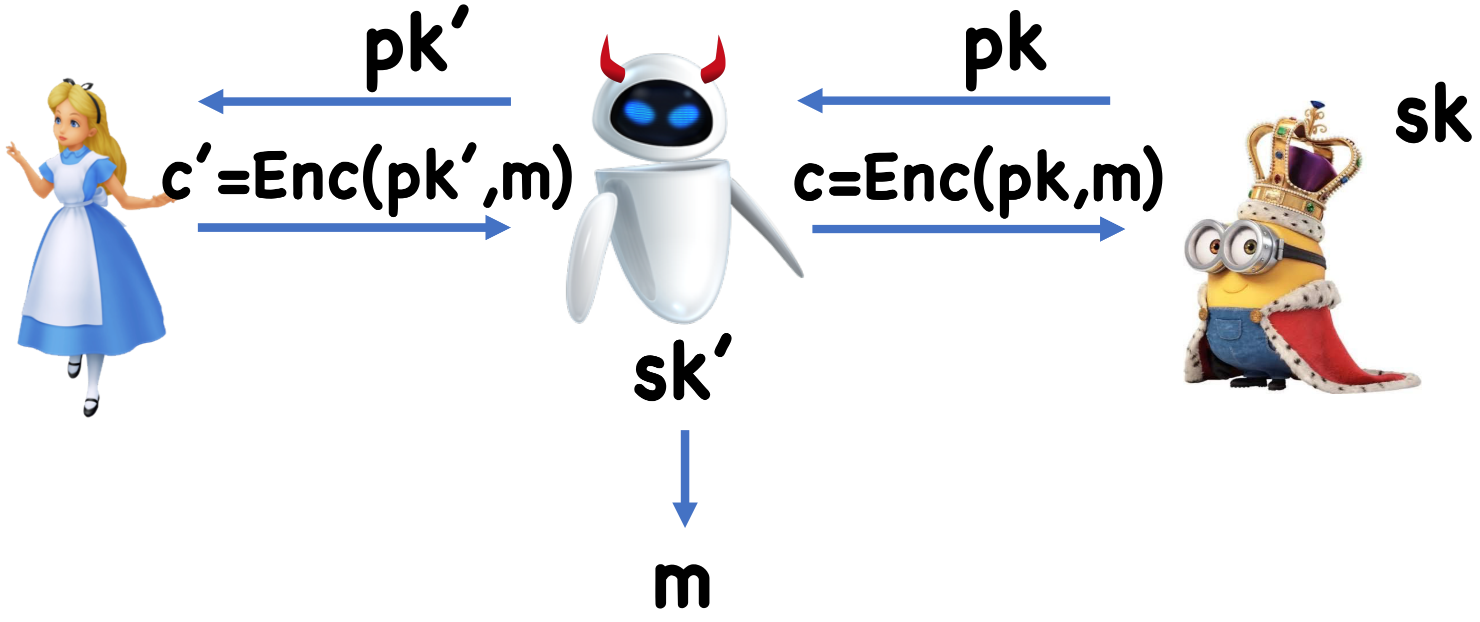
Digital Signatures and the Public Key Infrastructure



