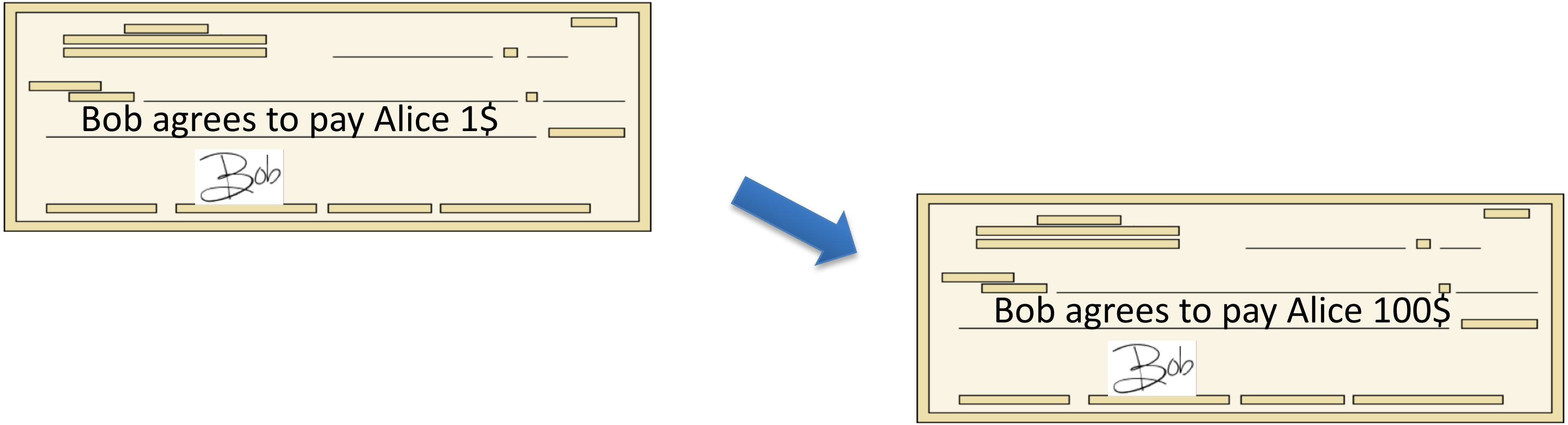
Digital Signatures

What is a digital signature?

Physical signatures

Goal: bind document to author

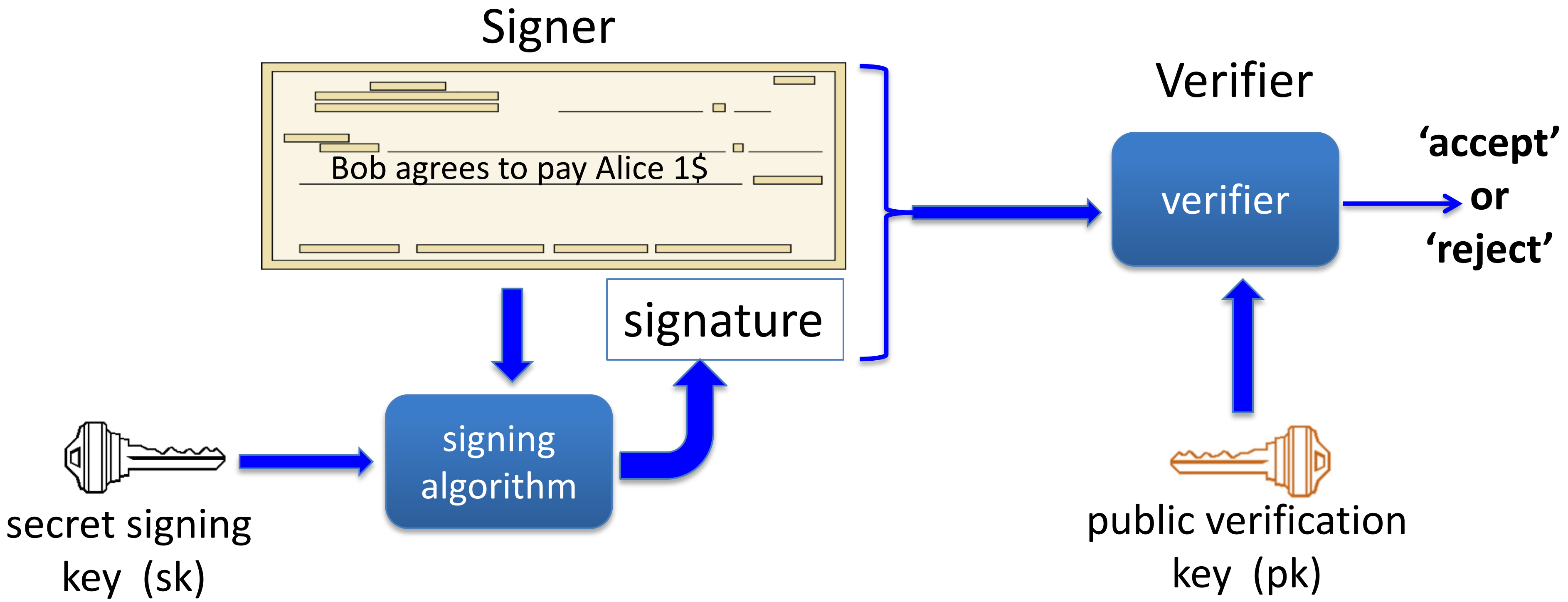


Problem in the digital world:

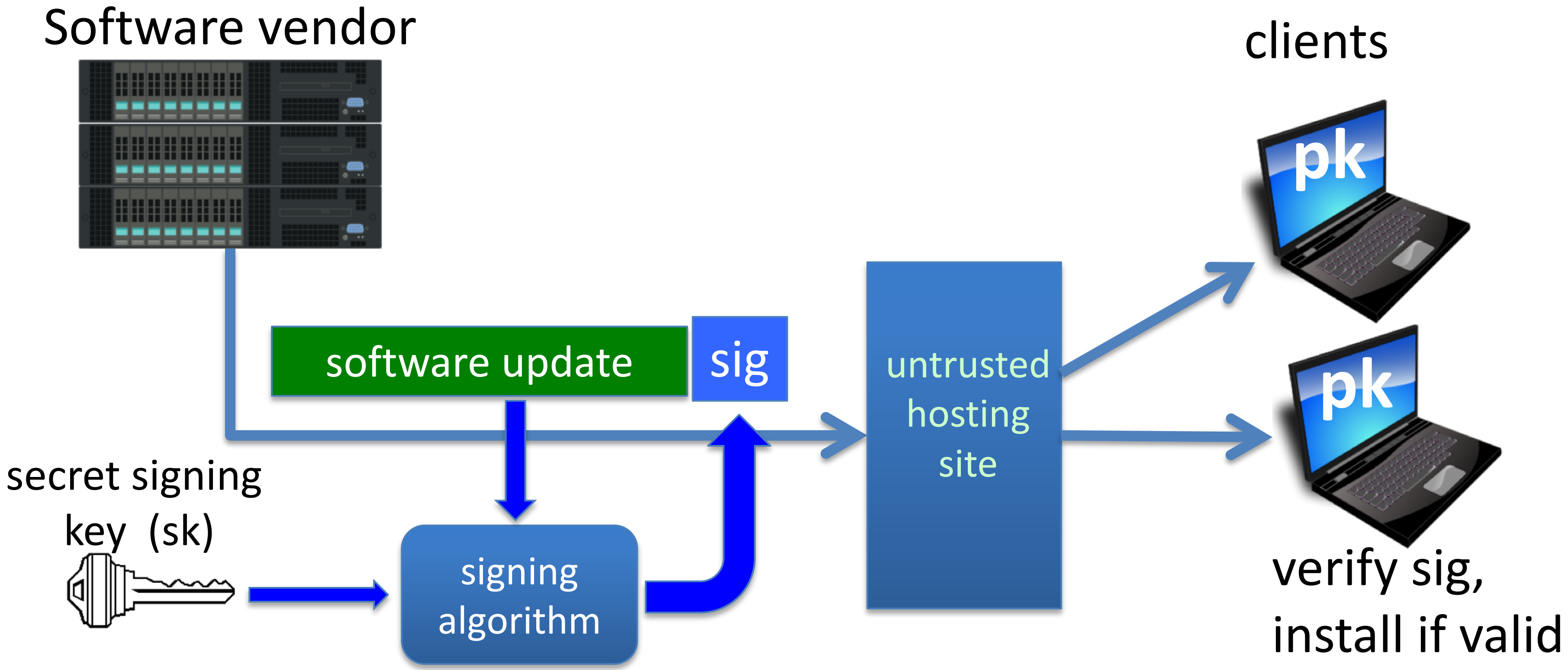
anyone can copy Bob's signature from one doc to another

