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

Bob agrees to pay Alice $1

Verifier

'accept' or 'reject'

Signer

signature

secret signing key (sk)

signing algorithm

public verification key (pk)
A more realistic example

Software vendor

secret signing key (sk)

software update

signing algorithm

untrusted hosting site

clients

verify sig, install if valid

pk

pk
Digital signatures: syntax

Def: a signature scheme \((\text{Gen}, S, V)\) is a triple of algorithms:

- \(\text{Gen}()\): randomized alg. outputs a key pair \((pk, sk)\)
- \(S(sk, m \in M)\) outputs sig. \(\sigma\)
- \(V(pk, m, \sigma)\) outputs ‘accept’ or ‘reject’

Consistency: for all \((pk, sk)\) output by \(\text{Gen}\):

\[
\forall m \in M: \quad V(pk, m, S(sk, m)) = \text{‘accept’}
\]
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 **new** valid message/sig pair $(m, \sigma)$.

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

$\Rightarrow$ attacker cannot produce a valid sig. for a **new** message
Secure signatures

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

\[
(\text{pk}, \text{sk}) \leftarrow \text{Gen}
\]

\[
\text{Chal.}
\]

\[
\text{Adv.}
\]

Adv. wins if \(V(\text{pk}, m, \sigma) = \text{`accept’} \) and \(m \not\in \{m_1, \ldots, m_q\}\)

**Def:** \(SS=(\text{Gen}, S, V)\) is **secure** if for all “efficient” \(A:\)

\[
\text{Adv}_{\text{SIG}}[A, SS] = \text{Pr}[A \text{ wins}] \quad \text{is “negligible”}
\]
Let \((Gen,S,V)\) be a signature scheme.

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

\[ V(pk, m_0, \sigma) = V(pk, m_1, \sigma) \]

for all \(\sigma\) and keys \((pk, sk) \leftarrow Gen\).

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 \(m=\text{“pay Bob 100$”}\) properly signed by Alice, i.e. \(V(pk,m,sig) = \text{‘yes’}\)

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
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) \[\implies\] many-time authenticated channel

Initial software install is authenticated, but not private.

Sender

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

sig\(_1\) \(\leftarrow\) S(sk, m\(_1\))

sig\(_2\) \(\leftarrow\) S(sk, m\(_2\))

Recipients

\(\text{pk}\)

One-time authenticated channel

eavesdrop, but not modify

\(\text{pk}\)
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

Certificate Authority (CA) chooses (pk, sk) for Gmail

Sign Cert using sk_{CA}

pk and proof “I am Gmail”

check proof

pk is key for Gmail

CA sig

pk is key for Gmail

CA sig

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

Important fields:

**Serial Number**: 5814744448373890497

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

**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

Gmail user Gmail.com signing key Gmail.com From: bob@gmail.com body sig Recipients DNS

query Gmail pk verify sig
example DKIM header from gmail.com

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

\[ \text{d=1e100.net; s=20130820; (lookup 20130820._domainkey.1e100.net in DNS for public key)} \]

\[ \text{h=x-gm-message-state:mime-version:in-reply-to:references:from:date:}
\]
\[ \text{message-id:subject:to:content-type;} \]

\[ \text{bh=MDr/xwte+/JQSgCG+T2R2Uy+SuTK4/gxqdxMc273hPQ=; (hash of message body)} \]

\[ \text{b=dOTpUVoaCrWS6AzmcPMreo09G9viS+sn1z6g+GpC/ArkfMEMcfffOJ1s9u5Xa5KC+6KXRzwZhAWYqFr2a0ywCjbGECBPIE5ccOi9DwMjnJRYEwNk7/sMzFfx+0L3nTqgTyd0EDEGWdN3upzSXwBrXo82wVcRRCnQ1yULTddnHgEoEfg5WV37DRP/eq/hOB6zFNTRBwkvfS0tC/DNdRwftspO+UboRU2eiWaqJWPjxL/abS7xA/q1VGz0ZoI0y3/SCkxdg4H80c61DUjdVYhCUd+dSV5fISouLQT/q5DYEjINQbi+EcbL00liu4o623SDEeyx2isUgcvi2VxTWQm80Q==} \]

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

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
Def: a signature scheme $(\text{Gen}, S, V)$ is a triple of algorithms:

- $\text{Gen}()$: randomized alg. outputs a key pair $(pk, sk)$
- $S(sk, m \in M)$ outputs sig. $\sigma$
- $V(pk, m, \sigma)$ outputs ‘yes’ or ‘no’

Security:

- Attacker’s power: chosen message attack
- Attacker’s goal: existential forgery
Extending the domain with CRHF

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

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

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

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

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

⇒ 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 $\text{Sig}^{\text{big}}$?