Digital Signatures and the Public Key Infrastructure



Digital Signatures and the Public Key Infrastructure



Takeaway

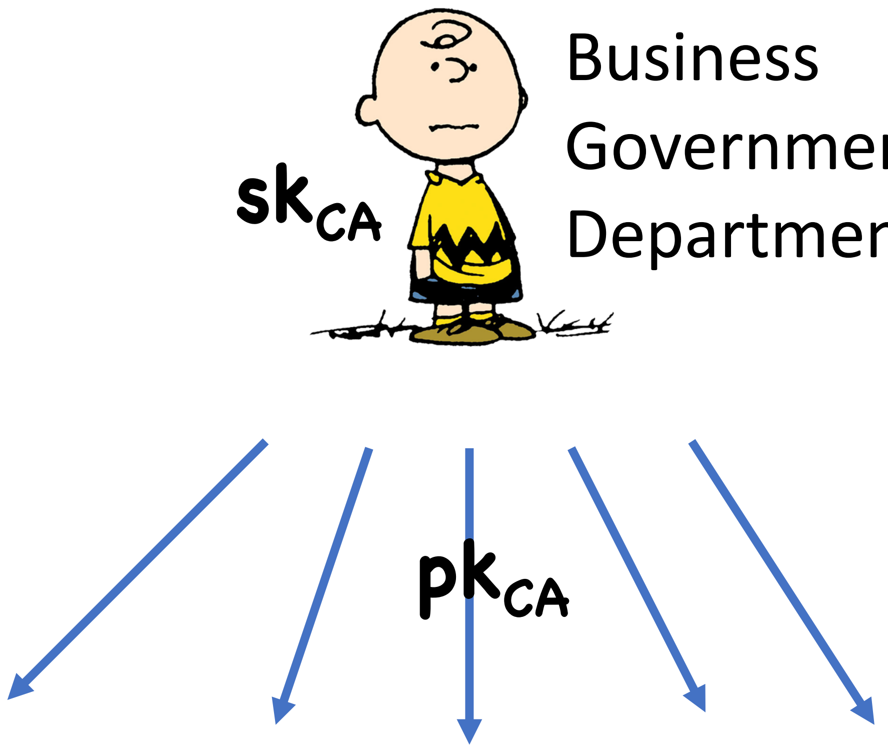
Need some authenticated channel to ensure distribution of public keys

But how to authenticate channel in the first place without being able to distribute public keys?