Digital signatures

Solution: make signature depend on document



A more realistic example



Digital signatures: syntax

Def: a signature scheme (Gen,S,V) is a triple of algorithms:

- Gen(): randomized alg. outputs a key pair (pk, sk)
- S(sk, $m \in M$) outputs sig. σ
- V(pk, m, σ) outputs 'accept' or 'reject'

Consistency: for all (pk, sk) output by Gen :

$$\forall m \in M: \quad V\big(pk,\ m,\ S(sk, m)\big) = \text{'accept'}$$

Digital signatures: security

Attacker's power: **chosen message attack**

- for $m_1, m_2, \ldots, m_q$ attacker is given $\sigma_i \leftarrow S(sk, m_i)$

Attacker's goal: **existential forgery**

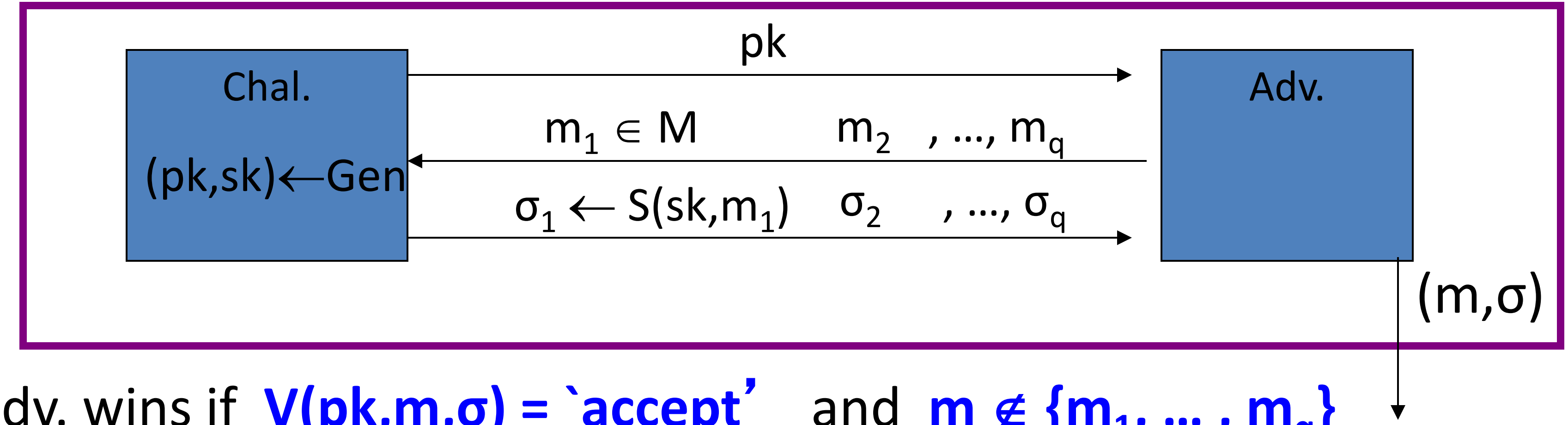
- produce some **<u>new</u>** valid message/sig pair (m, σ).

$$m \notin \{ m_1, \ldots, m_q \}$$

$\Rightarrow$ attacker cannot produce a valid sig. for a <u>new</u> message

Secure signatures

For a sig. scheme (Gen,S,V) and adv. A define a game as:



Adv. wins if **$V(pk,m,\sigma)$ = `accept'** and **$m \notin \{m_1, \ldots, m_q\}$**

Def: SS=(Gen,S,V) is **secure** if for all "efficient" A:

$$\text{Adv}_{\text{SIG}}[A,SS] = \Pr[\,A \text{ wins}] \quad \text{is "negligible"}$$

Let (Gen,S,V) be a signature scheme.

Suppose an attacker is able to find $m_0 \neq m_1$ such that

$$\mathbf{V(pk, m_0, \sigma) = V(pk, m_1, \sigma)} \quad \text{for all } \sigma \text{ and keys } (pk, sk) \leftarrow \text{Gen}$$

Can this signature be secure?

- Yes, the attacker cannot forge a signature for either m_0 or m_1
- No, signatures can be forged using a chosen msg attack
- It depends on the details of the scheme

Alice generates a (pk,sk) and gives pk to her bank.

Later Bob shows the bank a message m="**pay Bob 100$**" properly signed by Alice, i.e. V(pk,m,sig) = 'yes'

Alice says she never signed m. Is Alice lying?

- ○ Alice is lying: existential unforgeability means Alice signed m and therefore the Bank should give Bob 100$ from Alice's account
- ○ Bob could have stolen Alice's signing key and therefore the bank should not honor the statement
- ○ What a mess: the bank will need to refer the issue to the courts

End of Segment

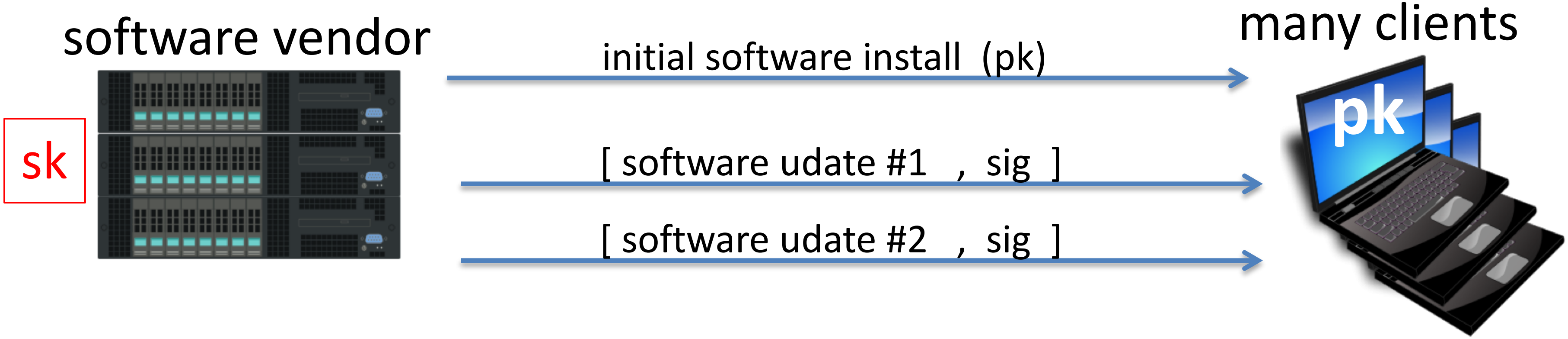
Digital Signatures

Applications

Applications

Code signing:

- Software vendor signs code
- Clients have vendor's pk.   Install software if signature verifies.

