COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2017

Previously…

Digital Signatures

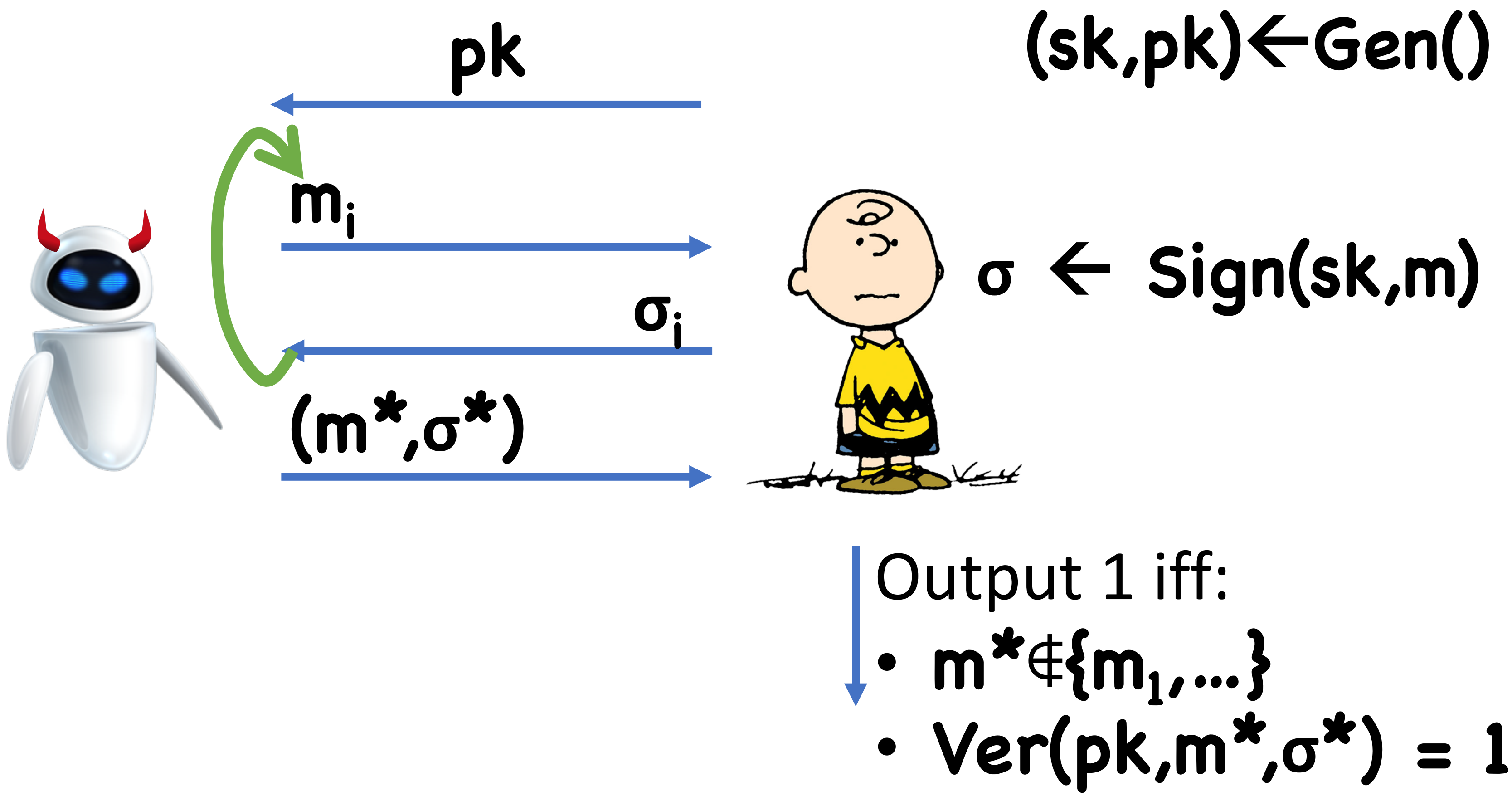
Algorithms:

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

Correctness:

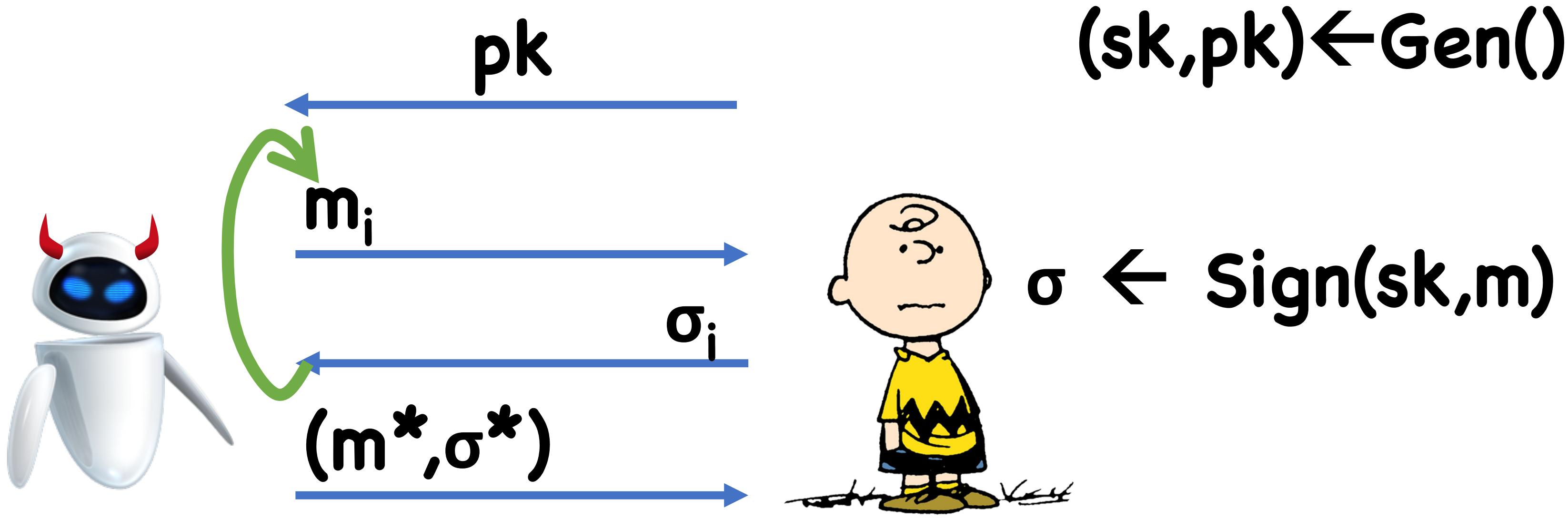
Pr[Ver(pk,m,Sign(sk,m))=1: (sk,pk)←Gen()] = 1

Many-time Signatures



CMA-Adv() = Pr[ outputs 1]

Strong Security



Output 1 iff:

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

CMA-Adv() = Pr[ outputs 1]

Signatures from TDPs

$\mathbf{Gen_{Sig}() = Gen()}$

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

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

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

Basic Rabin Signatures

$\mathbf{Gen_{Sig}}$(): let **p,q** be random large primes

$$sk = (p,q),\ pk = N = pq$$

Sign(sk,m): Solve equation $\sigma^2 = H(m) \bmod N$ using factors **p,q**

- Output σ

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

Signatures from One-way Functions

One-way functions are sufficient to build signature schemes

Therefore, can build signatures from:

- RSA, DDH, Block Ciphers, CRHF, etc.