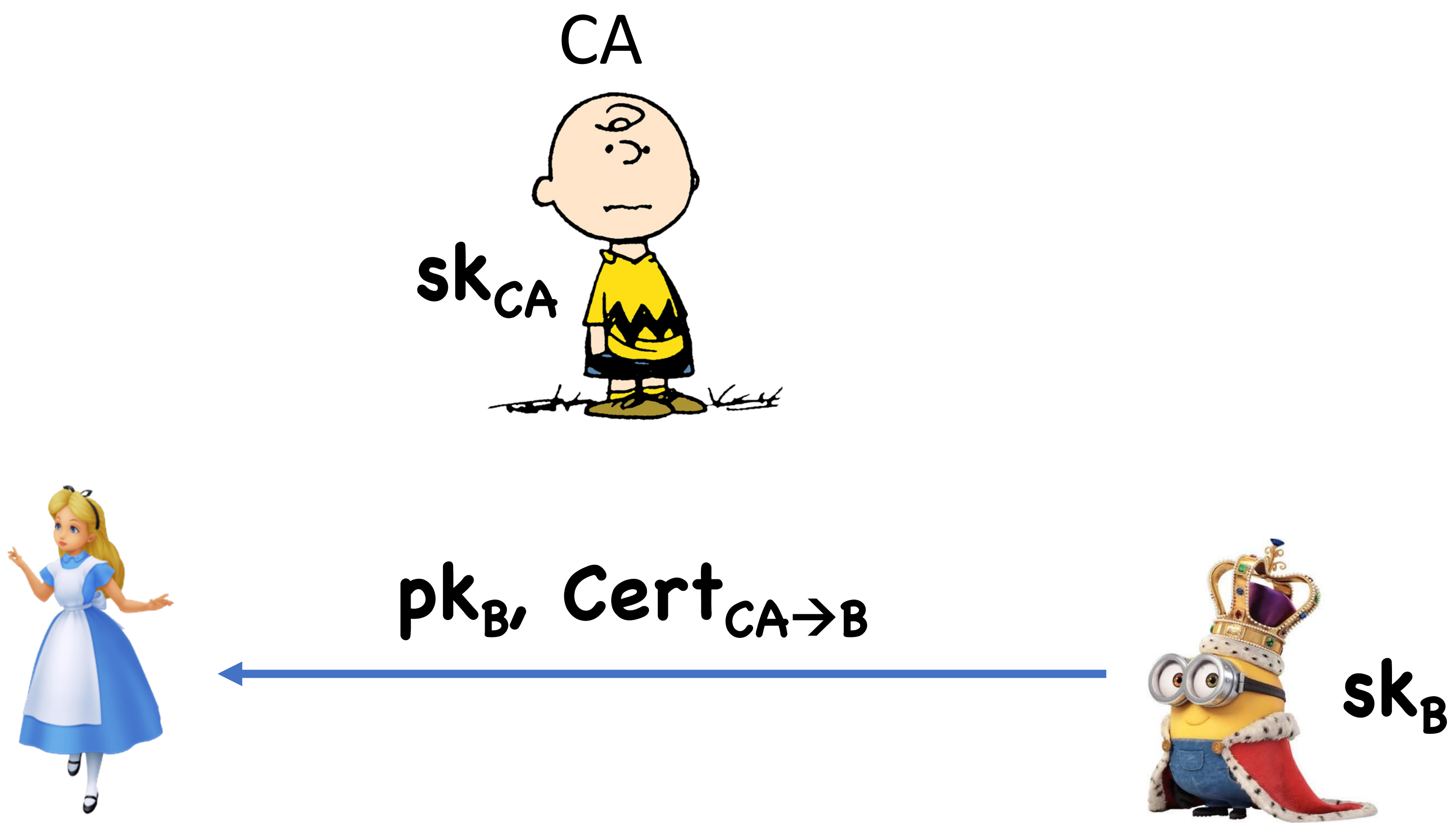
Solution: Certificate Authorities

CA

Business
Government Agency
Department within company



Solution: Certificate Authorities



$\mathbf{Cert}_{CA\rightarrow B}$=$\mathbf{Sign}(\mathbf{sk}_{CA}$, "Bob's public key is $\mathbf{pk}_B$")

Solution: Certificate Authorities

Bob is typically some website

- Obtains **Cert** by, say, sending someone in person to CA with $\mathbf{pk_B}$
- Only needs to be done once

If Alice trusts CA, then Alice will be convinced that $\mathbf{pk_B}$ belongs to Bob

Alice typically gets $\mathbf{pk_{CA}}$ bundled in browser

Limitations

Everyone must trust same CA

- May have different standards for issuing certs

Single point of failure: if $\mathbf{sk}_{CA}$ is compromised, whole system is compromised

Single CA must handle all verification

Solutions?

Multiple CAs

There are actually many CA’s, CA_1, CA_2,…

Bob obtains cert from all of them, sends all the certs with his public key

As long as Alice trusts one of the CA’s, she will be convinced about Bob’s public key

Certificate Chaining

CA issues $\mathbf{Cert}_{CA \rightarrow B}$ for Bob

Bob can now use his signing key to issue $\mathbf{Cert}_{B \rightarrow D}$ to Donald

Donald can now prove his public key by sending $(\mathbf{Cert}_{CA \rightarrow B}, \mathbf{Cert}_{B \rightarrow D})$

- Proves that CA authenticated Bob, and Bob authenticated Donald

Certificate Chaining

For Bob to issue his own certificates, a standard cert should be insufficient

- CA knows who Bob is, but does not trust him to issue certs on its behalf

Therefore, Bob should have a stronger cert:

$\mathbf{Cert}_{\mathbf{CA\rightarrow B}}\mathbf{=Sign(sk_{CA},}$"Bob's public key is $\mathbf{pk_B}$ and he can issue certificates on behalf of CA"$\mathbf{)}$

Certificate Chaining

One root CA

Many second level CAs CA_1, CA_2,…

- Each has $\mathbf{Cert}_{CA \rightarrow Ca_i}$

Advantage: eases burden on root

Disadvantage: now multiple points of failure

Web of Trust Model

Anyone can issue certs for anyone else

- Each user can decide who to trust, and only accept certificates from people they trust

Public keys and Certs distributed at “key-signing parties” (e.g. conferences)

- May not know other person, but can verify identify by looking at driver’s license, etc

Invalidating Certificates

Sometimes, need to invalidate certificates

- Private key stolen
- User leaves company
- Etc

Options:

- Expiration
- Explicit revocation

Next Time

Digital Signatures in the standard model from one-way functions