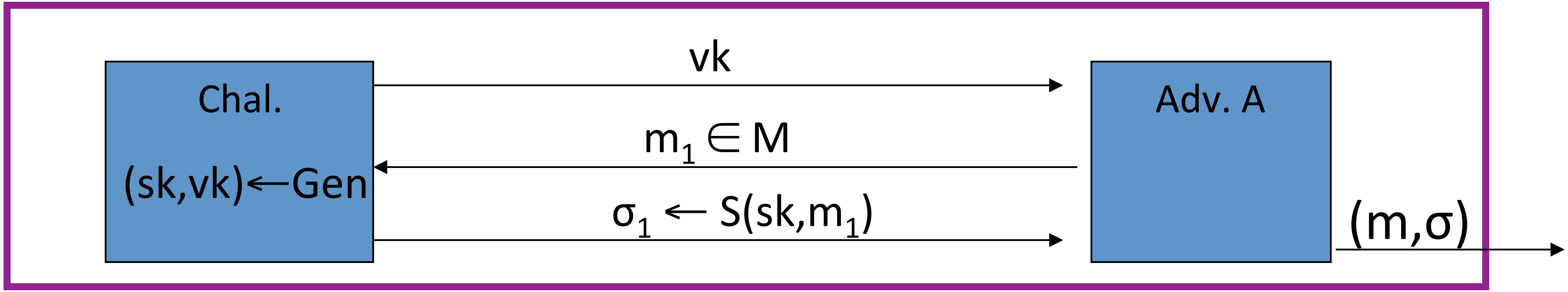
Sigs. with special properties

Fast one-time signatures and applications

One-time signatures: definition

Suppose signing key is used to sign a <u>single</u> message

Can we give a simple (fast) construction SS=(Gen,S,V) ?



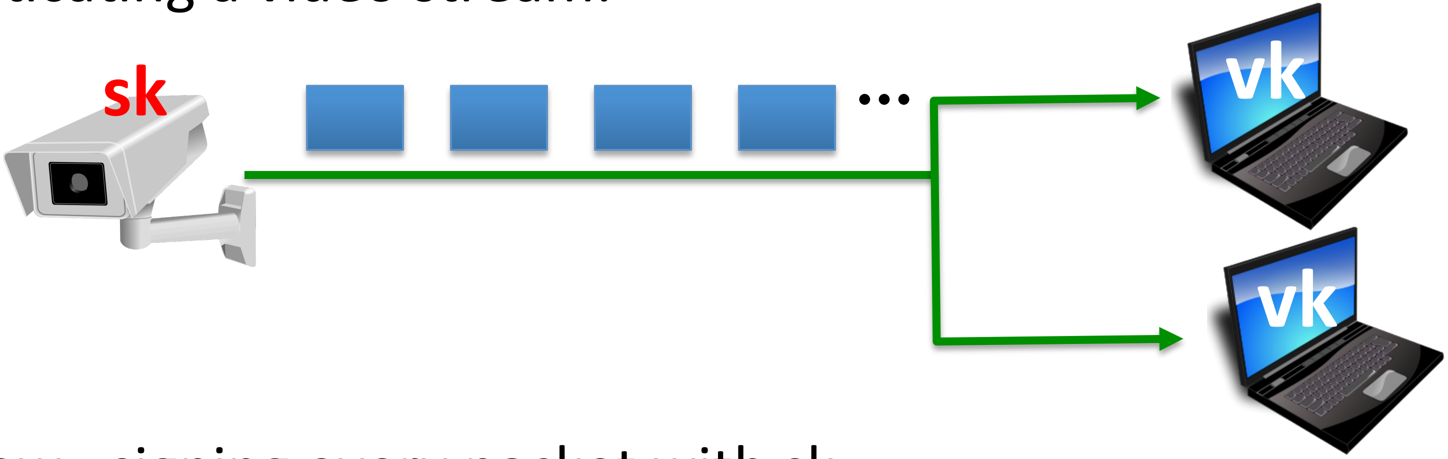
A wins if $V(vk,m,\sigma)$ = \`accept' and $m \neq m_1$

Security: for all "efficient" A, $\mathrm{Adv}_{\text{1-SIG}}[A,SS]$ = Pr[A wins] ≤ negl

Application: authenticating streams

1. Next section: secure one-time sigs ⇒ secure many-time sigs

2. Authenticating a video stream:

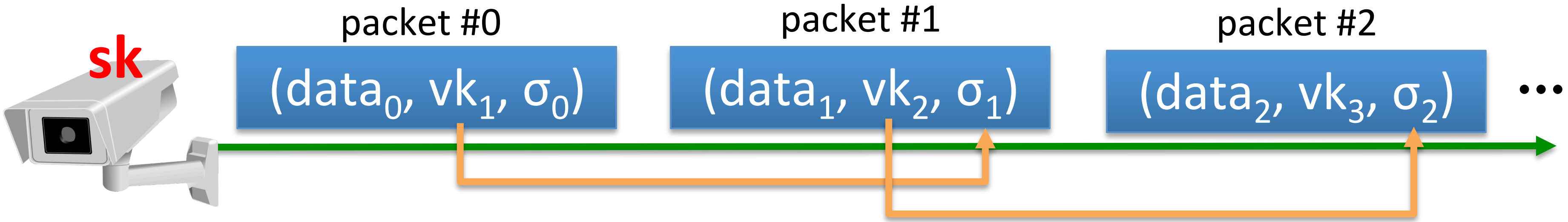


Too slow: signing every packet with sk

Solution using a fast one-time sig

(sk,vk): key-pair for a many-time signature scheme

$(Gen_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)



Packet #0: $(sk_1, vk_1) \longleftarrow Gen_{1T}$, $\sigma_0 \longleftarrow S\big(sk, (data_0, vk_1)\big)$

Packet #1: $(sk_2, vk_2) \longleftarrow Gen_{1T}$, $\sigma_1 \longleftarrow S_{1T}\big(sk_1, (data_1, vk_2)\big)$

Packet #2: $(sk_3, vk_3) \longleftarrow Gen_{1T}$, $\sigma_2 \longleftarrow S_{1T}\big(sk_2, (data_2, vk_3)\big)$

Recipient accepts packet #2 = ($data_2$, vk_3, σ_2) once it verifies σ_2

How does the recipient verify the signature σ_2 in packet #2?

Accept if σ_0 and σ_1 were valid and:

- $V_{1T}(vk_3, (data_2, vk_3), \sigma_2)$ = "accept"
- $V(vk, (data_2, vk_3), \sigma_2)$ = "accept"
- $V_{1T}(vk_2, (data_2, vk_3), \sigma_2)$ = "accept"
- $V(vk_2, (data_2, vk_3), \sigma_2)$ = "accept"

Application: authenticating streams

Practical difficulties:

- Packet loss, out of order delivery
- Many solutions: see further reading at end of module

Authenticating streams with a MAC:

- Harder, but can be done: TESLA

End of Segment

Sigs. with special properties

Constructing fast one-time signatures

One-time signatures

Secure when sk only signs a <u>single</u> message

Attacker: gets vk and can ask for sig. on any <u>single</u> m_1 of her choice.