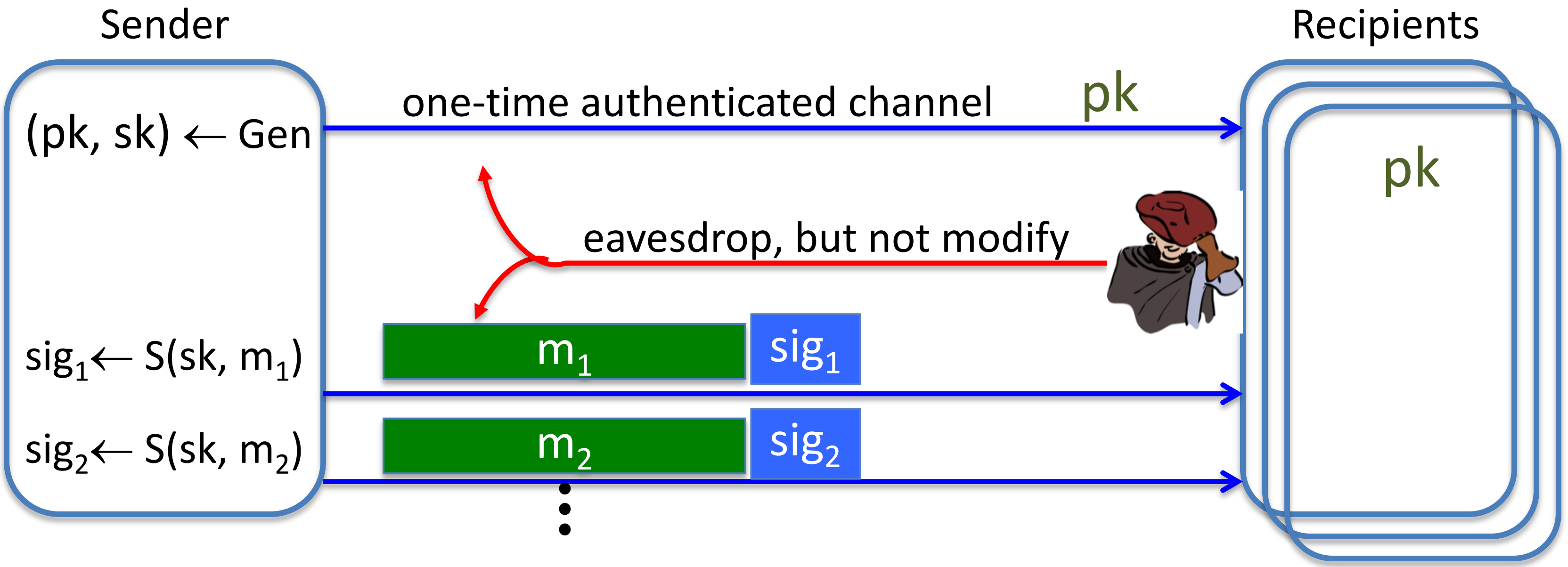


More generally:

One-time authenticated channel (non-private, one-directional)

$\Longrightarrow$ many-time authenticated channel

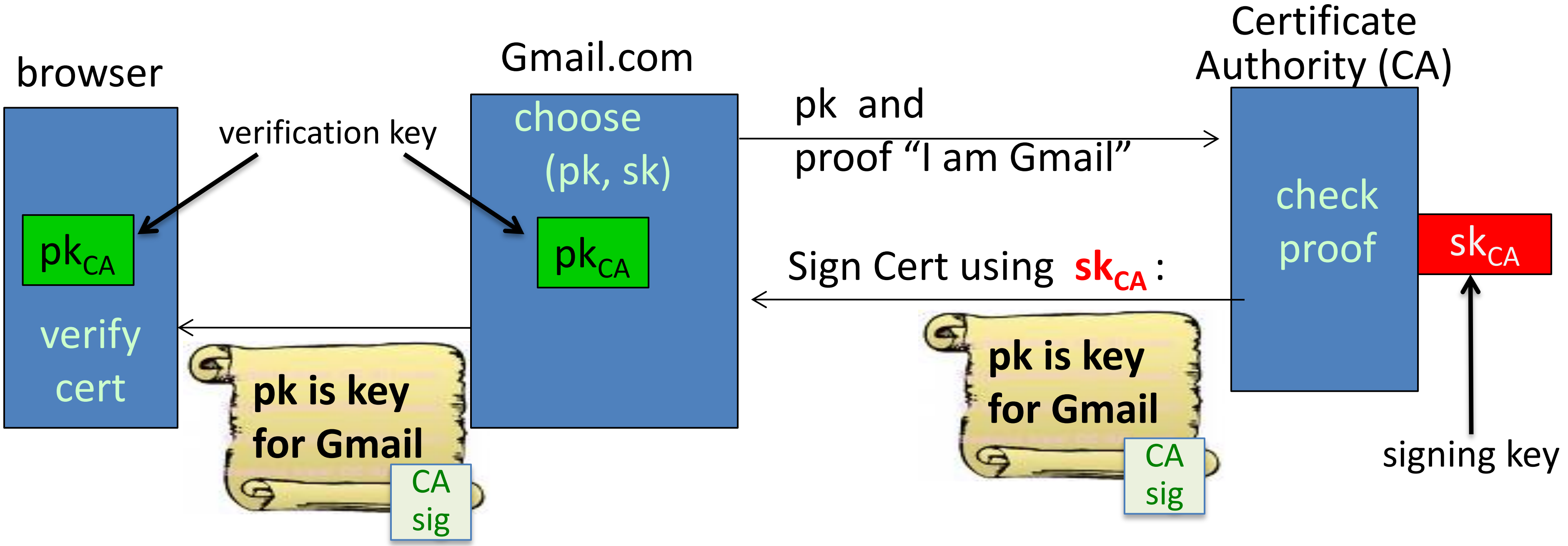
Initial software install is authenticated, but not private



Important application: Certificates

Problem: browser needs server's public-key to setup a session key

Solution: server asks trusted 3rd party (CA) to sign its public-key pk



Server uses Cert for an extended period (e.g. one year)

Certificates: example

Important fields:

Serial Number	5814744488373890497
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Wednesday, July 31, 2013 4:59:24 AM Pacific Daylight Time
Not Valid After	Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
Public Key Info	
Algorithm	Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters	Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key	65 bytes : 04 71 6C DD E0 0A C9 76 …
Key Size	256 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : 8A 38 FE D6 F5 E7 F6 59 …

Equifax Secure Certificate Authority
↳ GeoTrust Global CA
↳ Google Internet Authority G2
↳ mail.google.com

mail.google.com
Issued by: Google Internet Authority G2
Expires: Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
This certificate is valid

Details

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	mail.google.com
Issuer Name	
Country	US
Organization	Google Inc
Common Name	Google Internet Authority G2

What entity generates the CA's secret key sk_{CA} ?

- ○ the browser
- ○ Gmail
- ○ the CA
- ○ the NSA

Applications with few verifiers

EMV payments:

(greatly simplified)

transaction details and signature

Point of Sale terminal

transaction details

(nonce, PIN)

signature

(card cert)

sk

Signed email: sender signs email it sends to recipients

- Every recipient has sender’s public-key (and cert).
  A recipient accepts incoming email if signature verifies.

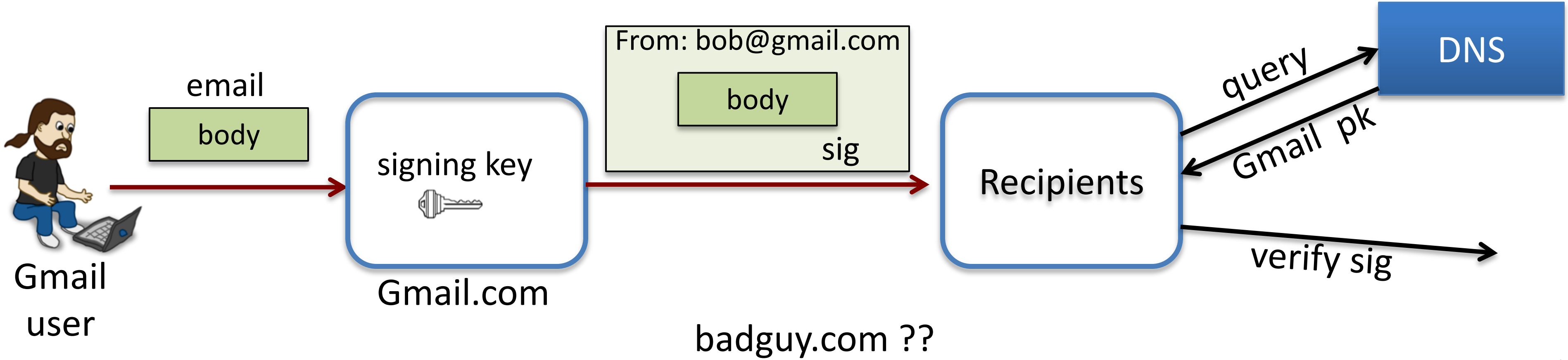
Signing email: DKIM (domain key identified mail)

Problem: bad email claiming to be from **someuser@gmail.com**

but in reality, mail is coming from domain **baguy.com**

⇒ Incorrectly makes gmail.com look like a bad source of email

Solution: **gmail.com** (and other sites) sign every outgoing mail



example DKIM header from gmail.com

X-Google-DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;

d=1e100.net; s=20130820; (lookup 20130820. _domainkey.1e100.net in DNS for public key)

h=x-gm-message-state:mime-version:in-reply-to:references:from:date:
message-id:subject:to:content-type;

bh=MDr/xwte+/JQSgCG+T2R2Uy+SuTK4/gxqdxMc273hPQ=; (hash of message body)

b=dOTpUVOaCrWS6AzmcPMreo09G9viS+sn1z6g+GpC/ArkfMEmcffOJ1s9u5Xa5KC+6K
XRzwZhAWYqFr2a0ywCjbGECBPIE5ccOi9DwMjnvJRYEwNk7/sMzFfx+0L3nTqgTyd0ED
EGWdN3upzSXwBrXo82wVcRRCnQ1yUlTddnHgEoEFg5WV37DRP/eq/hOB6zFNTRBwkvfS
0tC/DNdRwftspO+UboRU2eiWaqJWPjxL/abS7xA/q1VGz0ZoI0y3/SCkxdg4H80c61DU
jdVYhCUd+dSV5flSouLQT/q5DYEjlNQbi+EcbL00liu4o623SDEeyx2isUgcvi2VxTWQ
m80Q==

Gmail’s signature on headers, including DKIM header (2048 bits)

Suppose recipients could retrieve new data from DNS for every email received, could Gmail implement DKIM without signatures?