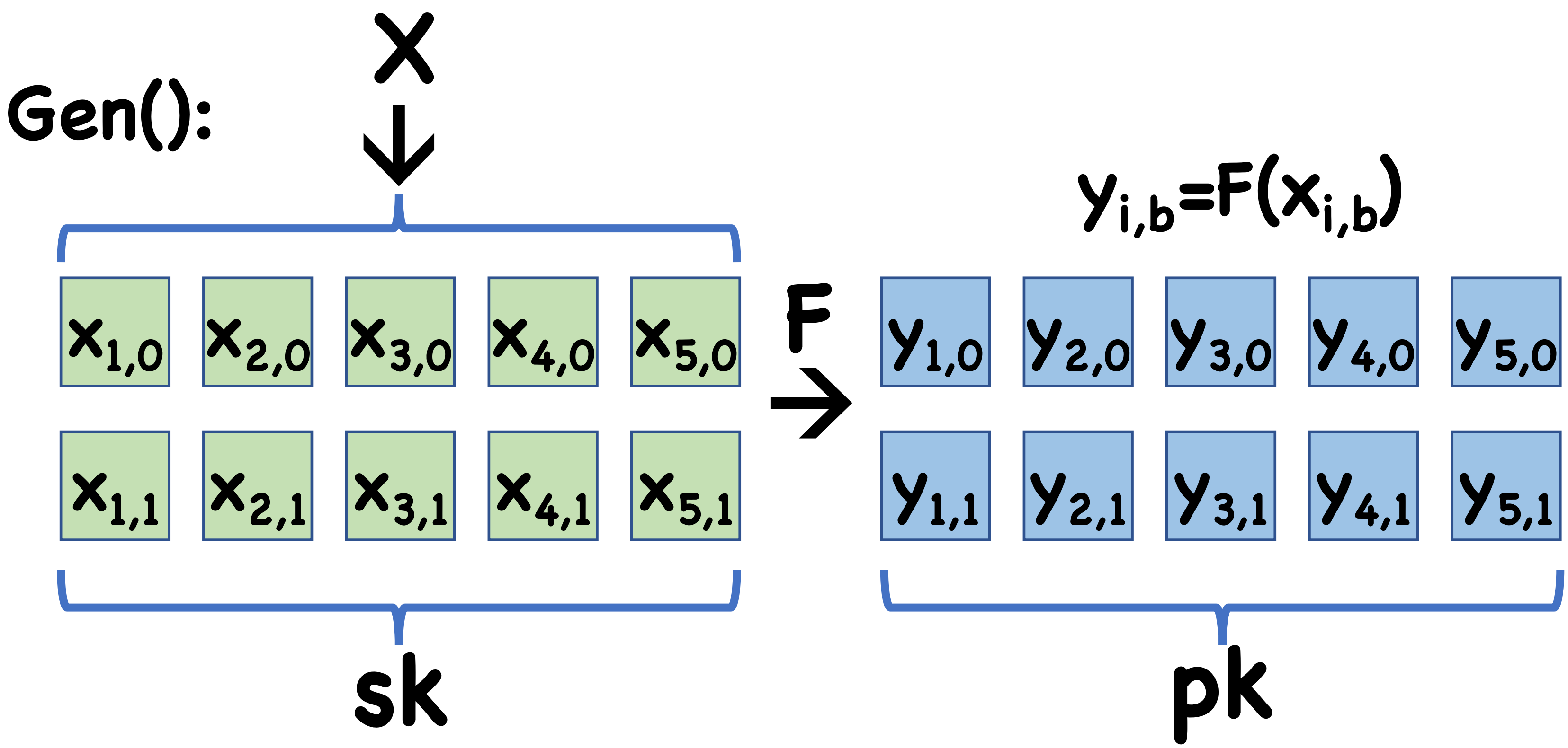
Limitation:

- Poor performance in practice

Lamport Signatures

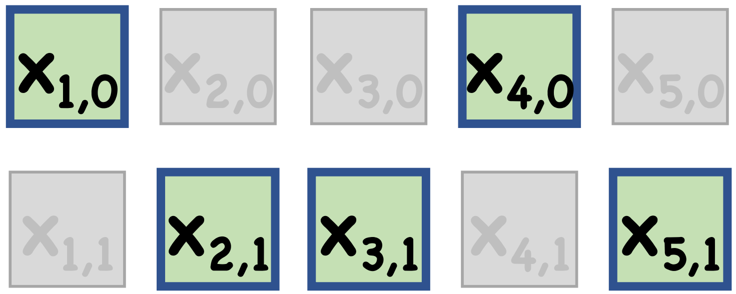
Let $F: X \rightarrow Y$ be a one-way function