should be unable to forge signature on $m \neq m_1$

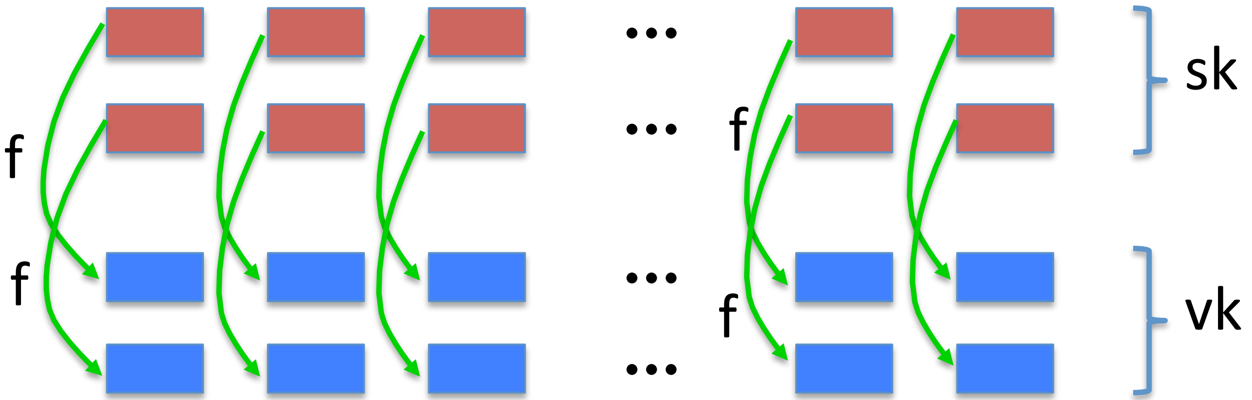
This module: one-time sigs from fast **one-way functions** (OWF)

- $f\colon X \longrightarrow Y$ is a OWF if (1) $f(x)$ is efficiently computable, (2) hard to invert on random $f(x)$
- Examples: (1) $f(x) = \text{AES}(\underbrace{x}_{\text{key}}, 0^{128})$, (2) $f(x) = \text{SHA256}(x)$

Lamport one-time signatures (simple)

f: X ⟶ Y a one-way function. Msg space: $M = \{0,1\}^{256}$

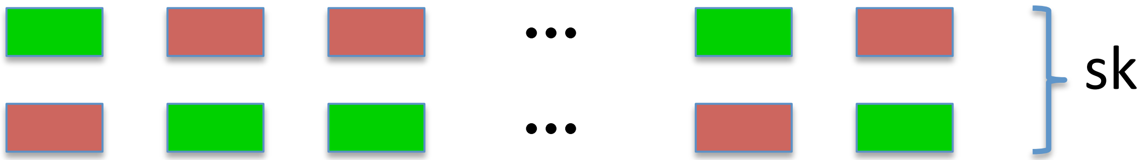
Gen: generate 2×256 random elements in X



Lamport one-time signatures (simple)

f: X $\longrightarrow$ Y a one-way function. Msg space: M = $\{0,1\}^{256}$

Gen: generate 2×256 random elements in X



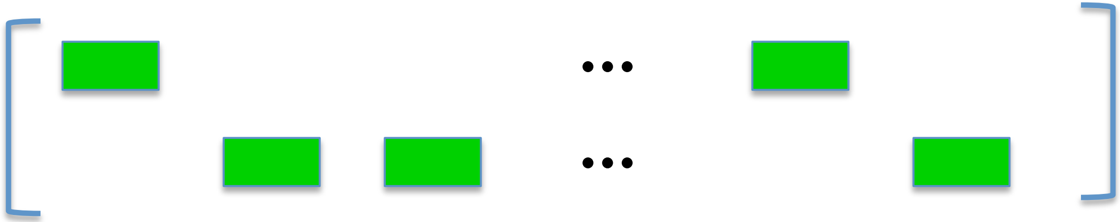
m = 0 1 1 … 0 0

S(sk, m): σ = (pre-images corresponding to bits of m)

Lamport one-time signatures (simple)

$f: X \longrightarrow Y$ a one-way function. Msg space: $M = \{0,1\}^{256}$

Gen: generate 2×256 random elements in X

$\sigma =$  $\in X^{256}$ (2KB)

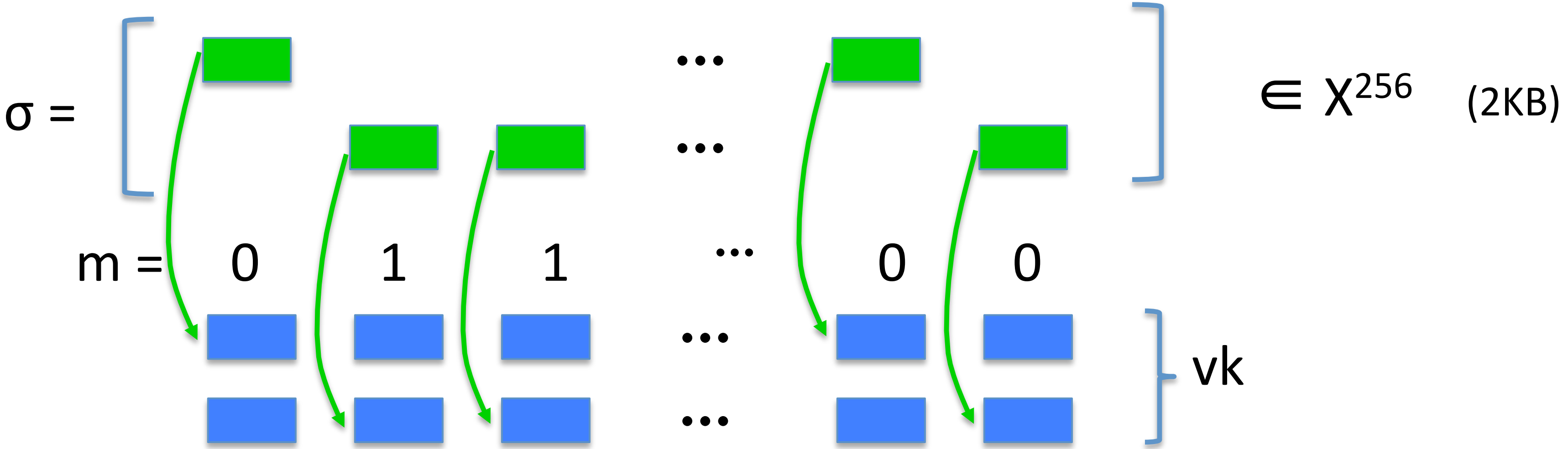
m = 0 1 1 … 0 0

S(sk, m): $\sigma =$ (pre-images corresponding to bits of m)

Lamport one-time signatures (simple)

$f: X \longrightarrow Y$ a one-way function. Msg space: $M = \{0,1\}^{256}$

Gen: generate $2{\times}256$ random elements in X



V(vk, m, σ): accept if all pre-images in σ match values in vk

Very fast signature system. Will prove one-time security in a bit.

Is it two-time secure? That is, if **sk** is used to sign two messages, can an attacker do an existential forgery?

- ○ No, one-time security implies two-time security
- ○ It depends on the one-way function used
- ○ The attacker can ask for a signature on 0^{128} and on 1^{128}. He gets all of **sk** which he can use to sign new messages.

Abstraction: cover free set systems

Sets: $S_1, \; S_2, \; \ldots, \; S_{2^{256}} \;\subseteq\; \{1, \ldots, n\}$

<u>Def</u>: $\boldsymbol{S} = \{S_1, \; S_2, \; \ldots, \; S_{2^{256}}\}$ is **cover-free** if $S_i \not\subseteq S_j$ for all $i \neq j$

Example: if all sets in ***S*** have the same size k then ***S*** is cover free

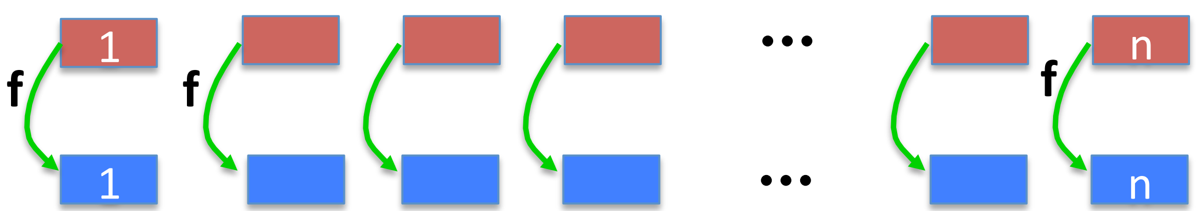
Abstract Lamport signatures

$f: X \longrightarrow Y$ a one-way function. Msg space: $M = \{0,1\}^{256}$

$\boldsymbol{S} = \{S_1, \quad S_2, \quad \ldots, \quad S_{2^{256}} \}$ is **cover-free** over $\{1,..,n\}$