(ignoring, for now, the increased load on the DNS system)

- ○ Yes, Gmail would write to DNS a collision-resistant hash of every outgoing email. The recipient retrieves the hash from DNS and compares to the hash of the incoming message.
- ○ No, the proposal above is insecure.

⇒ Signatures reduce the frequency that recipients need to query DNS

Applications: summary

- Code signing
- Certificates
- Signed email (e.g. DKIM)
- Credit-card payments: EMV

and many more.

When to use signatures

Generally speaking:

- If one party signs and **one** party verifies: **use a MAC**
  - Often requires interaction to generate a shared key
  - Recipient can modify the data and re-sign it before passing the data to a 3rd party

- If one party signs and **many** parties verify: **use a signature**
  - Recipients **cannot** modify received data before passing data to a 3rd party (non-repudiation)

Review: three approaches to data integrity

1. **Collision resistant hashing**: need a read-only public space

Software Vendor

Small read-only public space

2. **Digital signatures**: vendor must manage a long-term secret key

- Vendor's signature on software is shipped with software
- Software can be downloaded from an untrusted distribution site

3. **MACs**: vendor must compute a new MAC of software for every client

- and must manage a long-term secret key (to generate a per-client MAC key)

End of Segment

Digital Signatures

Constructions overview

Review: digital signatures

Def: a signature scheme (Gen,S,V) is a triple of algorithms:

- Gen(): randomized alg. outputs a key pair (pk, sk)
- S(sk, $m \in M$) outputs sig. σ
- V(pk, m, σ) outputs 'yes' or 'no'

Security:

- Attacker's power: chosen message attack
- Attacker's goal: existential forgery

Extending the domain with CRHF

Let **Sig**=(Gen, S, V) be a sig scheme for short messages, say $M = \{0,1\}^{256}$

Let $H: M^{big} \rightarrow M$ be a hash function (s.g. SHA-256)

Def: $\mathbf{Sig}^{big}$ = (Gen, S^{big} , V^{big}) for messages in M^{big} as:

$$S^{big}(sk, m) = S(sk, H(m)) \quad ; \quad V^{big}(pk, m, \sigma) = V(pk, H(m), \sigma)$$

Thm: If **Sig** is a secure sig scheme for M and H is collision resistant then $\mathbf{Sig}^{big}$ is a secure sig scheme for M^{big}

$\Longrightarrow$ suffices to construct signatures for short 256-bit messages

Suppose an attacker finds two distinct messages m_0, m_1 such that $H(m_0) = H(m_1)$. Can she use this to break **Sig**^{big} ?

- ○ No, **Sig**^{big} is secure because the underlying scheme **Sig** is
- ○ It depends on what underlying scheme **Sig** is used
- ○ Yes, she would ask for a signature on m_0 and obtain an existential forgery for m_1

Primitives that imply signatures: OWF

Recall: $f: X \longrightarrow Y$ is a **one-way function** (OWF) if:

- easy: for all $x \in X$ compute $f(x)$
- inverting f is hard:

Example: $f(x) = AES(x, 0)$

key

Signatures from OWF: Lamport-Merkle (see next module), Rompel

- Signatures are long: $\begin{cases} \text{stateless} \Rightarrow\ > 40\text{KB} \\ \text{stateful} \Rightarrow\ > 4\text{KB} \end{cases}$

Primitives that imply signatures: TDP

Recall: $f: X \longrightarrow X$ is a **trapdoor permutation** (TDP) if:

- easy: for all $x \in X$ compute $f(x)$
- inverting f is hard, **unless one has a trapdoor**

Example: RSA

Signatures from TDP: very simple and practical (next segment)

- Commonly used for signing certificates

Primitives that imply signatures: DLOG

$G = \{1, g, g^2, \ldots, g^{q-1}\}$: finite cyclic group with generator g , $|G| = q$

discrete-log in G is hard if $f(x) = g^x$ is a one-way function

- note: $f(x+y) = f(x) \cdot f(y)$

Examples: $\mathbb{Z}_p^*$ = (multiplication mod p) for a large prime p

$E_{a,b}(\mathbb{F}_p)$ = (group of points on an elliptic curve mod p)

Signatures from DLOG: ElGamal, Schnorr, DSA, EC-DSA, …

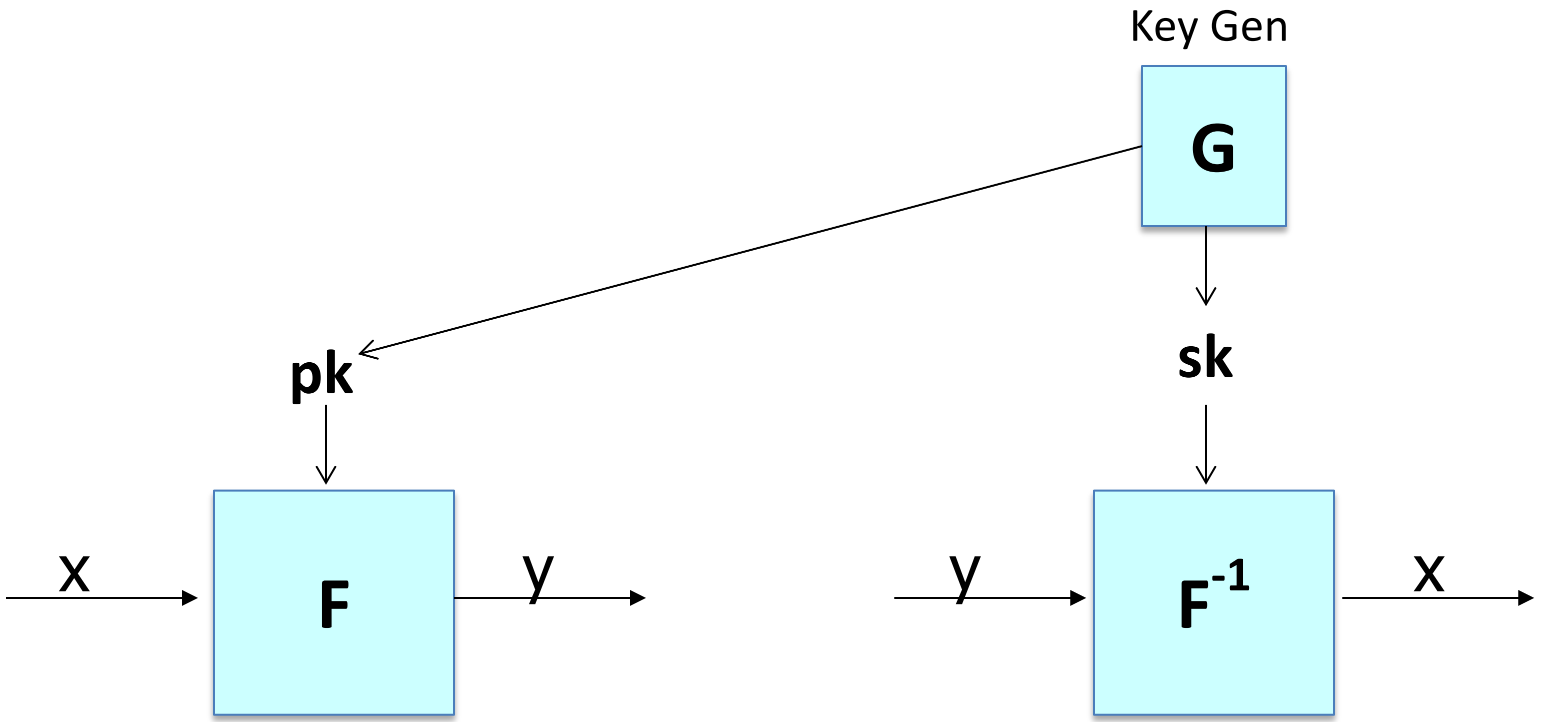
- Will construct these signatures in week 3