Let $M = \{0,1\}^n$ be message space

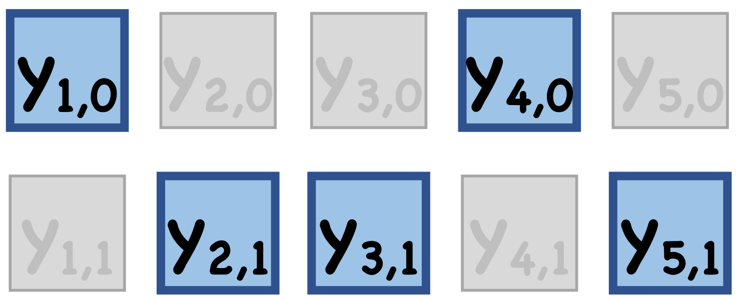


Lamport Signatures

Sign(sk, m): $(x_{i,m_i})_{i=1,\ldots,n}$



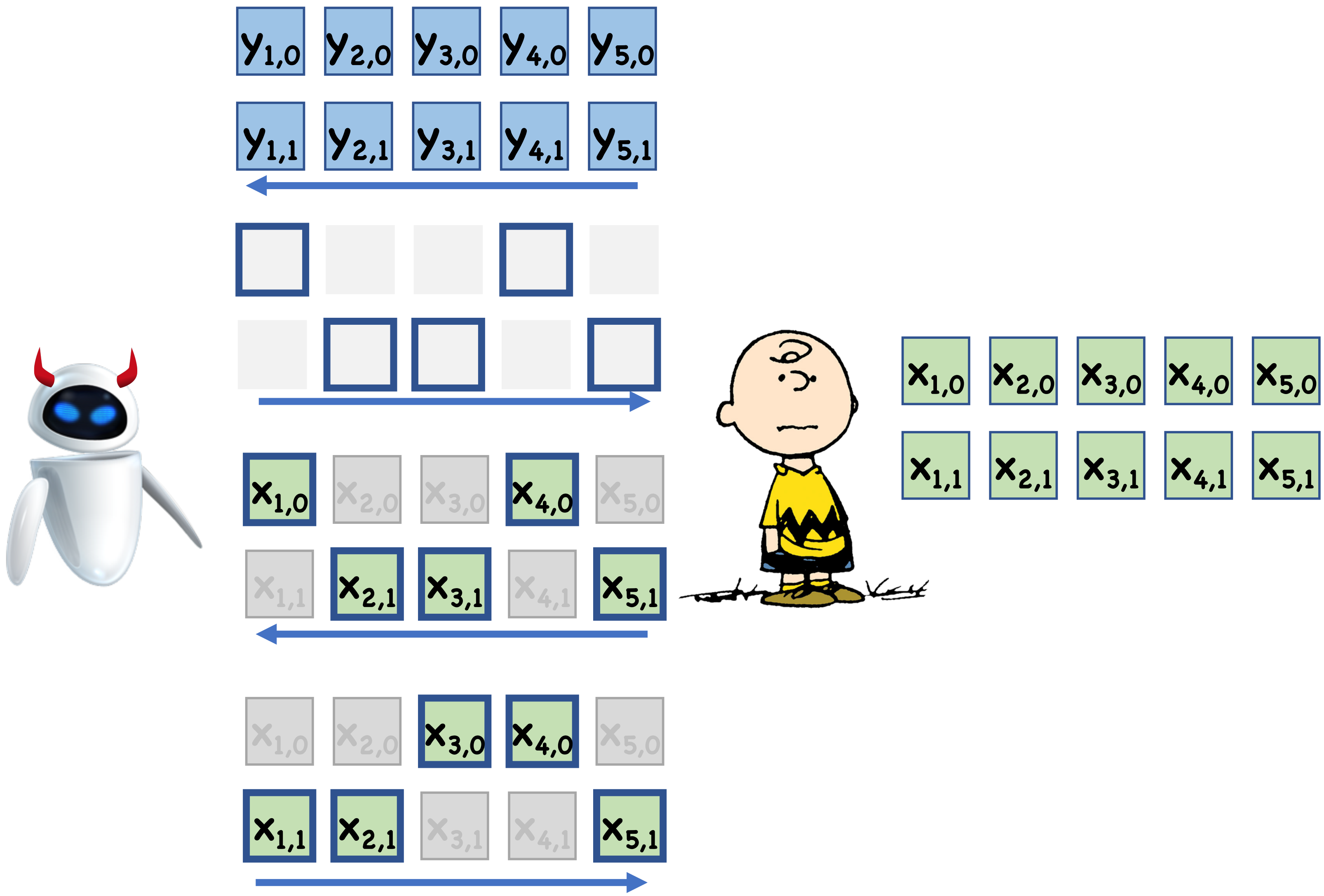
Ver(pk,m,σ): $F(x_{i,m_i}) = y_{i,m_i}$



Lamport Signatures

Theorem: If **F** is a secure OWF, then **(Gen,Sign,Ver)** is a (weakly) secure one-time signature scheme

Proof



$y_{1,0}$ $y_{2,0}$ $y_{3,0}$ $y_{4,0}$ $y_{5,0}$
$y_{1,1}$ $y_{2,1}$ $y_{3,1}$ $y_{4,1}$ $y_{5,1}$
$x_{1,0}$ $x_{2,0}$ $x_{3,0}$ $x_{4,0}$ $x_{5,0}$
$x_{1,1}$ $x_{2,1}$ $x_{3,1}$ $x_{4,1}$ $x_{5,1}$
$x_{1,0}$ $x_{2,0}$ $x_{3,0}$ $x_{4,0}$ $x_{5,0}$
$x_{1,1}$ $x_{2,1}$ $x_{3,1}$ $x_{4,1}$ $x_{5,1}$
$x_{1,0}$ $x_{2,0}$ $x_{3,0}$ $x_{4,0}$ $x_{5,0}$
$x_{1,1}$ $x_{2,1}$ $x_{3,1}$ $x_{4,1}$ $x_{5,1}$