$H: \{0,1\}^{256} \longrightarrow \boldsymbol{S}$ a bijection (one-to-one)

Gen: generate n random elements in X



sk $\in X^n$

vk $\in Y^n$

Abstract Lamport signatures

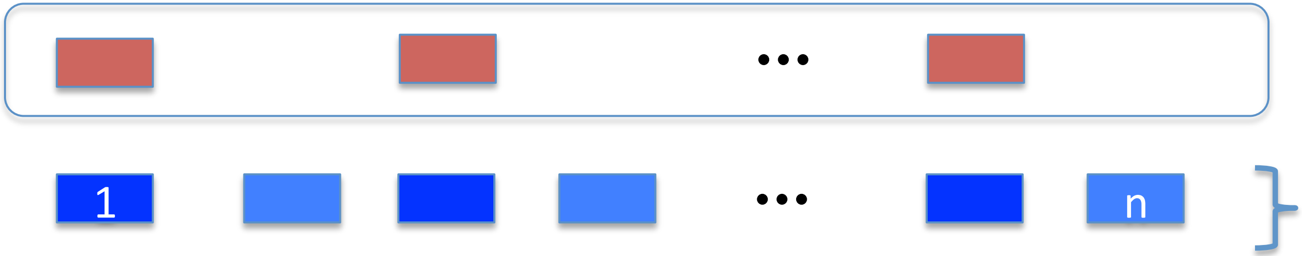
$f: X \longrightarrow Y$ a one-way function. Msg space: $M = \{0,1\}^{256}$

$\boldsymbol{S} = \{S_1, \quad S_2, \quad \ldots, \quad S_{2^{256}}\}$ is **cover-free** over $\{1,..,n\}$

$H: \{0,1\}^{256} \longrightarrow \boldsymbol{S}$ a bijection (one-to-one)

Gen: generate n random elements in X

$\sigma =$



vk $\in Y^n$

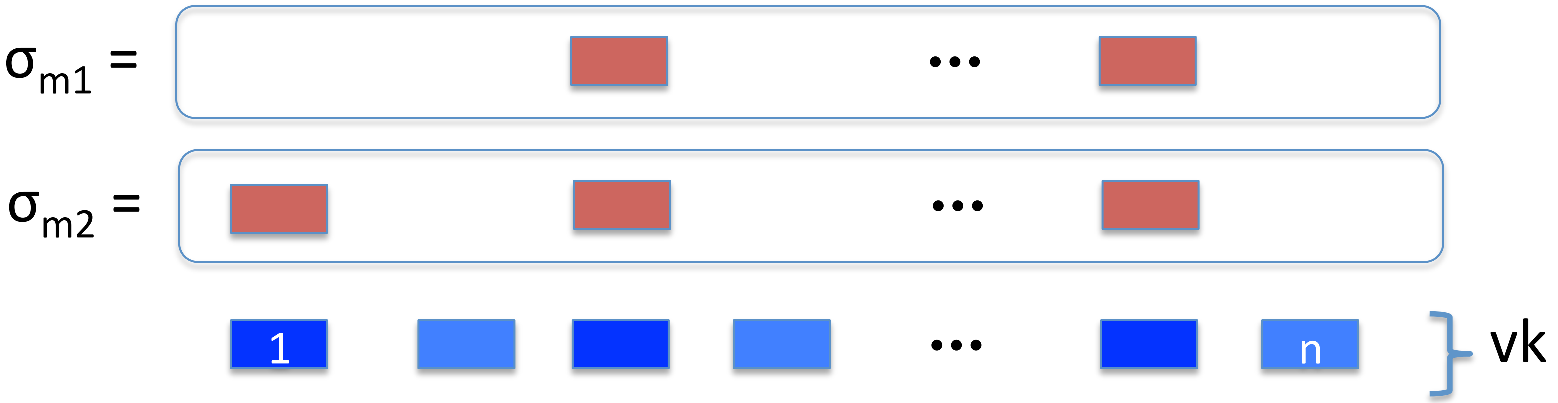
S(sk, m): $\sigma = \Big($ pre-images corresponding to elements of H(m) $\Big)$

Why cover free?

Suppose S were not cover free

$\Rightarrow$ exists m_1, m_2 such that $H(m_1) \subset H(m_2)$

$\Rightarrow$ signature on m_2 gives signature on m_1



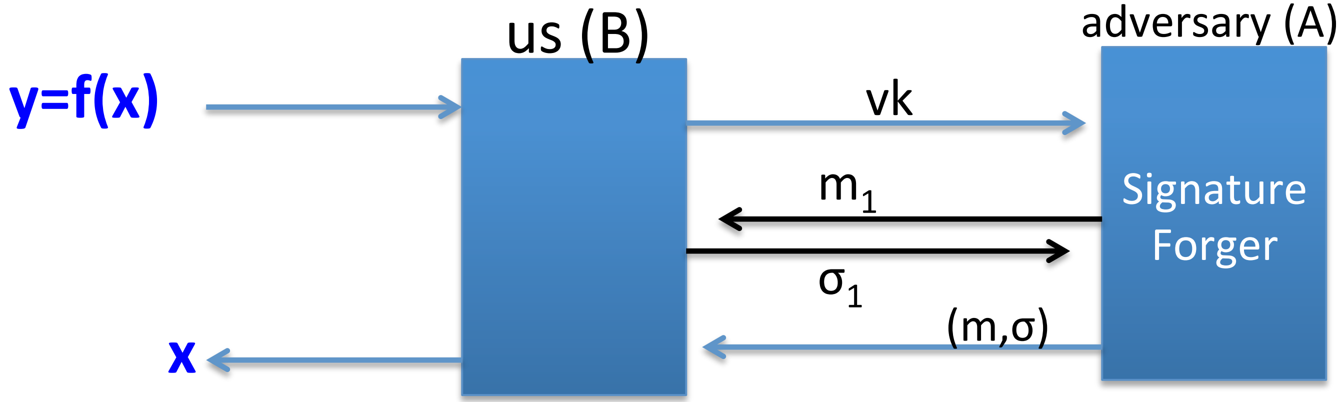
S(sk, m): $\sigma = \Big($ pre-images corresponding to elements of H(m) $\Big)$

Security statement

Thm: if $f: X \longrightarrow Y$ is one-way and $\boldsymbol{S}$ is cover-free
then Lamport signatures (Lam) are one-time secure.

$$\forall A \ \exists B: \qquad \mathrm{Adv}_{\text{1-SIG}}[A,\mathrm{Lam}] \le n \cdot \mathrm{Adv}_{\text{OWF}}[B,f]$$