End of Segment

Digital Signatures

Signatures From Trapdoor Permutations

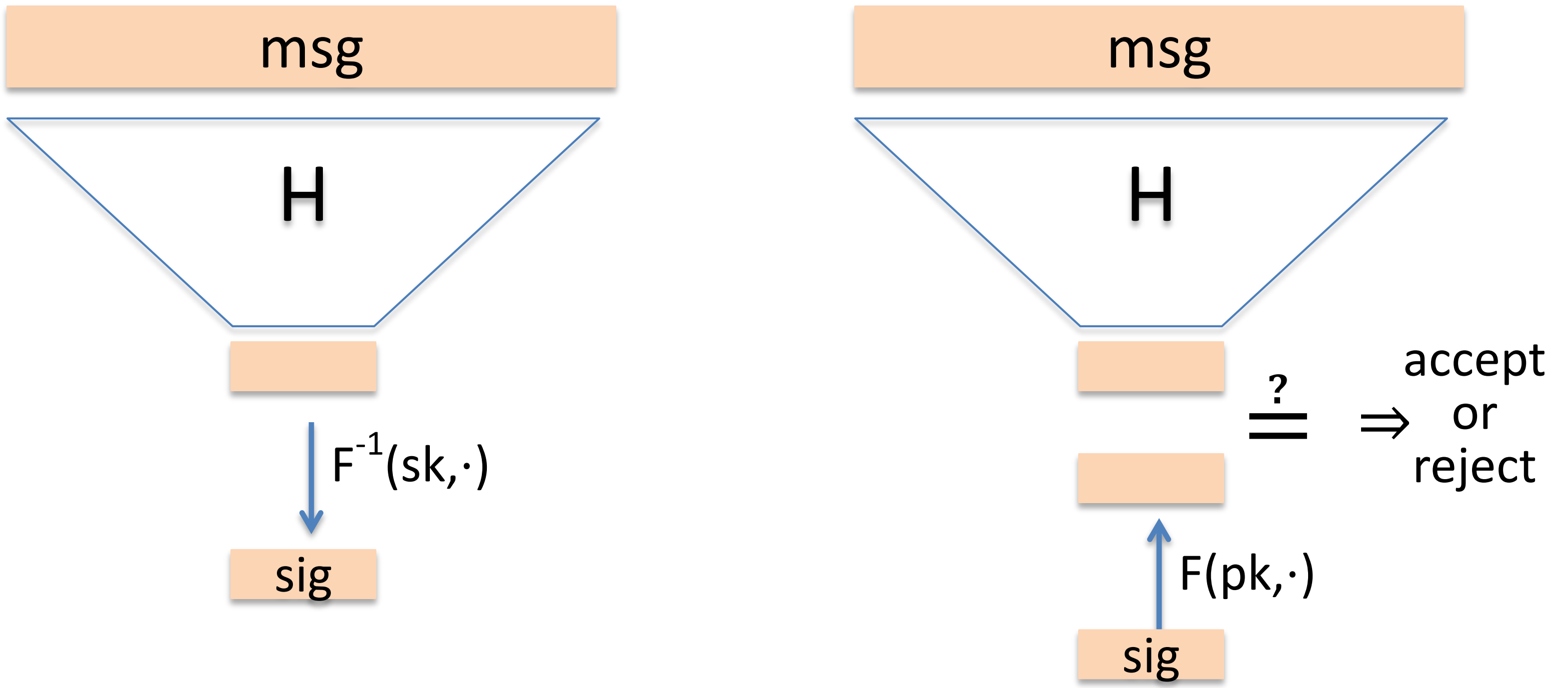
Review: Trapdoor permutation (G, F, F^{-1})



$f(x) = F(pk, x)$ is one-to-one $(X \longrightarrow X)$ and is a **one-way function**.

Full Domain Hash Signatures: pictures

S(sk, msg): **V(pk, msg, sig):**



Full Domain Hash (FDH) Signatures

(G_{TDP}, F, F^{-1}): Trapdoor permutation on domain X

$H: M \longrightarrow X$ hash function (FDH)

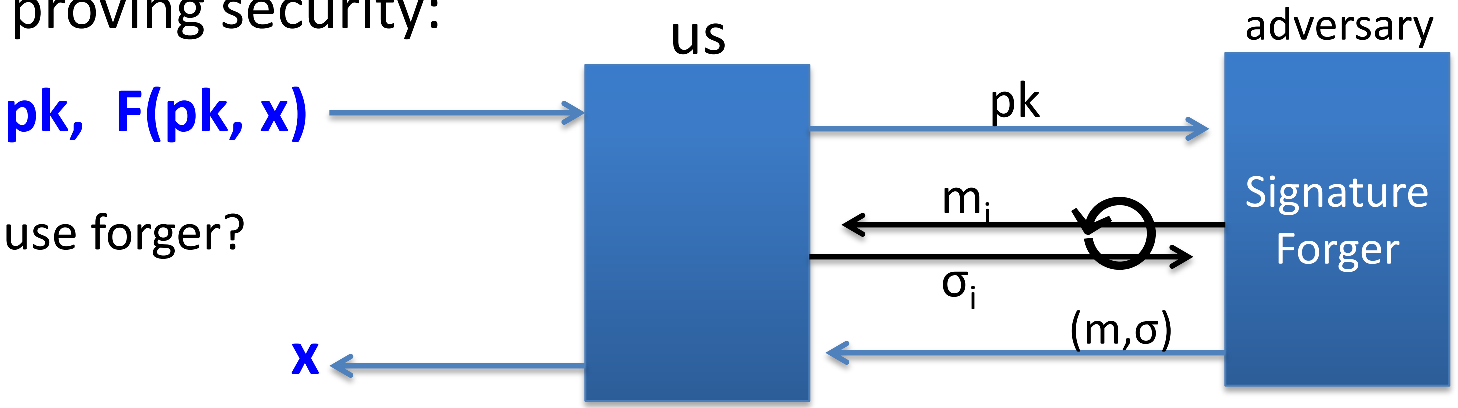
(Gen, S, V) signature scheme:

- **Gen**: run G_{TDP} and output pk, sk
- **S(sk, m∈M)**: output $\sigma \longleftarrow F^{-1}(sk, H(m))$
- **V(pk, m, σ)**: output $\begin{cases} \text{'accept'} & \text{if } F(pk, \sigma) = H(m) \\ \text{'reject'} & \text{otherwise} \end{cases}$

Security

Thm [BR]: **(G_{TDP}, F, F^{-1})** secure TDP ⇒ **(Gen, S, V)** secure signature
when **H:** M ⟶ X is modeled as an "ideal" hash function

Difficulty in proving security:



How can use use forger?

Solution: "we" will know sig. on **all-but-one** of m where adv. queries H().
Hope adversary gives forgery for that single message.

Why hash the message?

Suppose we define NoHash-FDH as:

- **S'(sk, m∈X)**: output **$\sigma \longleftarrow F^{-1}(sk, m)$**
- **V'(pk, m, σ):** output 'accept' if **$F(pk, \sigma) = m$**

Is this scheme secure?

- ◯ Yes, it is not much different than FDH
- ◯ No, for any $\sigma \in X$, σ is a signature forgery for the msg $m=F(pk, \sigma)$
- ◯ Yes, the security proof for FDH applies here too
- ◯ It depends on the underlying TDP being used

RSA-FDH

Gen: generate an RSA modulus $N = p \cdot q$ and $e \cdot d = 1 \bmod \phi(N)$

construct CRHF $H: M \longrightarrow Z_N$

output $pk = (N,e,H)$, $sk = (N,d,H)$

- **S(sk, m∈M)**: output $\sigma \longleftarrow H(m)^d \bmod N$
- **V(pk, m, σ):** output 'accept' if $H(m) = \sigma^e \bmod N$

Problem: having H depend on N is slightly inconvenient

PKCS1 v1.5 signatures

RSA trapdoor permutation: pk = (N,e) , sk = (N,d)

- **S(sk, m∈M):**

$$EM = \underbrace{\overset{\text{16 bits}}{\boxed{01}}\boxed{\texttt{0xFF 0xFF 0xFF} \ \ldots \ \texttt{0xFF 0xFF}}\boxed{00}\overbrace{\boxed{H(m)}}^{\text{256 bits}}}_{\text{RSA modulus size (e.g. 2048 bits)}}$$

output: $\sigma \longleftarrow (EM)^d \bmod N$

- **V(pk, m∈M, σ):** verify that $\sigma^e \bmod N$ has the correct format

Security: no security analysis, not even with ideal hash functions

RSA signatures in practice often use $e=65537$ (and a large d).
As a result, sig verification is ≈20x faster than sig generation.