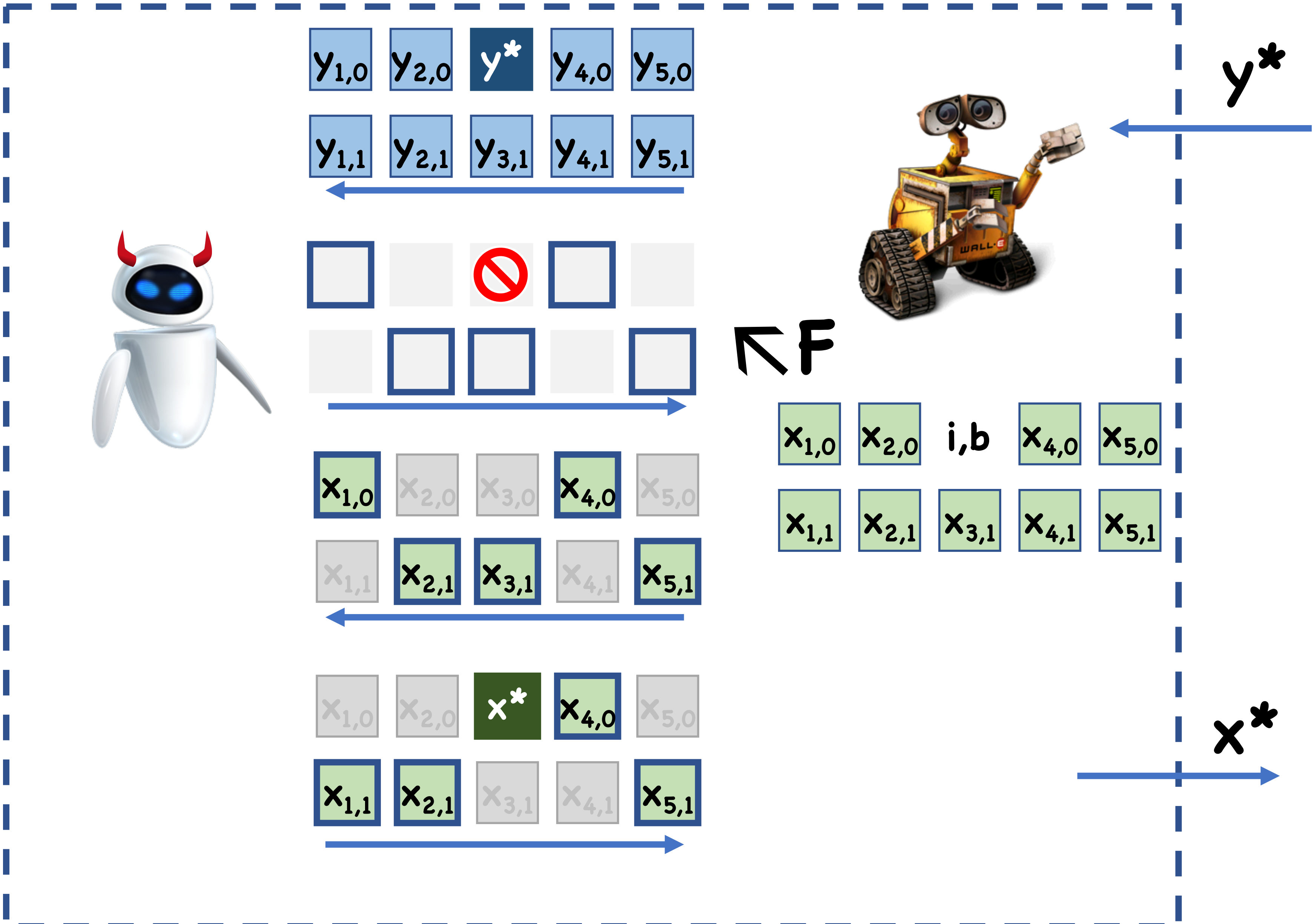
Proof

Since $m^* \neq m$, $\exists\, i$ s.t. $m^*_i \neq m_i$

Suppose we know i, $m_i = 1-b$, $m^*_i = b$

Construct adversary that inverts OWF

Proof



$y_{1,0}$ $y_{2,0}$ y^* $y_{4,0}$ $y_{5,0}$
$y_{1,1}$ $y_{2,1}$ $y_{3,1}$ $y_{4,1}$ $y_{5,1}$
y^*
F
$x_{1,0}$ $x_{2,0}$ i,b $x_{4,0}$ $x_{5,0}$
$x_{1,1}$ $x_{2,1}$ $x_{3,1}$ $x_{4,1}$ $x_{5,1}$
$x_{1,0}$ $x_{2,0}$ $x_{3,0}$ $x_{4,0}$ $x_{5,0}$
$x_{1,1}$ $x_{2,1}$ $x_{3,1}$ $x_{4,1}$ $x_{5,1}$
$x_{1,0}$ $x_{2,0}$ x^* $x_{4,0}$ $x_{5,0}$
$x_{1,1}$ $x_{2,1}$ $x_{3,1}$ $x_{4,1}$ $x_{5,1}$
x^*
WALL·E

Proof

View of  exactly as in 1-time CMA experiment, assuming

- **i**th bit of **m = b**
- **i**th bit of **m* = 1-b**

If  always chooses **m,m*** with these properties, and forges with probability **ε**, then  inverts with probability **ε**

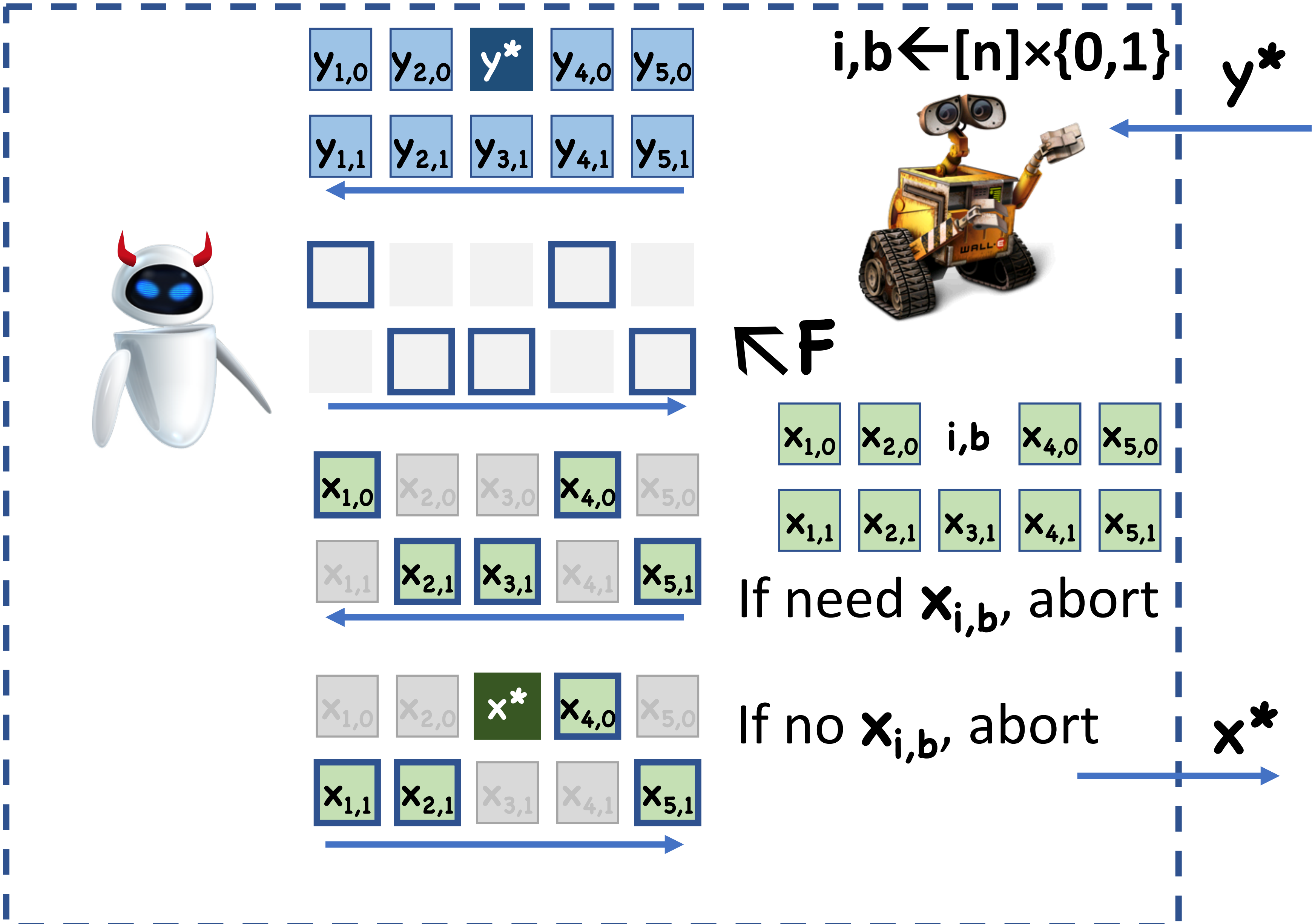
Proof

In general,  may choose **m,m*** to differ at arbitrary places

- May be randomly chosen, may depend on **pk**, may even depend on **σ**
- May never be at certain places

How do we make  still succeed?