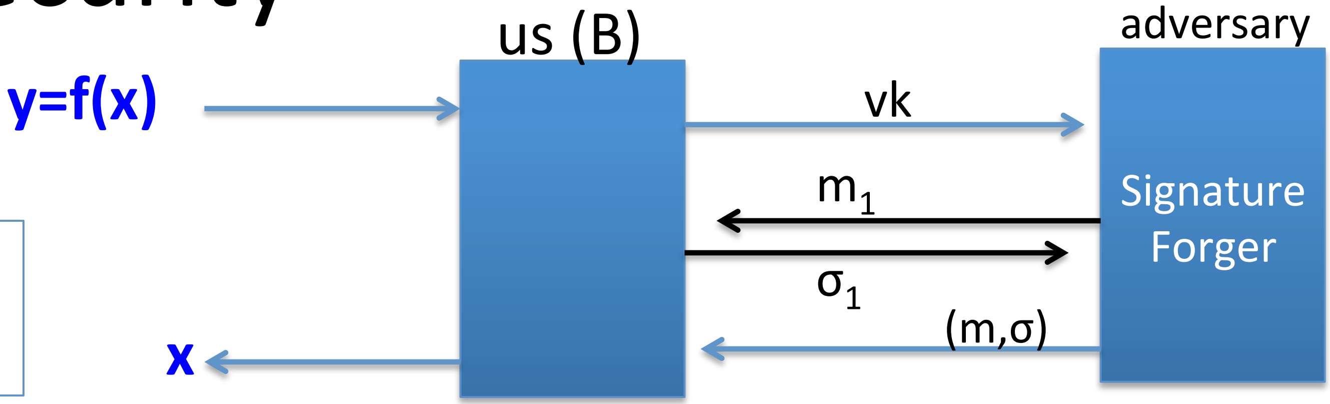
Proving security:



Proving security

choose: $i \longleftarrow \{1,\ldots,n\}$

$x_1,\ldots,x_n \longleftarrow X$



vk= $f(x_1)$ $\cdots$ $f(x_{i-1})$ y $f(x_{i+1})$ $\cdots$ $f(x_n)$

Parameters $(f: X \longrightarrow Y \quad \text{where } X = Y)$

$\boldsymbol{S} = \{S_1, \quad S_2, \quad \ldots, \quad S_{2^{256}}\}$ is **cover-free** over $\{1,..,n\}$

In particular: $\boldsymbol{S} = \binom{\text{all subsets of } \{1,\ldots,n\}}{\text{of size } k}$

$vk \in Y^n \quad \Rightarrow \quad$ vk size = (n elements of Y)

sig. size = (k elements of X)

Msg-space $= \{0,1\}^{256} \quad \Rightarrow \quad |\boldsymbol{S}| = (n \text{ choose } k) \geq 2^{256}$

- To shrink signature size, choose small k

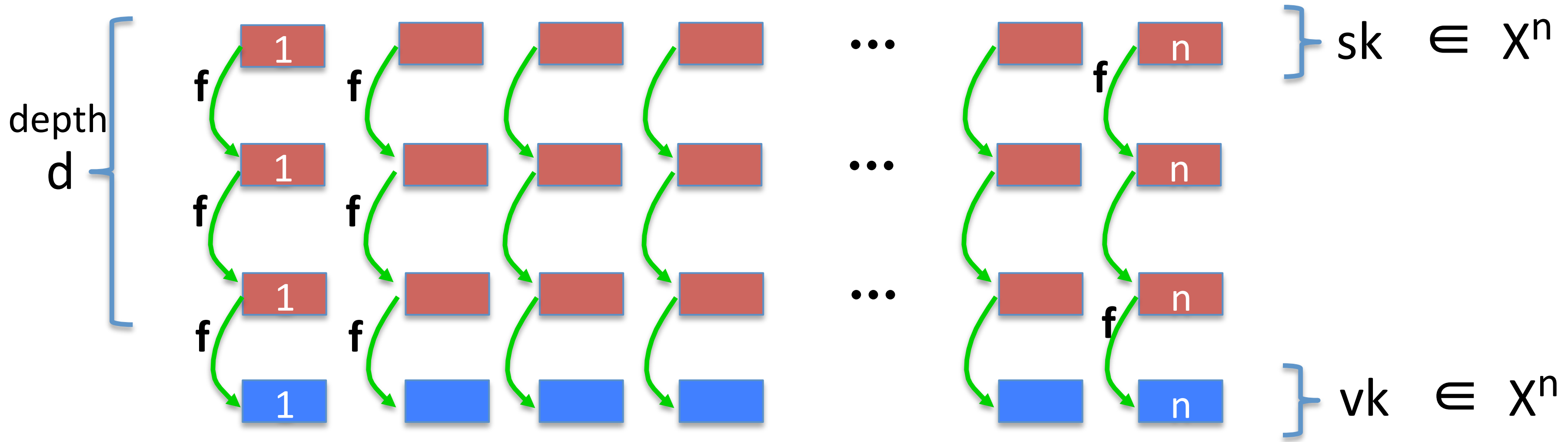
  example: $k=32 \quad \Rightarrow \quad n \geq 3290$

- For optimal (sig-size + vk-size) choose $n = 261, k = 123$

  (sig-size + vk-size) $\approx 1.5 \times 256$ elements of X

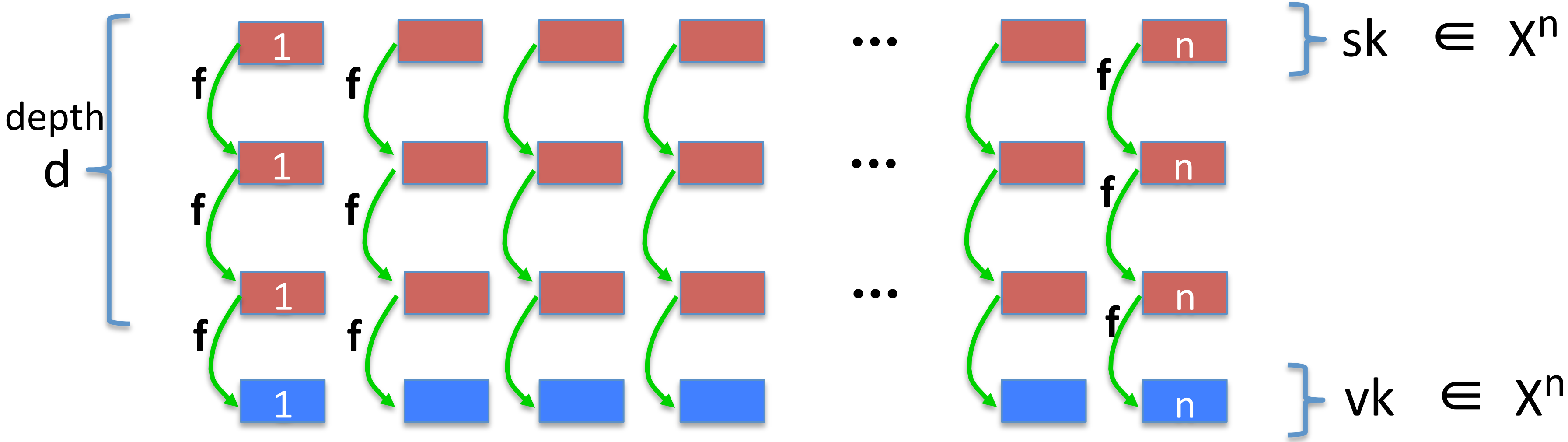
Further improvement: Winternitz

Gen: generate n random elements in X : (f: X ⟶ X)