$e=3$ gives even faster signature verification.

Suppose an attacker finds an $m^* \in M$ such that

EM is a perfect cube (e.g. $8=2^3$, $27=3^3$, $64=4^3$).

Can she use this m^* to break PKCS1?

- ○ Yes, the cube root of EM (over the integers) is a sig. forgery for m^*
- ○ No, this has no impact on PKCS1 signatures
- ○ Yes, but the attack only works for a few 2048-bit moduli N
- ○ It depends on what hash function is begin used

End of Segment

Digital Signatures

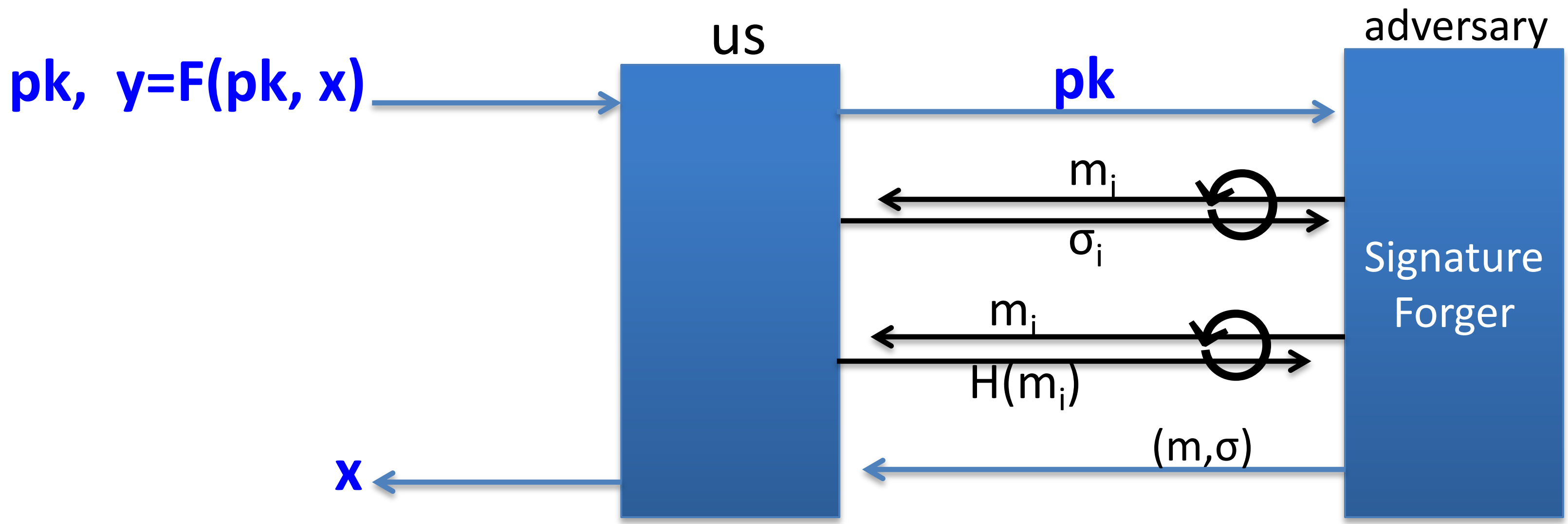
Security Proofs (optional)

Proving security of RSA-FDH

(G, F, F^{-1}): secure TDP with domain X

Recall FDH sigs: $\mathbf{S(sk, m) = F^{-1}(sk, H(m))}$ where $H: M \longrightarrow X$

We will show: TDP is secure ⇒ FDH is secure, when H is a random function



Proving security

Thm [BR]: (G_{TDP}, F, F^{-1}) secure TDP $\Rightarrow$ (G_{TDP}, S, V) secure signature when $\mathbf{H}: M \longrightarrow X$ is modeled as a random oracle.

$$\forall A\ \exists B: \qquad \mathrm{Adv}^{(RO)}_{SIG}[A,FDH] \le q_H \cdot \mathrm{Adv}_{TDP}[B,F]$$

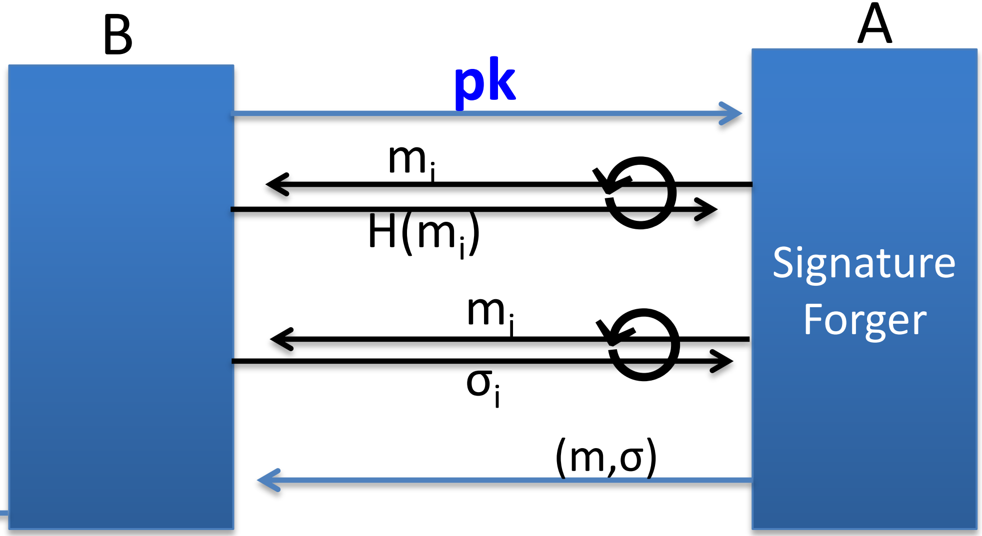
Proof:

pk, y=F(pk, x)

choose $i^* \longleftarrow \{1,\ldots,q_H\}$

if $i \neq i^*$: $x_i \longleftarrow X$,	$H(m_i) = F(pk, x_i)$
else:	$H(m_i) = y$

$m = m_{i^*} \Rightarrow \sigma = F^{-1}(sk, y) = x$



$\Pr[m=m_{i^*}] = 1/q_H$

Proving security

Thm [BR]: (G_{TDP}, F, F^{-1}) secure TDP $\Rightarrow$ (G_{TDP}, S, V) secure signature when **H:** $M \longrightarrow X$ is modeled as a random oracle.

$\forall A\ \exists B$: $\quad \mathrm{Adv}_{SIG}^{(RO)}[A,FDH] \le q_H \cdot \mathrm{Adv}_{TDP}[B,F]$

Proof:

$\Uparrow$

So: $\underbrace{\mathrm{Adv}_{TDP}[B,F]}_{\text{Prob. B outputs x}} \ge \underbrace{(1/q_H)}_{\Pr[m=m_{i^*}]} \cdot \underbrace{\mathrm{Adv}_{SIG}[A,FDH]}_{\text{Prob. forger A outputs valid forgery}}$

Alg. B has table:

$m_1, \quad x_1 : \quad H(m_1) = F(pk, x_1)$

$m_2, \quad x_2 : \quad H(m_2) = F(pk, x_2)$

$\vdots$

$m_{i*}, \qquad\qquad H(m_{i*}) = y$

$\vdots$

$m_q, \quad x_q : \quad H(m_q) = F(pk, x_q)$

How B answers a signature query m_i :

Partial domain hash:

Suppose (G_{TDP}, F, F^{-1}) is defined over domain X = {0,…,B-1}

but **H: M ⟶ {0,…,B/2}** .

Can we prove FDH secure with such an H?

- ◯ No, FDH is only secure with a full domain hash
- ◯ Yes, but we would need to adjust how B defines $H(m_i)$ in the proof
- ◯ It depends on what TDP is used

PSS: Tighter security proof

Some variants of FDH:

tight reduction from forger to inverting the TDP (no q_H factor).
Still assuming hash function H is "ideal."