Proof



y1,0 y2,0 y* y4,0 y5,0
y1,1 y2,1 y3,1 y4,1 y5,1
i,b←[n]×{0,1}
y*
F
x1,0 x2,0 i,b x4,0 x5,0
x1,1 x2,1 x3,1 x4,1 x5,1
x1,0 x2,0 x3,0 x4,0 x5,0
x1,1 x2,1 x3,1 x4,1 x5,1
If need xi,b, abort
x1,0 x2,0 x* x4,0 x5,0
x1,1 x2,1 x3,1 x4,1 x5,1
If no xi,b, abort
x*

Proof

pk independent of **(i,b)**
- **m** independent of **(i,b)**
- Therefore, $\Pr[m_i=1-b]=½$

Conditioned on $m_i=1-b$,
- Signing succeeds
- **σ** independent of **i**
-  forges with probability **ε**, independent of **i**

Proof

We know if  forges, then $\mathbf{m^* \neq m}$

Since $\mathbf{m^*}$ independent of $\mathbf{i}$, have prob at least $\mathbf{1/n}$ that $\mathbf{m^*_i = 1 - m_i = b}$

In this case,  succeeds in inverting $\mathbf{y^*}$

- Prob = $\frac{1}{2} \times \varepsilon \times 1/n = \varepsilon/2n$

Limitations of Lamport Signatures

Only weakly secure

- Why?
- How to fix?

$|pk|, |\sigma| >> |m|$

- How to fix?

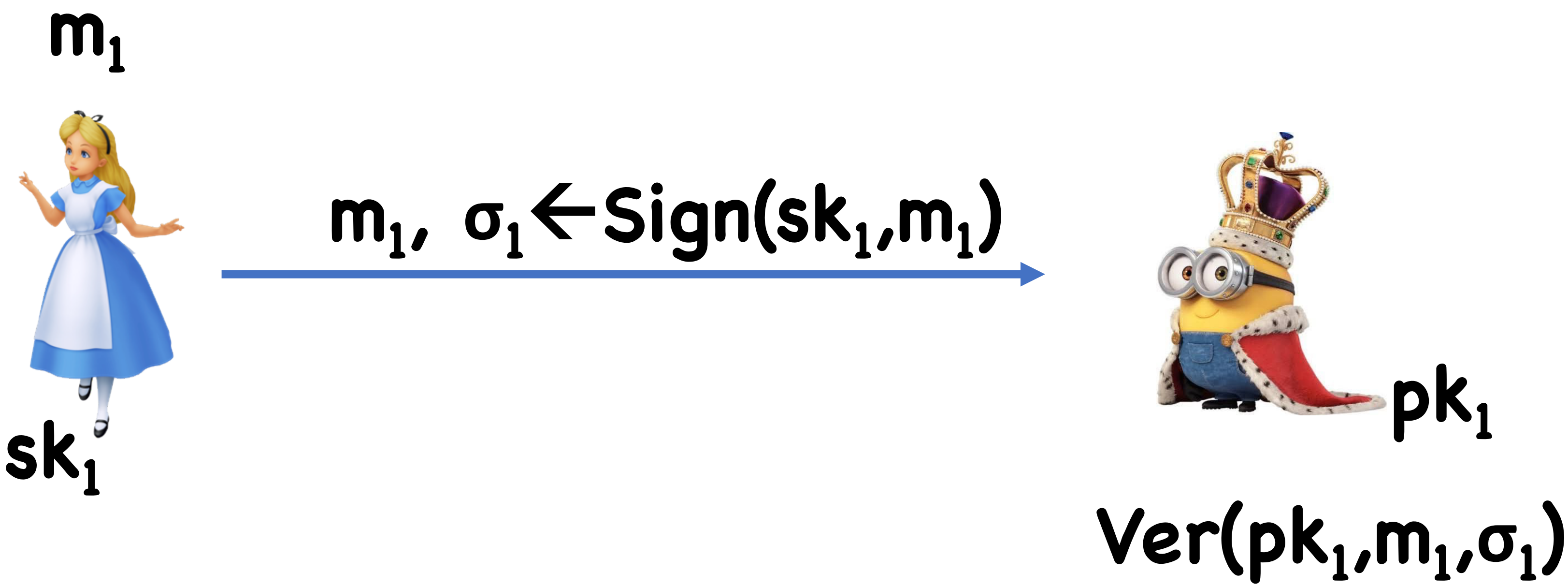
Theorem: Given a secure OWF, it is possible to construct a strongly secure 1-time signature scheme where $|m| \gg |pk|,|\sigma|$

Signing Multiple Messages

Once adversary sees two signed messages, security is lost (why?)

How do we sign multiple messages?

Signature Chaining



Signature Chaining

m_1

$m_1,\ \sigma_1 = (pk_2, \sigma_1')$

$\sigma_1' \leftarrow \mathrm{Sign}(sk_1,\ (m_1, pk_2)\ )$

pk_1

sk_1

$(sk_2, pk_2) \leftarrow \mathrm{Gen}()$

$\mathrm{Ver}(pk_1, (m_1, pk_2)\ , \sigma_1')$

Signature Chaining

m_2

sk_1

$(sk_2, pk_2) \leftarrow Gen()$

$m_2,\ \sigma_2$

$\sigma_1' \leftarrow Sign(sk_2,\ m_2)$

pk_1

pk_2

$Ver(pk_2, m_2, \sigma_2)$

Signature Chaining

Idea: Bob can be assured that $\mathbf{pk_2}$ was in fact generated by Alice

- If Eve tampered with $\mathbf{pk_2}$, then signature on first message would have been invalid

Therefore, Alice can sign $\mathbf{m_2}$ using $\mathbf{sk_2}$, and Eve cannot produce a forgery $\mathbf{m_2'}$ with valid signature

Can repeat process to sign arbitrarily many messages

Signature Chaining

m_2

$m_2,\ \sigma_2 = (pk_3, \sigma_2')$

$\sigma_1' \leftarrow \mathrm{Sign}(sk_2,\ (m_2, pk_3)\ )$

sk_1

pk_1

pk_2

$(sk_2, pk_2) \leftarrow \mathrm{Gen}()$

$(sk_3, pk_3) \leftarrow \mathrm{Gen}()$

$\mathrm{Ver}(pk_2, (m_2, pk_3)\ , \sigma_2')$

Limitations

Alice and Bob must stay synchronized

- Else, Bob won’t be using correct public key to verify

If many users, every pair needs to be syncronized

- What if Alice is sending messages to Bob and Charlie?