Further improvement: Winternitz

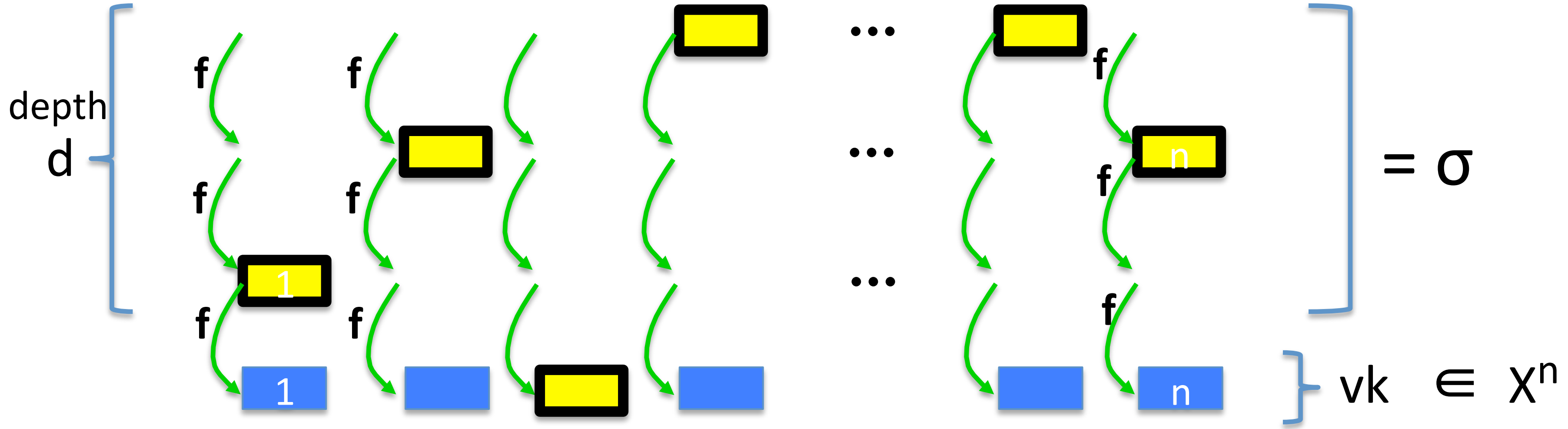
$$H: \{0,1\}^{256} \longrightarrow \{0,1,\ldots,d\}^n$$



S(sk, m): $\sigma = \big($ pre-images indicated by H(m) $\big)$

Further improvement: Winternitz

H: $\{0,1\}^{256} \longrightarrow \{0,1,\ldots,d\}^n$ ex: $H(0^{256}) = (2, 1, 3, 0, \ldots, 0, 1)$



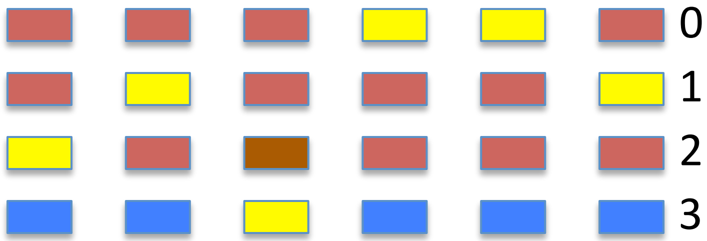
S(sk, m): $\sigma = \big($ pre-images indicated by H(m) $\big)$

For what H is this a secure one-time signature?

Suppose $H(0^{256}) = (2, 1, 3, 0, 0, 1)$

$H(1^{256}) = (2, 2, 3, 1, 1, 2)$

Is the signature one-time secure?



- No, from a sig. on 0^{256} one can construct a sig. on 1^{256}
- No, from a sig. on 1^{256} one can construct a sig. on 0^{256}
- Yes, the signature is one-time secure
- It depends on how H behaves at other points

Optimized parameters

For one-time security need that:

for all $m_0 \neq m_1$ we have $H(m_0)$ does not "cover" $H(m_1)$

Parameters:

- Time(sign) = Time(verify) = $O(n \cdot d)$
- vk size = sig. size = (n elements in X)
- msg-space = $\{0,1\}^{256}$ $\Rightarrow$ $n > 256 / \log_2(d)$ (approx.)

(vk size)+(sig. size) $\approx$ $256 \times (2/\log_2(d))$ elems. of X

For Lamport: (vk size)+(sig. size) $\approx$ $256 \times (1.5)$ elems. of X

End of Segment

Sigs. with special properties

One-time signatures ⇒ many-time signatures

Review

Recall: one-time signatures need not be 2-time secure

example: Lamport signatures

Goal: convert any one-time signature into a many-time signature

Main tool: collision resistant hash functions

Construction

$(Gen_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)

Four-time signature: (stateful version)

- **Gen**:

$Gen_{1T} \longrightarrow \boxed{(vk_{0123}, sk_{0123})}$

(vk_{01}, sk_{01}) (vk_{23}, sk_{23})

(vk_0, sk_0) (vk_1, sk_1) (vk_2, sk_2) (vk_3, sk_3)

Construction

$(Gen_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)

Four-time signature: (stateful version)

- **Gen**:

$\boxed{vk_{0123},\ \sigma_{0123}} \leftarrow S_{1T}\big(sk, (vk_{01}, vk_{23})\big)$

(vk_{01}, sk_{01}) (vk_{23}, sk_{23})

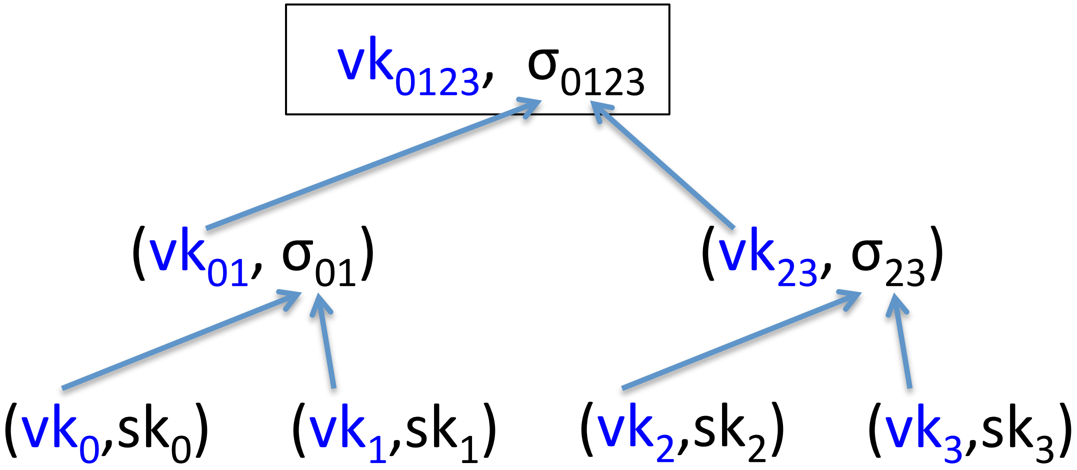
(vk_0, sk_0) (vk_1, sk_1) (vk_2, sk_2) (vk_3, sk_3)

Construction

$(\mathrm{Gen}_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)

Four-time signature: (stateful version)

- **Gen**:



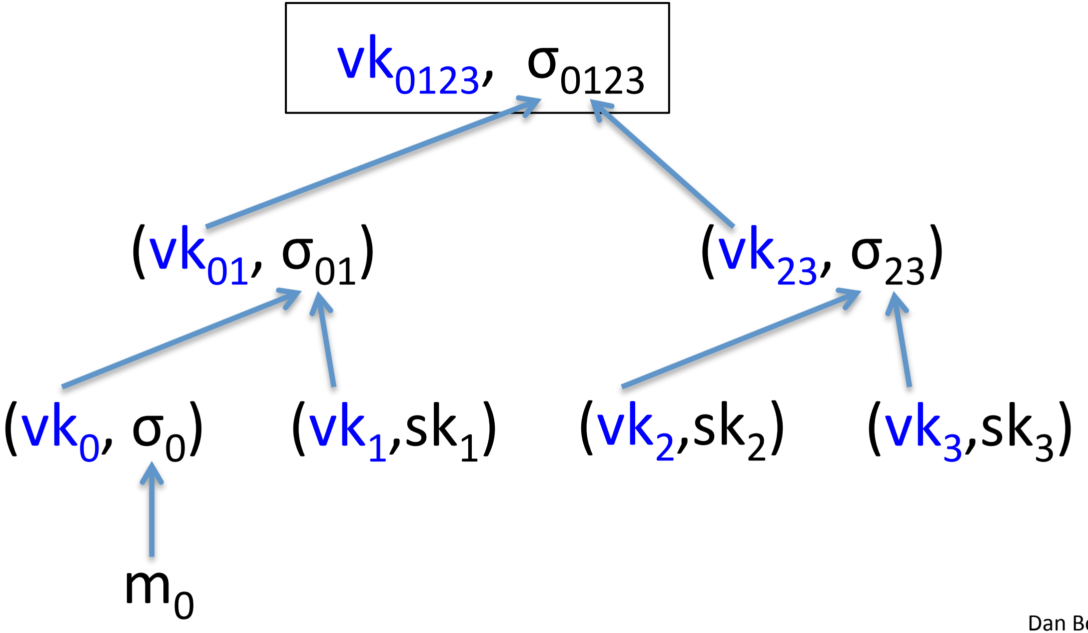
Construction

$(Gen_{1T}, S_{1T}, V_{1T})$:   secure one-time signature (fast)