Examples:

- PSS [BR'96]: part of the PKCS1 v2.1 standard
- KW'03: $S(\,(sk,k),\, m) = \big[\; b \longleftarrow PRF(k,m) \in \{0,1\} \;,\;\; F^{-1}(sk,\; H(b \| m)) \;\big]$
- many others

End of Segment

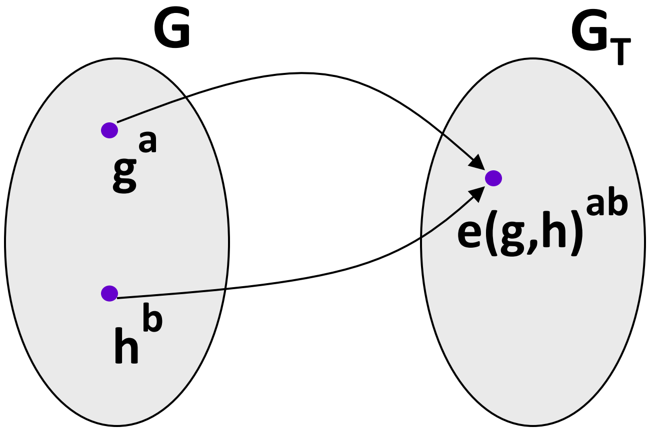
Digital Signatures

Secure Signatures Without Random Oracles

A new tool: pairings

Secure signature without "ideal" hash function (a.k.a. random oracles):

- can be built from RSA, but
- most efficient constructions use **pairings**



G , G_T :   finite cyclic groups $G=\{1,g,\dots,g^{p-1}\}$

Def: A **pairing** $e: G\times G \to G_T$ is a map:

– bilinear: $\mathbf{e(g^a, h^b) = e(g,h)^{ab}}$ $\forall a,b\in Z,\ g,h\in G$

– efficiently computable and non-degenerate:

  g generates G $\Rightarrow$ e(g,g) generates G_T

BLS: a simple signature from pairings

$e: G \times G \to G_T$ a pairing where $|G|=p$, $g \in G$ generator, $H: M \longrightarrow G$

Gen: $sk = (\text{random } \alpha \text{ in } Z_p)$, $pk = g^\alpha \in G$

S(sk, m): output $\sigma = H(m)^\alpha \in G$

V(pk, m, σ): accept if $e(g, \sigma) \stackrel{?}{=} e(pk, H(m))$

Thm: secure assuming CDH in G is hard, when H is a random oracle

Security without random oracles [BB'04]

Gen: sk = (rand. $\alpha, \beta \leftarrow Z_p$) , pk = (g , $y=g^{\alpha} \in G$, $z=g^{\beta} \in G$)

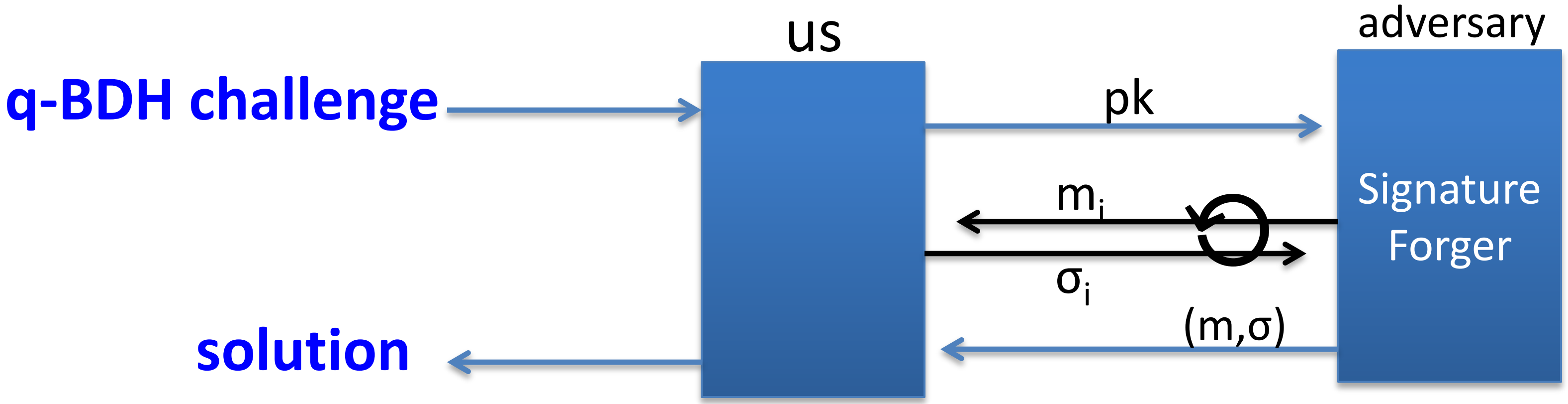
$S(sk, m \in Z_p)$: $r \leftarrow Z_p$, $\sigma = g^{1/(\alpha+r\beta+m)} \in G$, output (r,σ)

$V(pk, m, (r,\sigma))$: accept if $e(\sigma, y \cdot z^r \cdot g^m) \stackrel{?}{=} e(g,g)$

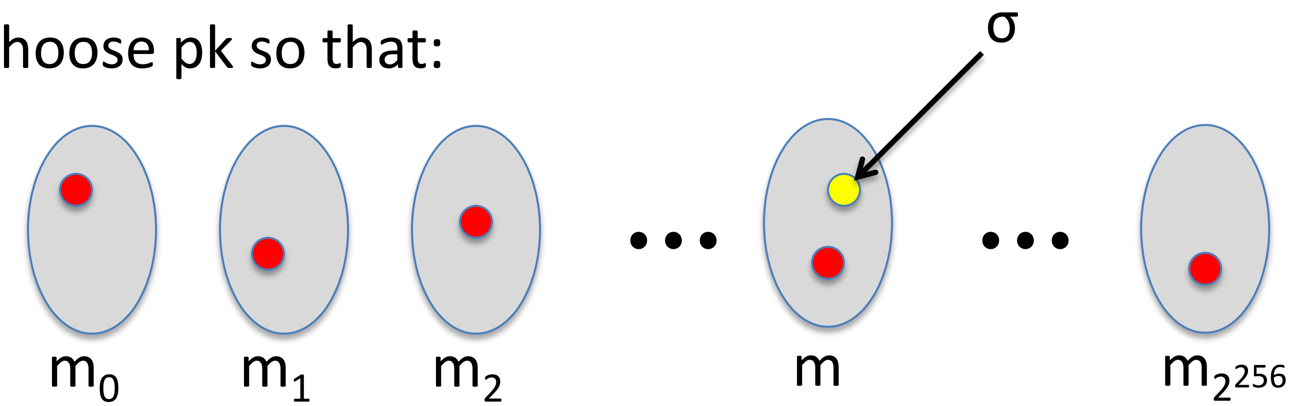
Thm: secure assuming q_S-BDH in G is hard

$$\forall A\ \exists B: \quad Adv_{SIG}[A, BBsig] \le Adv_{q_S\text{-}BDH}[B,G] + (q_S/p)$$

Proof strategy



We choose pk so that:



End of Segment

Digital Signatures

Reducing signature size

Signature lengths

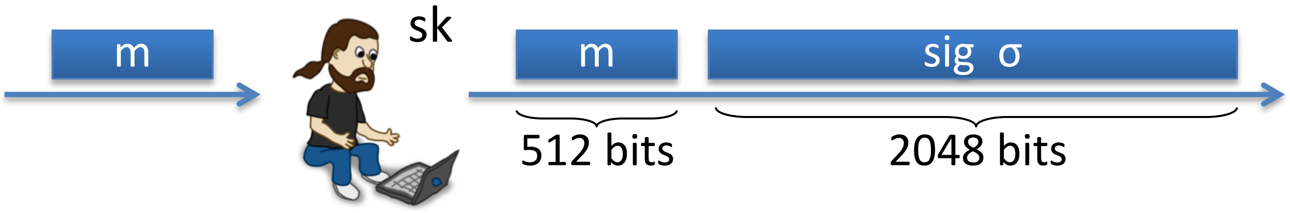
Goal: best existential forgery attack time $\geq 2^{128}$

algorithm	signature size
RSA	2048-3072 bits
EC-DSA	512 bits
Schnorr	384 bits
BLS	256 bits