(Almost) Stateless Signature Chaining

m_2

sk_1

$m_2,\ \sigma_2 = (m_1, pk_2, \sigma_1', pk_3, \sigma_2')$

$\sigma_1' \leftarrow \mathrm{Sign}(sk_2,\ (m_2, pk_3)\ )$

pk_1

$(sk_2, pk_2) \leftarrow \mathrm{Gen}()$

$(sk_3, pk_3) \leftarrow \mathrm{Gen}()$

$\mathrm{Ver}(pk_1, (m_1, pk_2)\ , \sigma_1')$

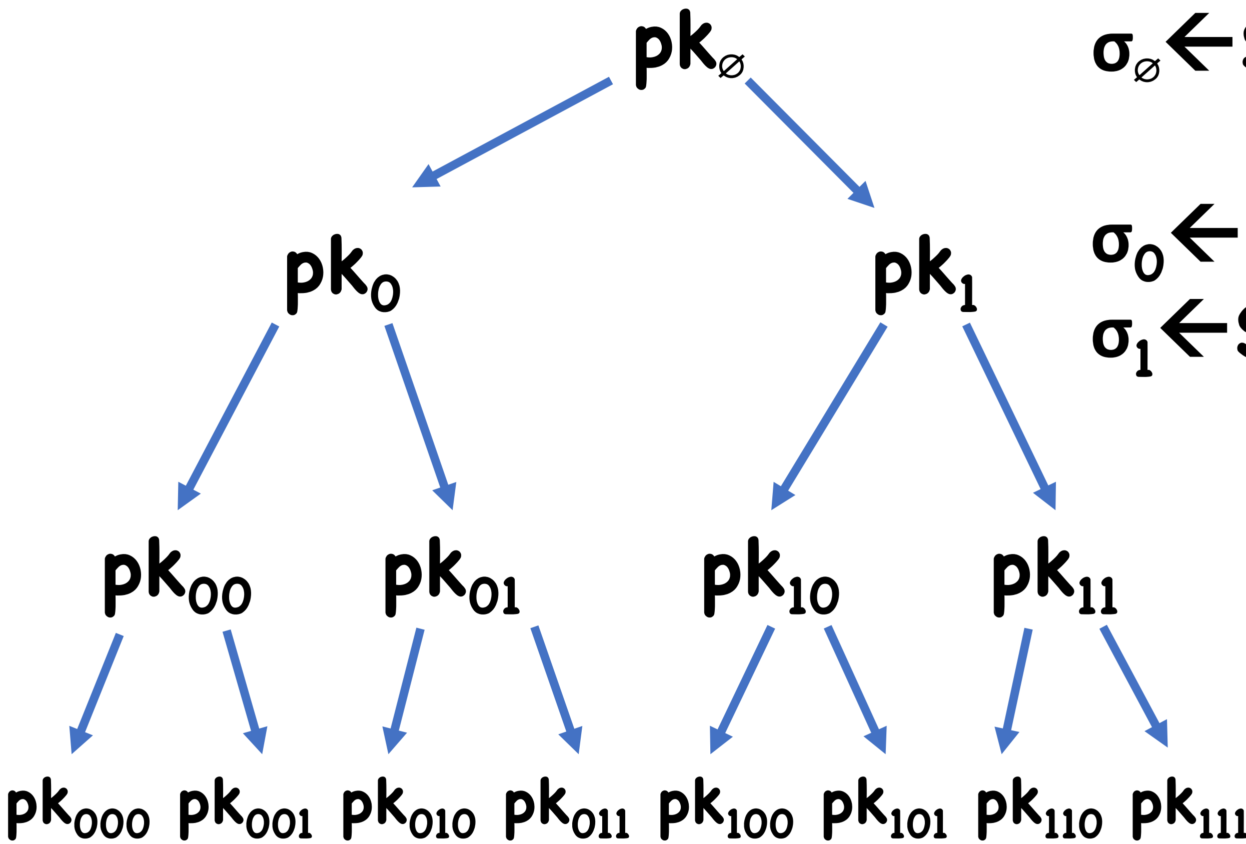
$\mathrm{Ver}(pk_2, (m_2, pk_3)\ , \sigma_2')$

Still Limitations

Now Bob (and Charlie, etc) are stateless

However, Alice is still stateful
- Needs to remember all messages sent
- Signature length grows with number of messages signed

Signature Trees



$\sigma_\varnothing \leftarrow \text{Sign}(sk_\varnothing,\ (pk_0, pk_1))$

$\sigma_0 \leftarrow \text{Sign}(sk_0,\ (pk_{00}, pk_{01}))$

$\sigma_1 \leftarrow \text{Sign}(sk_1,\ (pk_{10}, pk_{11}))$

$\sigma_{00},\ \sigma_{01},\ \sigma_{10},\ \sigma_{11}$

Signature Trees

To sign $\mathbf{m_i}$,

- Compute $\boldsymbol{\sigma_i} \leftarrow \mathbf{Sign(sk_i,m_i)}$, where $\mathbf{sk_i}$ is the **i**th leaf
- Must include $\mathbf{pk_i}$ in signature so Bob can verify $\boldsymbol{\sigma_i}$
- Must authenticate $\mathbf{pk_i}$, so include $\boldsymbol{\sigma_{P(i)}}$ (and $\mathbf{pk_{S(i)}}$)
- Must include $\mathbf{pk_{P(i)}}$ so Bob can verify $\boldsymbol{\sigma_{P(i)}}$
- Must auth $\mathbf{pk_{P(i)}}$, so include $\boldsymbol{\sigma_{P(P(i))}}$ (and $\mathbf{pk_{S(P(i))}}$)
- …

Comparison to Chaining

Limitations:

- Bounded number of messages ($\mathbf{2^d}$)
- Still requires Alice to keep state (all the **sk**'s, **pk**'s).

  Size of state ≈ $\mathbf{2^d}$

Advantages:

- Signature size ≈ **d**, logarithmic in number of messages signed

Avoid Large State?

Alice keeps PRF key **k** as part of secret key

- For all internal nodes or leaves **i**,

$$(sk_i,pk_i) \leftarrow Gen(;\ PRF(k,\ i)\ )$$

- Alice never stores signatures or public keys
- Instead, she computes needed signatures/public keys on the fly

Unbounded Messages

Set $d{=}128$ or 256

- Can now sign up to 2^{128} messages
- Signature size $\propto d = 128$, so shortish signatures
- Size of state independent of d, so short
- Time to compute signature?
  - Only need pk's,σ's on path from root to leaf, plus neighbors
  - Only $O(d)$ terms
  - Can efficiently compute from PRF key k

Fully Stateless?