Four-time signature: (stateful version)

Sig. on msg m_0:

($\sigma_{0123}, \sigma_{01}, \boldsymbol{\sigma_0},$
$vk_{01}, vk_{23}, vk_0, vk_1$)



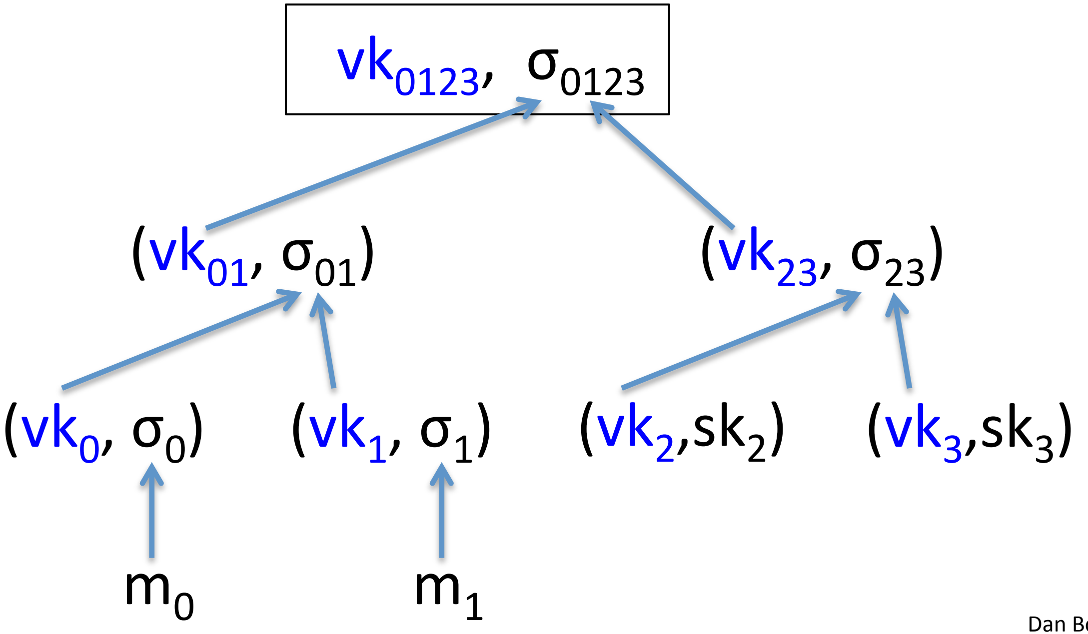
Construction

$(Gen_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)

Four-time signature: (stateful version)

Sig. on msg m_1:

$(\sigma_{0123}, \sigma_{01}, \boldsymbol{\sigma_1},$
$vk_{01}, vk_{23}, vk_0, vk_1)$



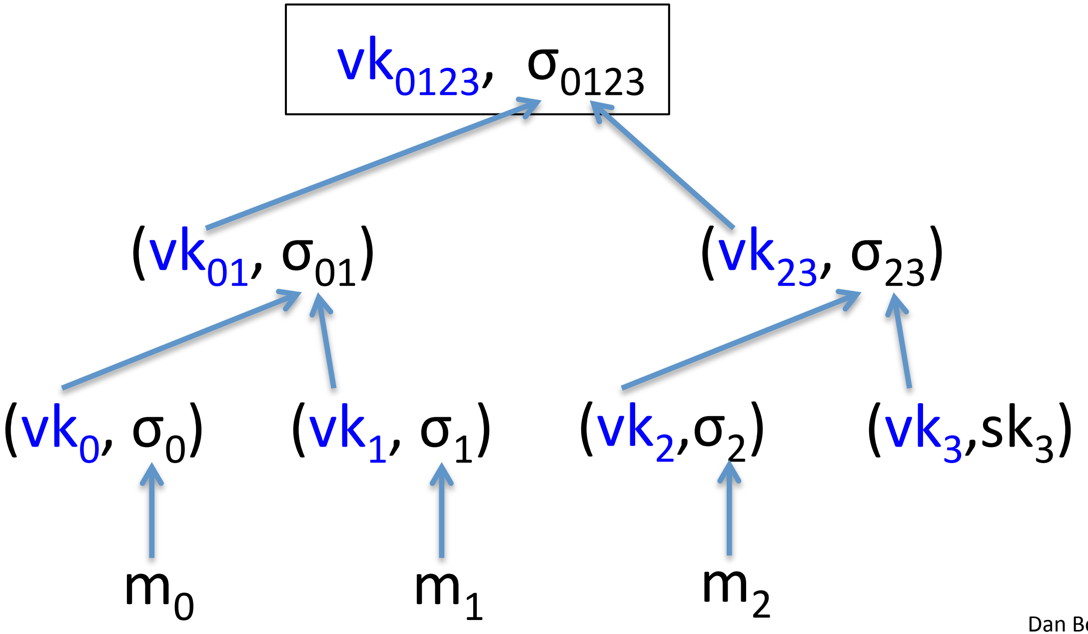
Construction

$(Gen_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)

Four-time signature: (stateful version)

Sig. on msg m_2:

($\sigma_{0123}, \sigma_{01}, \boldsymbol{\sigma_2}$,
$vk_{01}, vk_{23}, \mathbf{vk_2}, \mathbf{vk_3}$)



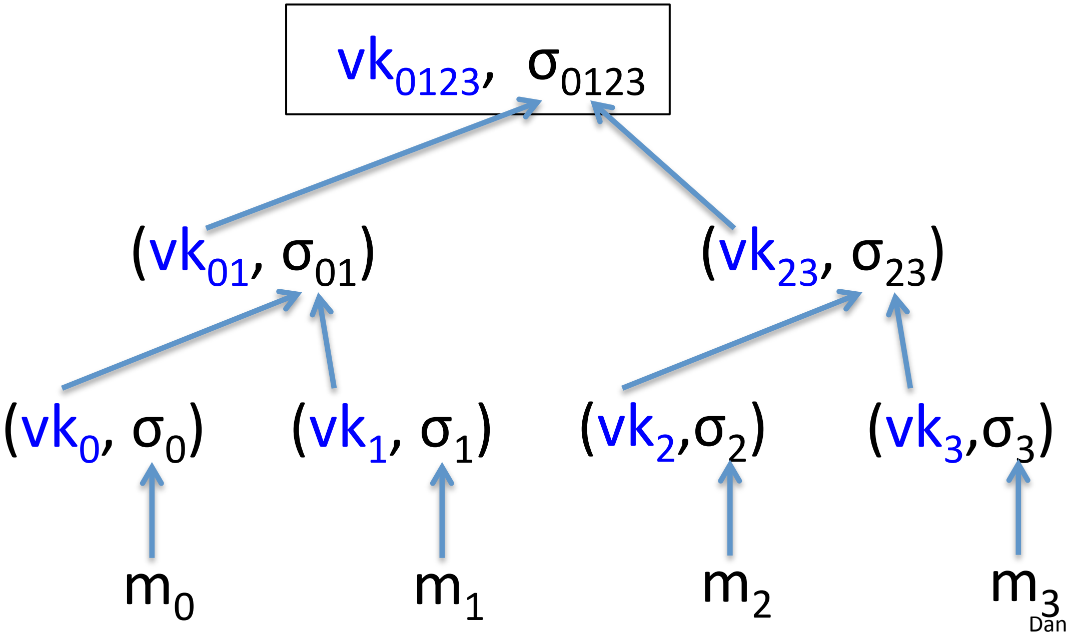
Construction

$(Gen_{1T}, S_{1T}, V_{1T})$: secure one-time signature (fast)