- No, $\text{Sig}^{\text{big}}$ is secure because the underlying scheme $\text{Sig}$ is.
- It depends on what underlying scheme $\text{Sig}$ 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: \( f: X \rightarrow Y \) is a \textbf{one-way function} (OWF) if:

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

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

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

- Signatures are long: \[
\begin{cases} 
\text{stateless} & \Rightarrow \quad > 40\text{KB} \\
\text{stateful} & \Rightarrow \quad > 4\text{KB}
\end{cases}
\]
Primitives that imply signatures: TDP

Recall: $f : X \rightarrow X$ 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 \((X \rightarrow X)\) and is a one-way function.
Full Domain Hash Signatures: pictures

\[ S(sk, \text{msg}): \]

\[ \text{msg} \]

\[ H \]

\[ F^{-1}(sk, \cdot) \]

\[ \text{sig} \]

\[ V(pk, \text{msg}, \text{sig}): \]

\[ \text{msg} \]

\[ H \]

\[ F(pk, \cdot) \]

\[ ? \]

\[ \Rightarrow \text{accept or reject} \]
Full Domain Hash (FDH) Signatures

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

\(H: M \rightarrow X\) hash function (FDH)

\((\text{Gen}, S, V)\) signature scheme:

- **Gen**: run \(G_{TDP}\) and output \(pk, sk\)
- **S(sk, m \in M)**: output \(\sigma \leftarrow F^{-1}(sk, H(m))\)
- **V(pk, m, \sigma)**: output ‘accept’ if \(F(pk, \sigma) = H(m)\)
  ‘reject’ otherwise
Security

Thm [BR]: \((G_{TDP}, F, F^{-1})\) secure TDP \(\Rightarrow (Gen, S, V)\) secure signature when \(H: M \rightarrow X\) 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:

- \( S'(sk, m \in X) \): output \( \sigma \leftarrow F^{-1}(sk, m) \)
- \( V'(pk, m, \sigma) \): output ‘accept’ if \( F(pk, \sigma) = m \)

Is this scheme secure?

- Yes, it is not much different than FDH
- No, for any \( \sigma \in X \), \( \sigma \) 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 $e \cdot d = 1 \mod \phi(N)$

construct CRHF $H: M \rightarrow \mathbb{Z}_N$

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

• $S(sk, m \in M)$: output $\sigma \leftarrow H(m)^d \mod N$

• $V(pk, m, \sigma)$: output ‘accept’ if $H(m) = \sigma^e \mod N$

**Problem:** having $H$ depend on $N$ is slightly inconvenient
PKCS1 v1.5 signatures

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

- \( S(sk, m \in M) \):
  \[
  EM = \begin{array}{c}
  01 \\
  0xFF \ 0xFF \ 0xFF \ \ldots \ 0xFF \ 0xFF \\
  00 \\
  H(m)
  \end{array}
  \]
  RSA modulus size (e.g. 2048 bits)

  output: \( \sigma \leftarrow (EM)^d \mod N \)

- \( V(pk, m \in M, \sigma) \): verify that \( \sigma^e \mod N \) 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 $\approx 20x$ 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: $S(sk, m) = F^{-1}(sk, H(m))$ where $H: M \rightarrow X$

We will show: TDP is secure $\Rightarrow$ 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 \(H: M \rightarrow X\) is modeled as a random oracle.

\(\forall A \exists B: \text{Adv}^{(RO)}_{\text{SIG}}[A, \text{FDH}] \leq q_H \cdot \text{Adv}_{TDP}[B, F]\)

Proof:

pk, \ y = F(pk, x)

choose \(i^* \leftarrow \{1, \ldots, q_H\}\)

if \(i \neq i^*\): \(x_i \leftarrow X, \ H(m_i) = F(pk, x_i)\)
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]: \((G_{\text{TDP}}, F, F^{-1})\) secure TDP \(\Rightarrow\) \((G_{\text{TDP}}, S, V)\) secure signature when \(H: M \rightarrow X\) is modeled as a random oracle.

\[
\forall A \exists B: \text{Adv}_{\text{SIG}}^{\text{(RO)}}[A, \text{FDH}] \leq q_H \cdot \text{Adv}_{\text{TDP}}[B, F]
\]

Proof:

So:

\[
\text{Adv}_{\text{TDP}}[B, F] \geq \frac{1}{q_H} \cdot \text{Adv}_{\text{SIG}}[A, \text{FDH}]
\]

- Prob. B outputs x
- \(\Pr[m = m_{i^*}]\)
- Prob. forger A outputs valid forgery
Alg. B has table:

$$
\begin{array}{|c|c|c|}
\hline
m_1, & x_1: & H(m_1) = F(pk, x_1) \\
\hline
m_2, & x_2: & H(m_2) = F(pk, x_2) \\
\hline
\vdots & \vdots & \vdots \\
\hline
m_i, & x_i: & H(m_i) = y \\
\hline
\vdots & \vdots & \vdots \\
\hline
m_q, & x_q: & H(m_q) = F(pk, x_q) \\
\hline
\end{array}
$$

How B answers a signature query $m_i$:
Partial domain hash:

Suppose \( (G_{\text{TDP}}, F, F^{-1}) \) is defined over domain \( X = \{0,\ldots,B-1\} \) but \( H: M \rightarrow \{0,\ldots,B/2\} \).