So far, still need to keep state to remember which leaf we should use next

However, now we can do something different:

- Instead of choosing leafs sequentially, just choose leaf at random
- Except with probability $O(|\text{messages}|^2/2^d)$, never use the same leaf twice

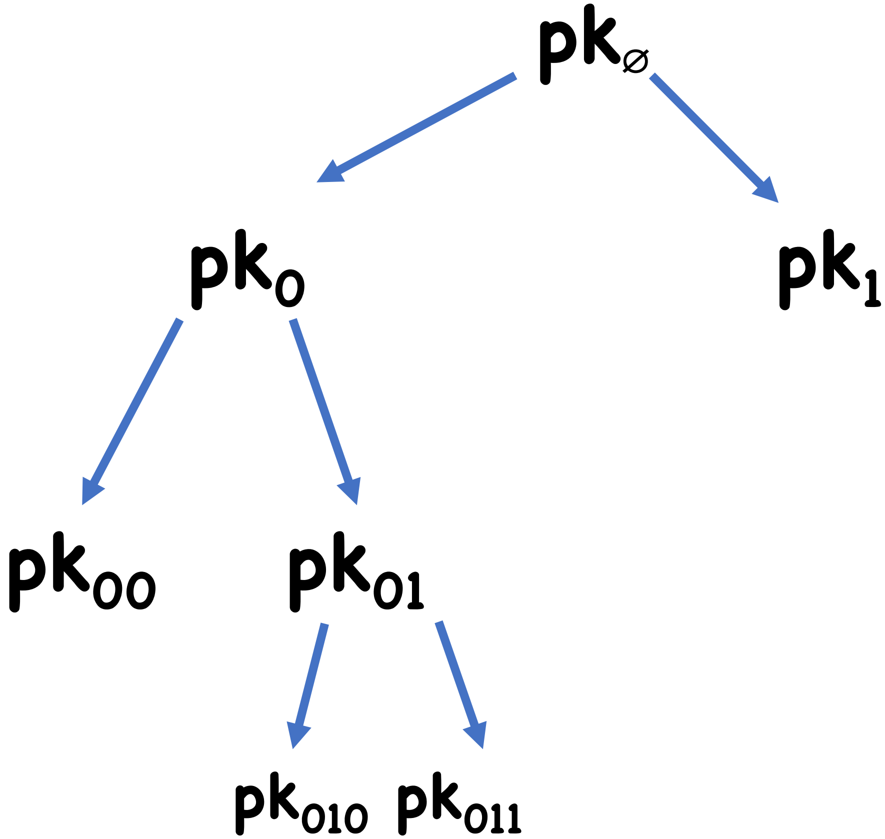
Putting it Together

$pk_\varnothing$

$sk=(sk_\varnothing, k)$

$i \leftarrow \{0,\ldots,2^d-1\}$

Putting it Together



$sk=(sk_\varnothing, k)$

$(sk_0,pk_0)\leftarrow Gen(; PRF(k, 0))$
$(sk_1,pk_1)\leftarrow Gen(; PRF(k, 1))$
$(sk_{00},pk_{00})\leftarrow Gen(; PRF(k, 00))$
$(sk_{01},pk_{01})\leftarrow Gen(; PRF(k, 01))$
…

$\sigma_\varnothing\leftarrow Sign(sk_\varnothing, (pk_0,pk_1))$
$\sigma_0\leftarrow Sign(sk_0, (pk_{00},pk_{01}))$
…

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

Output all $\mathbf{pk_j}$'s and all $\boldsymbol{\sigma}$'s as signature

Putting it Together

OWF to get 1-time signatures (with large **pk**’s, **σ**’s)