Four-time signature: (stateful version)

Sig. on msg m_3:

$(\ \sigma_{0123}, \sigma_{01}, \boldsymbol{\sigma_3},$
$vk_{01}, vk_{23}, \mathbf{vk_2}, \mathbf{vk_3}\ )$



More generally: 2^d-time signature

Tree of depth d:

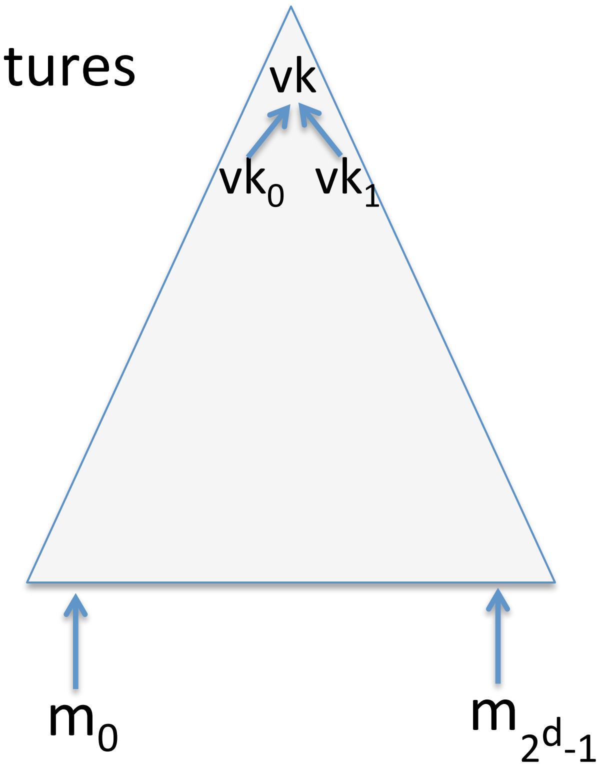
- Every signature contains d+1 one-time signatures along with associated vk's

Tree is generated on-the fly:

- Signer stores only d secret keys at a time

Stateful signature:

- Signer maintains a counter indicating which leaf to use for signature
- Every leaf must only be used once!



Optimized 2^d-time signatures

Combined with Lamport signatures:

- collision resistant hash funs ⇒ many-time signature

With further optimizations:

- For 2^{40} signatures: signature size is ≈ 5KB

  … signing time is about the same as RSA signatures

- Recall: RSA sig size is 256 bytes (2048 bit RSA modulus)

End of Segment

Sigs. with special properties

Super-fast online signatures

Goals

Problem: generating RSA, ECDSA, BLS signatures can be slow

- On low power devices

sk

Goal:

- Do heavy signature computation **<u>before</u>** message is known
- Quickly output signature once user supplies message

Method 1: using one-time sigs

(Gen, S, V): secure many-time signature (slow)

(Gen_{1T}, S_{1T}, V_{1T}): secure one-time signature (fast)

- Gen $\longrightarrow$ (sk,vk)
- PreSign(sk): $(sk_{1T}, vk_{1T}) \longleftarrow Gen_{1T}$, $\sigma \longleftarrow S(sk, vk_{1T})$ slow
- $S_{online}\big((\sigma, sk_{1T}, vk_{1T}) , m\big)$: $\sigma_{1T} \longleftarrow S_{1T}(sk_{1T}, m)$ fast

  output $\boldsymbol{\sigma^* \longleftarrow (vk_{1T}, \sigma, \sigma_{1T})}$
- $V_{online}\big(vk, m, \boldsymbol{\sigma^*=(vk_{1T}, \sigma, \sigma_{1T})}\big)$:

  accept if $V(vk, vk_{1T}, \sigma) = V_{1T}(vk_{1T}, m, \sigma_{1T}) =$ "accept"

Method 1: using one-time sigs

One-time sigs. $\Rightarrow$ fast-online sigs.

Problem: Lamport results in very long signatures

A more suitable one-time signature:

- Hard Dlog in group G $\Rightarrow$ secure one-time sigs. with **fast** signing
  - Signature size: if $|G|=p$ then signature is $\underbrace{(\mathbf{r},\mathbf{s})}_{\text{64 bytes}} \in (Z_p)^2$
  - How: see homework problem

Better method: chameleon hash

G: finite cyclic group of order p. $g,\ h=g^{\alpha} \in G$ generators.

Define: $H(m,r) = g^r \cdot h^m \in G$

Properties:

- $H(m,r)$ can be efficiently evaluated
- H is collision resistant if Dlog in G is hard (collision $\longrightarrow$ $\alpha = \mathrm{Dlog}_g(h)$)
- If α is known: given m and t can find r s.t. $H(m,r) = h^t$

  $r = (t-m) \cdot \alpha$. Indeed: $H(m,r) = g^r \cdot h^m =$

Fast online signatures

(Gen, S, V): secure many-time signature (slow)

G: finite cyclic group of order p. $g,\ h = g^{\alpha} \in G$ rand. generators

- Gen $\longrightarrow$ (sk,vk) , $sk^* = (sk, \alpha)$

slow

- PreSign(sk^*): random $t \leftarrow Z_p$, $\sigma \leftarrow S(sk,\ h^t)$
- $S_{online}\big(\ (\sigma, \alpha, t, h^t)\ ,\ m\big)$: $\mathbf{r \leftarrow (t\text{-}m) \cdot \alpha}$, output $\boldsymbol{\sigma^* \leftarrow (\sigma,\ h^t,\ r)}$

fast

- $V_{online}\big(\ vk,\ m,\ \boldsymbol{\sigma^* = (\sigma, h^t, r)}\big)$:

  accept if $V(vk, \mathbf{h^t})$ = "accept" and $H(m,r) = \mathbf{h^t}$

Dan Boneh

Fast online signatures

Shorter signatures than one-time sigs. method:

- Total overhead is only 64 bytes
- Signature time: one multiplication in Z_p

Security:

- A forger can be used to either
  - (1) forge signatures for (Gen, S, V), or
  - (2) find collisions on H(m,r)

Fast online signatures have a fast <u>online</u> signing time.

If we count the entire signing time (i.e.   PreSign + Sign), would the time be better or worse than a standard signature like RSA?

- ○ Online signatures are always faster than regular signatures
- ○ The PreSign step uses a regular signatures, so overall they cannot be faster than a regular signature
- ○ It depends on which online signature is used