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**: \( S((sk,k), m) = \left[ b \leftarrow \text{PRF}(k, m) \in \{0,1\} \right., \ F^{-1}(sk, H(b\|m)) \right] \)

- 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

\[ G, G_T: \text{ finite cyclic groups } G = \{1, g, \ldots, g^{p-1}\} \]

**Def:** A pairing \( e: G \times G \rightarrow G_T \) is a map:

- bilinear: \( e(g^a, h^b) = e(g,h)^{ab} \quad \forall a, b \in \mathbb{Z}, g, h \in G \)
- efficiently computable and non-degenerate:
  \( g \) generates \( G \quad \Rightarrow \quad e(g,g) \) generates \( G_T \)
BLS: a simple signature from pairings

e: G×G → G_T a pairing where |G|=p, g∈G generator, H: M → G

Gen:  \( sk = (\text{random } \alpha \text{ in } \mathbb{Z}_p) \), \( pk = g^\alpha \in G \)

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

V(pk, m, σ): accept if \( e(g, \sigma) \approx e(pk, H(m)) \)

**Thm:** secure assuming CDH in G is hard, when H is a random oracle
Security without random oracles [BB’04]

Gen: \( \text{sk} = (\text{rand. } \alpha, \beta \leftarrow \mathbb{Z}_p) \), \( \text{pk} = (g, y=g^\alpha \in G, z=g^\beta \in G) \)

\( S(\text{sk}, m \in \mathbb{Z}_p): \ r \leftarrow \mathbb{Z}_p, \ \sigma = g^{1/\alpha+r\beta+m} \in G \), output (r,\sigma)

\( V(\text{pk}, m, (r,\sigma)): \ \text{accept if} \ e(\sigma, y \cdot z^r \cdot g^m) \equiv e(g,g) \)

Thm: secure assuming \( q_s\)-BDH in G is hard

\[ \forall A \exists B : \ \text{Adv}_{\text{SIG}}[A,\text{BBsig}] \leq \text{Adv}_{q_s\text{-BDH}}[B,G] + \left( \frac{q_s}{p} \right) \]
Proof strategy

q-BDH challenge

solution

We choose pk so that:

\( m_0 \) \( m_1 \) \( m_2 \) \( \ldots \) \( m \) \( \ldots \) \( m_{2^{256}} \)
End of Segment
Digital Signatures
Reducing signature size
Signature lengths

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

<table>
<thead>
<tr>
<th>algorithm</th>
<th>signature size</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA</td>
<td>2048-3072 bits</td>
</tr>
<tr>
<td>EC-DSA</td>
<td>512 bits</td>
</tr>
<tr>
<td>Schnorr</td>
<td>384 bits</td>
</tr>
<tr>
<td>BLS</td>
<td>256 bits</td>
</tr>
</tbody>
</table>

Open problem: practical 128-bit signatures
Signatures with Message Recovery

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

Can we do better? Yes: signatures with message recovery.

Security: existential unforgeability under a chosen message attack.
Sigs with Message Recovery: Example

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

Hash functions:

\[ H: X_1 \rightarrow X_0 \]
\[ G: X_0 \rightarrow X_1 \]

Signing: \[ S(sk, m \in X_1): h \leftarrow H(m) \in X_0 \]

\[ EM = h \oplus m \oplus G(h) \in X_0 \times X_1 \]

Output: \[ \sigma \leftarrow F^{-1}(sk, EM) \]
Sigs with Message Recovery: Example

\[ S(sk, m\in X_1): \text{ choose random } h \leftarrow H(m) \in X_0 \]

\[ EM = \begin{array}{c|c}
  h & m \oplus G(h) \\
\end{array} \in X_0 \times X_1 \]

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

\[ V(pk, \sigma): (x_0, x_1) \leftarrow F(pk, \sigma), \quad m \leftarrow x_1 \oplus G(x_0) \]

if \( x_0 = H(m) \) output “accept, m” else “reject”

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

when \( H, G \) are modeled as random oracles
Consider the following MR signature:

\[ S(\text{sk}, m) = F^{-1}(\text{sk}, [m \| H(m)]) \]

\[ V(\text{pk}, \sigma): \ (m,h) \leftarrow F(\text{pk}, \sigma) \]

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?

- 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

Certificate chain:

 Aggregate sigs: lets anyone compress \( n \) signatures into one

\[
\text{pk}_1, \ m_1 \rightarrow \sigma_1 \\
\vdots \\
\text{pk}_n, \ m_n \rightarrow \sigma_n \\
\text{aggregate} \rightarrow \sigma^* \\
V_{\text{agg}}(\text{pk}, m, \sigma^*) = \text{“accept”} \\
\text{means for } i=1,\ldots,n: \text{ user } i \text{ signed msg } m_i
\]
Aggregate Signatures

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 sigs: let us compress n signatures into one

\[ pk_1, m_1 \rightarrow \sigma_1 \]
\[ \vdots \]
\[ pk_n, m_n \rightarrow \sigma_n \]

\[ \text{aggregate} \rightarrow \sigma^* \]

\[ V_{agg}( pk, m, \sigma^* ) = "accept" \]

means for i=1,...,n:
user i signed msg \( m_i \)
Further Reading


End of Segment