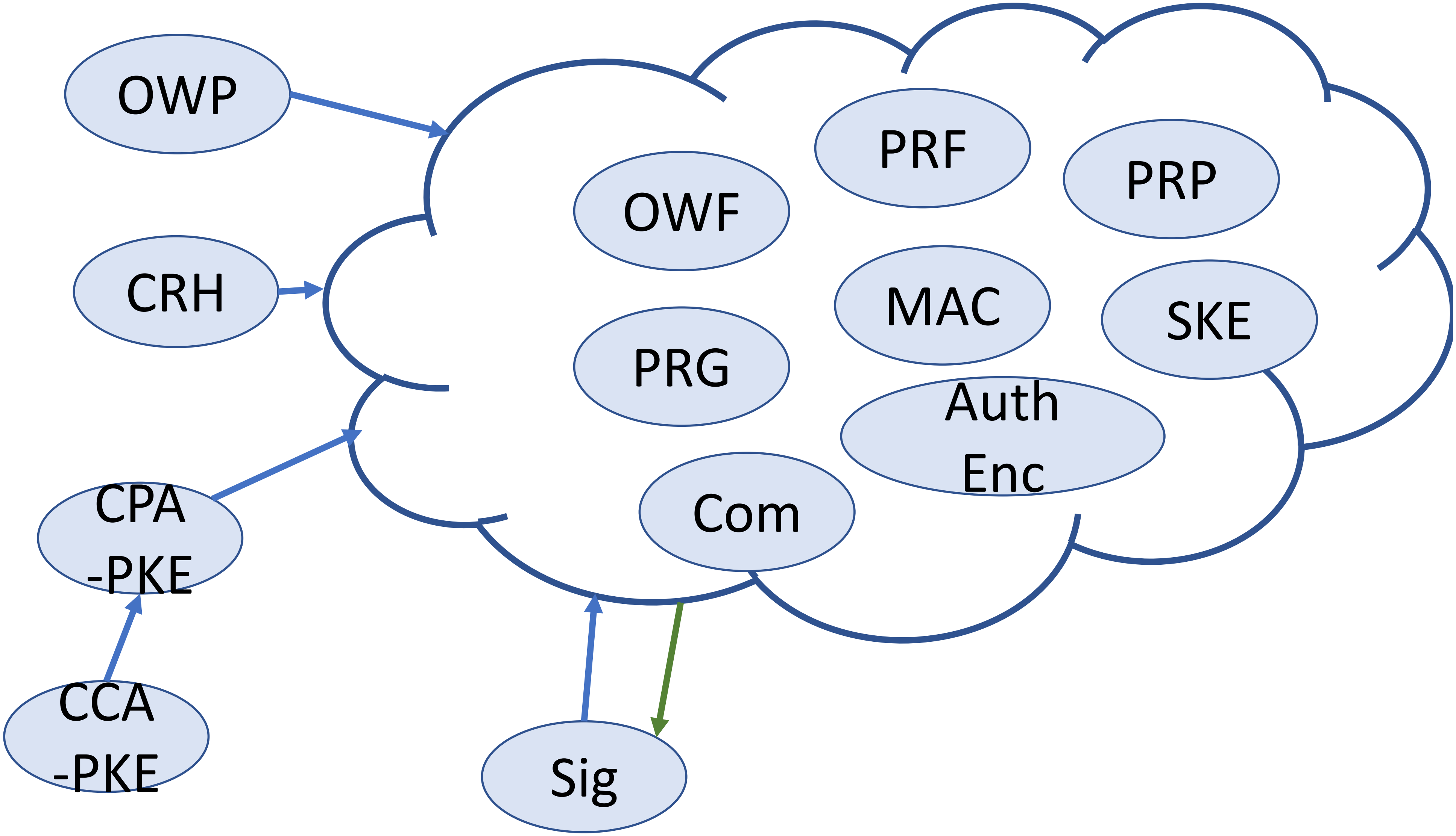
Hash message

- 1-time signatures with small **pk**’s, **σ**’s
- Can accomplish using just OWFs

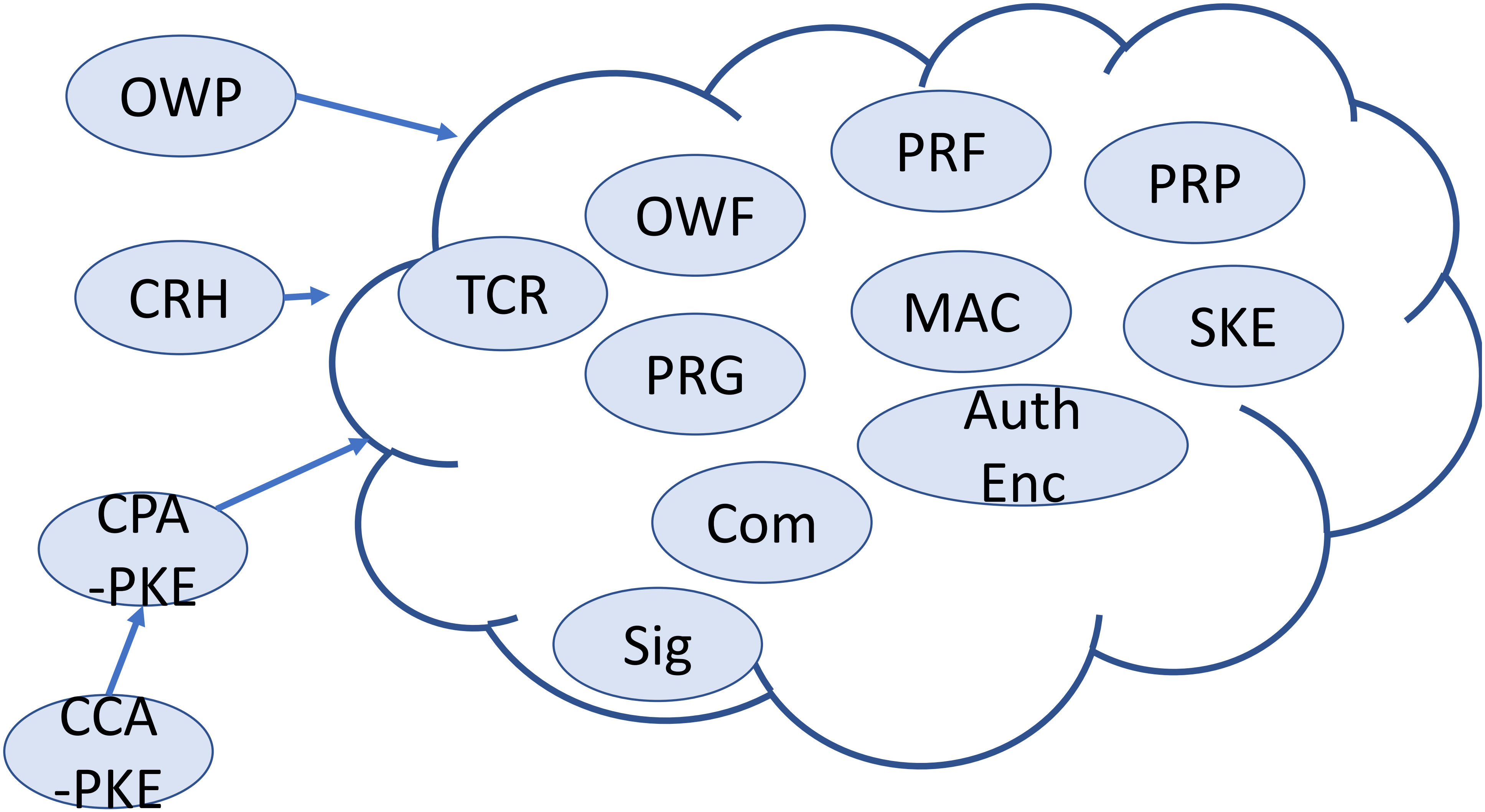
Create tree of signatures (stateful scheme)

Make stateless by using a PRF

What's Known



What's Known



OWP
CRH
CPA -PKE
CCA -PKE
OWF
TCR
PRG
PRF
PRP
MAC
SKE
Auth Enc
Com
Sig

Theorem: Given a secure OWF, it is possible to construct a strongly CMA-secure signature scheme

Practical Use?

Lamport signatures are fast:

- Signing is just revealing part of your secret key
- Verifying is just a few OWF evaluations

Tree-based signatures are a bit slower

- Need to generate many signatures
- Need to generate many public keys
- Need many PRF evals

Practical Use?

Main limitation: Signature size

- Basic Lamport: 128 bits per message bit
- With hashing, need to sign 256 bit messages
- For signature trees, signature consists of **d** Lamport signatures (plus public keys)
  - **d** must be big enough to prevent collisions
  - E.g. **d = 128**

Overall signature size: around a **megabit**

What’s the Smallest Signature?

Signature Trees: 1megabits

RSA Hash-and-Sign: 2 kilobits

ECDSA: around 512 bits

BLS: 256 bits

Are 128-bit signatures possible?

Obfuscation-Based Signatures

Let **(MAC,Ver)** be a message authentication code

Gen(): k←K
- **sk = k**
- **pk = Obf(Ver(k, . , .))**

Sign(sk,m) = MAC(k,m)
Ver(pk,m,σ) = pk(m,σ)

Signature size: 128 bits!
- But running time, public key size is horrible

Next Time

Identification protocols: how to prove you are who you say you are

Reminders

HW6 Due Wednesday

HW7 out Tonight