Open problem: practical 128-bit signatures

Signatures with Message Recovery

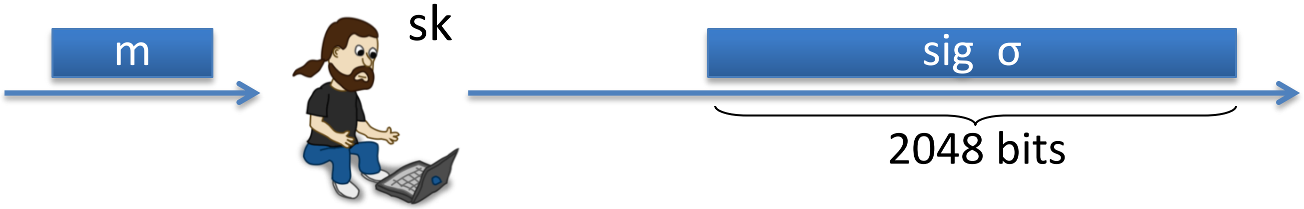
Suppose Alice needs to sign a short message, say $m \in \{0,1\}^{512}$



verifier

$$V(pk, m, \sigma) = \begin{cases} \text{accept} \\ \text{reject} \end{cases}$$

Can we do better?   Yes: signatures with message recovery

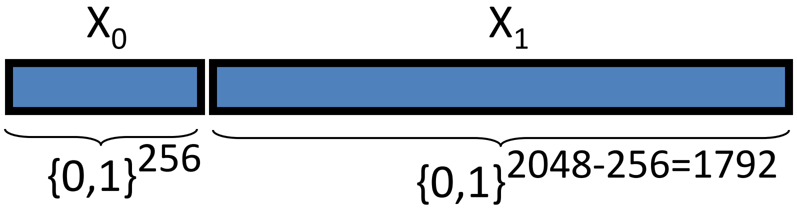


$$\color{blue}{\mathbf{V(pk, \sigma)}} = \begin{cases} \text{accept, } m \\ \text{reject} \end{cases}$$

Security: existential unforgeability under a chosen message attack

Sigs with Message Recovery: Example

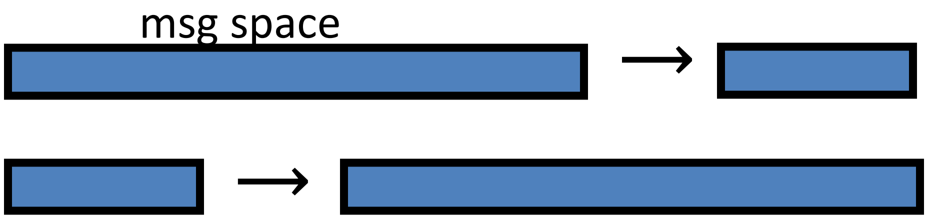
(G_{TDP}, F, F^{-1}): TDP on domain $(X_0 \times X_1)$



Hash functions:

$H: X_1 \longrightarrow X_0$

$G: X_0 \longrightarrow X_1$



Signing: $S(sk, \mathbf{m} \in X_1)$: $h \longleftarrow H(\mathbf{m}) \in X_0$

EM =

$\in X_0 \times X_1$

output: $\sigma \longleftarrow F^{-1}(sk, EM)$

Sigs with Message Recovery: Example

$S(sk,\ m \in X_1)$: choose random $h \leftarrow H(m) \in X_0$

$$\mathrm{EM} = \boxed{\overbrace{\quad h \quad}^{256\text{ bits}}}\boxed{\quad m \oplus G(h) \quad} \in X_0 \times X_1$$

output: $\sigma \leftarrow F^{-1}(sk, \mathrm{EM})$

$V(pk, \sigma)$: $(x_0, x_1) \leftarrow F(pk, \sigma)$, $m \leftarrow x_1 \oplus G(x_0)$

if $x_0 = H(m)$ output "accept, m" else "reject"

Thm: (G_{TDP}, F, F^{-1}) secure TDP $\Rightarrow$ (G_{TDP}, S, V) secure MR signature

when **H, G** are modeled as random oracles

Standard for sigs with message-recovery: **RSA-PSS-R** (PKCS1)

Consider the following MR signature: $S(sk, m) = F^{-1}(sk,\ [m \parallel H(m)]\ )$

$V(pk, \sigma)$: $(m,h) \longleftarrow F(pk, \sigma)$

if $h=H(m)$ outputs "accept, m"

Unfortunately, we can't prove security.

Should we use this scheme with RSA and with H as SHA-256?

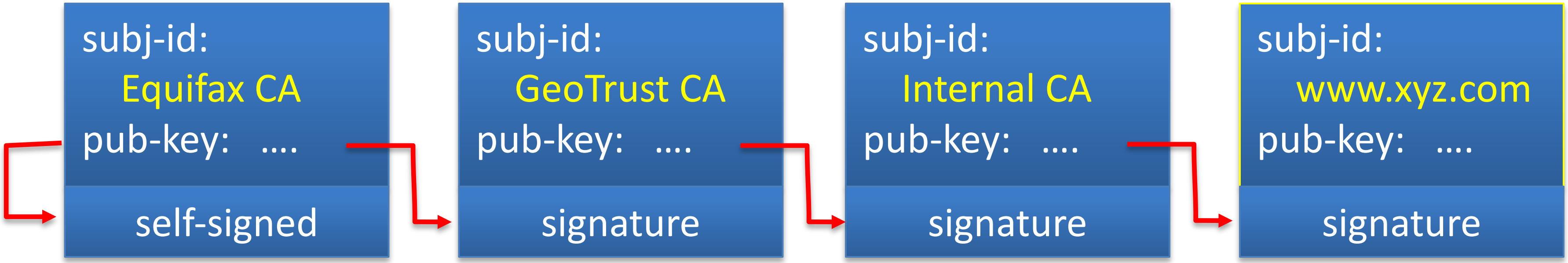
(ISO/IEC 9796-2 sigs. and EMV sigs.)

- ○ Yes, unless someone discovers an attack
- ○ No, only use schemes that have a clear security analysis
- ○ It depends on the size of the RSA modulus

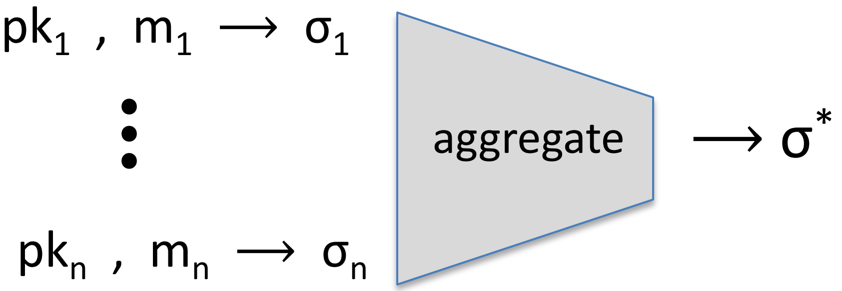
[Practical cryptanalysis of ISO/IEC 9796-2 and EMV signatures, in Proc. of Crypto 2009]

Aggregate Signatures [BGLS'03]

Certificate chain:



Aggregate sigs: lets anyone compress n signatures into <u>one</u>



$V_{agg}(\ \overline{\mathbf{pk}}\ ,\ \overline{\mathbf{m}}\ ,\ \sigma^*\ ) =$ "accept"

means for i=1,…,n:

user i signed msg m_i

Aggregate Signatures [BGLS'03]

Certificate chain with aggregates sigs:

subj-id: Equifax CA pub-key: ….	subj-id: GeoTrust CA pub-key: ….	subj-id: Internal CA pub-key: ….	subj-id: www.xyz.com pub-key: …. aggregate-sig

Aggregate sigs: let us compress n signatures into <u>one</u>

pk_1 , m_1 $\longrightarrow$ σ_1

⋮

pk_n , m_n $\longrightarrow$ σ_n

aggregate $\longrightarrow$ σ^*

$V_{agg}(\ \overline{\mathbf{pk}}\ ,\ \overline{\mathbf{m}}\ ,\ \sigma^*\ )$ = "accept"

means for i=1,…,n:

user i signed msg m_i

Further Reading

- PSS. The exact security of digital signatures: how to sign with RSA and Rabin, M. Bellare, P. Rogaway, 1996.

- On the exact security of full domain hash, J-S Coron, 2000.

- Short signatures without random oracles,
  D. Boneh and X. Boyen, 2004.

- Secure hash-and-sign signatures without the random oracle,
  R. Gennaro, S. Halevi, T. Rabin, 1999.

- A survey of two signature aggregation techniques,
  D. Boneh, C. Gentry, B. Lynn, and H. Shacham, 2003.

End of Segment