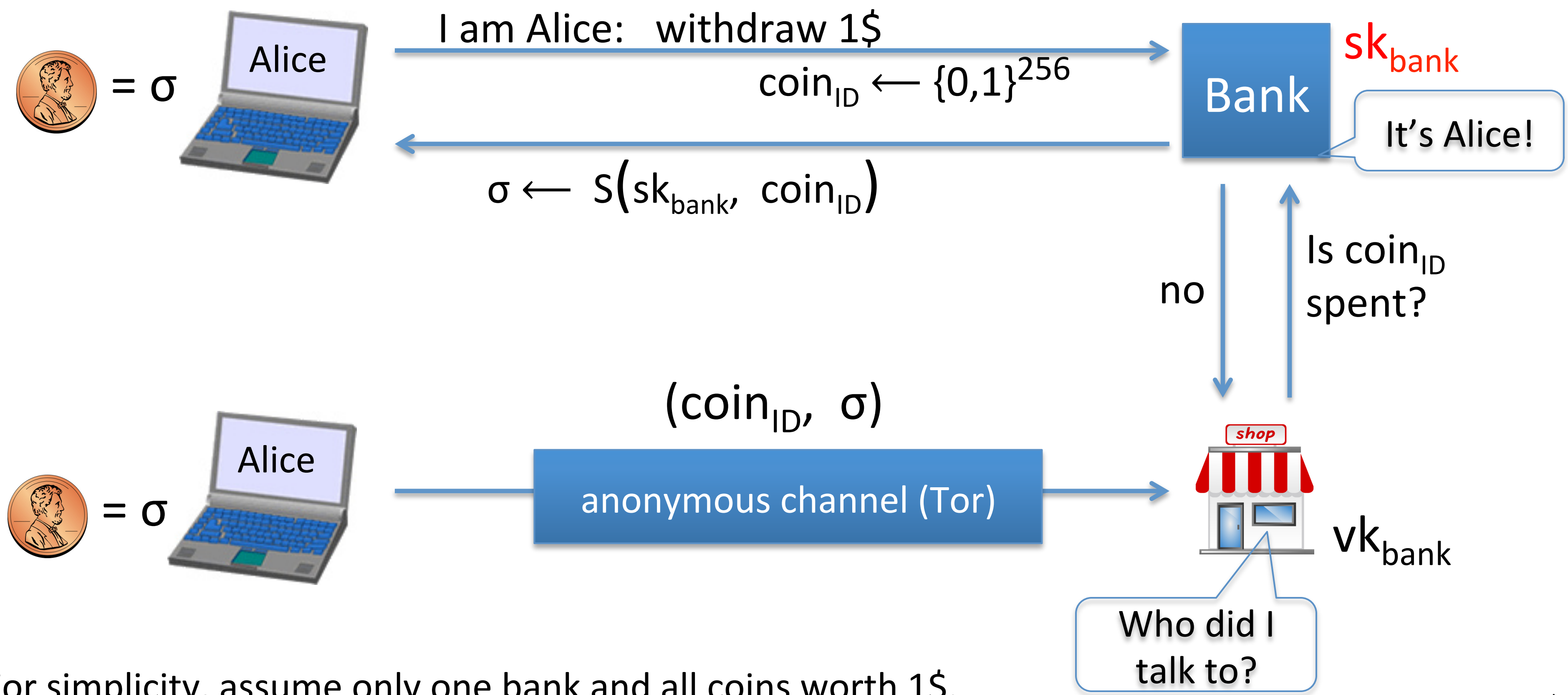
Note: signature verification time is always worse than regular sigs.

End of Segment

Sigs. with special properties

Blind signatures

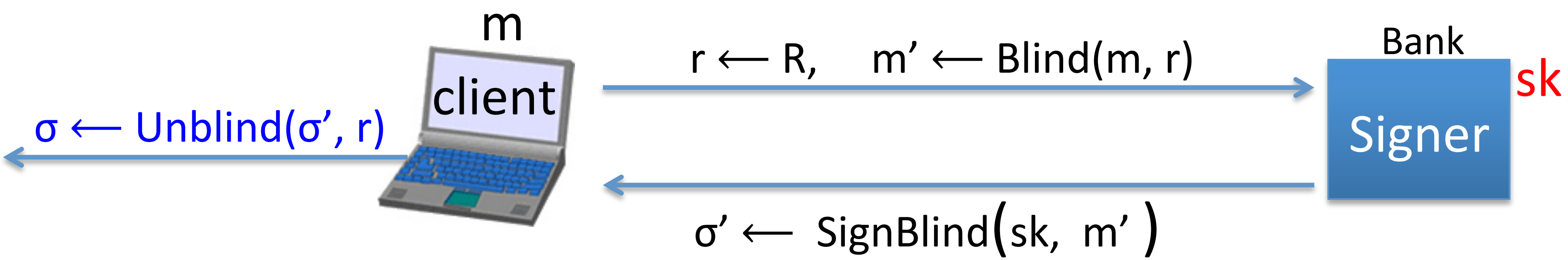
Problem: digital cash (centralized system)



For simplicity, assume only one bank and all coins worth 1$.

Solution: blind signatures

Goal: we want Bank to sign $coin_{ID}$, but without knowing $coin_{ID}$



Where:

(1) σ is a valid signature on m: $V(vk, m, \sigma) =$ "accept"

(2) $m' \leftarrow \mathrm{Blind}(m)$ is independent of m

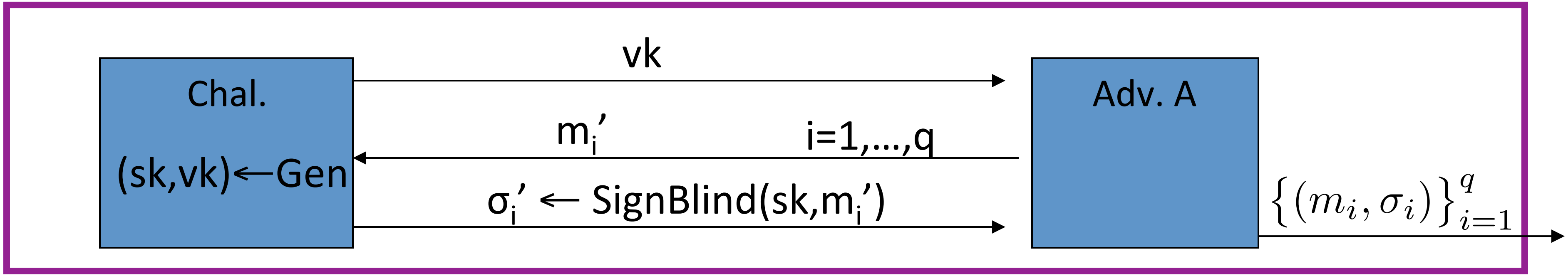
- That is, m' reveals no "information" about m

Blind signatures: security

New definition of existential forgery:

adversary asks for q blind signatures, and

outputs (q+1) message/signature pairs



A wins if $V(vk, m_i, \sigma_i)$ = \`accept' for all i=1,...,q+1

Security: for all "efficient" A, $\mathrm{Adv}_{\mathrm{Blind}}[A, SS]$ = Pr[A wins] ≤ negl

Blind signatures: applications

- Anonymous digital cash
- Anonymous voting systems
  - Election results are known, but not who voted how
- Adaptive oblivious transfer (week 4)

Simple Constructions: RSA and BLS

BLS review: G finite group of order p with a pairing

$sk = \alpha \in Z_p$, $vk = (g, g^{\alpha})$, $H: M \longrightarrow G$

$S(sk, m) = H(m)^{\alpha} \in G$

Independent of m

m

$r \leftarrow Z_p$, $m' \leftarrow H(m) \cdot g^r$

Bank

Signer

sk=α

$\sigma' \leftarrow (m')^{\alpha}$

$\sigma \leftarrow \sigma' / (g^{\alpha})^r$

Indeed: $\sigma = (m')^{\alpha} / (g^{\alpha})^r =$

Same method also works for RSA. Problem: security under strong assumption.

Suppose the signature scheme is changed so that

the random r is chosen as $r \longleftarrow \{0,1,\ldots,16\}$.

Would the resulting scheme be a secure blind signature?

- ○ No, an attacker can ask one query and generate two signatures
- ○ Yes, this has no impact on security and blindness
- ○ No, the sig. scheme is not blind: m' is not independent of m
- ○ It depends on the hash function H

Further Reading

- Hash Based Digital Signature Schemes. C. Dods, N. Smart, M. Stam, 2005
- One-Time Signatures Revisited: Practical Fast Signatures Using Fractal Merkle Tree Traversal. D. Naor, A. Shenhav, A. Wool, 2006.
- Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. L. Reyzin, N. Reyzin, 2002
- Improved Online/Offline Signature Schemes. A. Shamir, Y. Tauman, 2001
- The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme. M. Bellare, C. Namprempre, D. Pointcheval, M. Semanko, 2001
- Compact E-Cash.   J. Camenisch, S. Hohenberger, A. Lysyanskaya, 2005

End of Segment