Two Signature Schemes

Ron Rivest

MIT/LCS
Outline

- Review signatures, RSA, security, ...

- **Prefix aggregation scheme**: can compute $\sigma(x)$ from $\sigma(x_0)$ and $\sigma(x_1)$
  (joint with Suresh Chari & Tal Rabin)

- **Transitive signature scheme**: can compute $\sigma(A,C)$ from $\sigma(A,B)$ and $\sigma(B,C)$
  (joint with Silvio Micali)

\[ \text{Diagram:} \]

- $x \xrightarrow{\sigma} x_0 \xrightarrow{\sigma} x_1$
- $A \xrightarrow{\sigma} B \xrightarrow{\sigma} C$
Digital Signatures

- **Key generation** → public key $PK$  
  → secret key $SK$

- **Signing procedure**
  
  given message $m$
  
  signature is $\sigma(m, SK)$
  
  (or $\sigma(m)$ when $SK$ understood)

- **Verification procedure**
  
  $V(m, PK, s) = \text{true} \iff$
  
  $s = \sigma(m, SK)$
Digital Signature Security

A digital signature scheme is secure if adversary can not forge signature for any new message m, even if adversary knows PK (but not SK) & can first obtain valid signatures for any messages (other than m) that he wishes.
Basic RSA

- **Keygen:**
  - $p, q$ large primes
  - $n = p \cdot q$
  - $e$ public exponent
  - $d$ secret exponent
  - $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$
  - $PK = (n, e)$, $SK = d$

- **Signing:**
  - $\sigma(m) = m^d \pmod{n}$

- **Verification:**
  - $s^e \equiv m \pmod{n}$
RSA is multiplicative

- $\sigma(x) \cdot \sigma(y) = \sigma(xy)$

- This can be useful! (or dangerous!)

E.g. to get your blind signature on $m$:

- I give you $m \cdot \rho^e$ to sign
- Your sig = $(m \cdot \rho^e)^d = m^d \cdot \rho$
- I divide by $\rho$ to get $m^d = \sigma(m)$

You have no idea what $m$ is.

- Useful for e-cash, voting.
"Signature Algebras"

For what other operations on message space can we find corresponding signature scheme, such that $\sigma(x)$ and $\sigma(y)$ can be combined (by anyone) to obtain $\sigma(x \circledast y)$?

- Note: need to modify def. of security
Open Problem

Let $xy$ denote concatenation. Is there a signature scheme $\sigma$ such that anyone can compute $\sigma(xy)$ from $\sigma(x)$ and $\sigma(y)$? E.g., combine $\sigma(ababb)$ and $\sigma(aba)$ to get $\sigma(ababbaba)$?
Prefix Aggregation

Is there a signature scheme $\sigma$ such that anyone can compute $\sigma(x)$ from $\sigma(x0)$ and $\sigma(x1)$?

Signatures of both children $\implies$ signature of parent
**Motivation: Routing**

10****

B says: "I can route to IP addresses of the form 100***"

C says: "I can route to IP addresses of the form 101***"

⇒

A says: "I can route to IP addresses of the form 10****"
Scheme

- Uses basic RSA
- Let $H$ be a hash fn onto $\mathbb{Z}_n^*$
- Define label $\lambda(x)$ for each node $x$:
  \[
  \begin{align*}
  \lambda(\varepsilon) &= H(\varepsilon) \\
  \lambda(x_0) &= H(x_0) \\
  \lambda(x_1) &= \lambda(x)/\lambda(x_0)
  \end{align*}
  \]

  So $\lambda(x) = \lambda(x_0) \cdot \lambda(x_1) \mod n$

- Define $\sigma(x) = \lambda(x)^d \mod n$
  \[\text{so } \sigma(x) = \sigma(x_0) \cdot \sigma(x_1) \mod n\]

- Fact that you can compute $\sigma(x_1)$ from $\sigma(x)$ and $\sigma(x_0)$ not a problem in this application!
Security

• Assume it is hard to compute \( \sigma(m) = m^d \pmod{n} \) given \( n, e, \) and \( m \). (Basic RSA)

• Then it is hard for adversary to forge signature \( \sigma(x) \) in scheme, even if adversary can adaptively ask for signatures on other nodes, assuming \( \sigma(x) \) not implied by what has been asked for and relation \( \sigma(x) = \sigma(x_0) \cdot \sigma(x_1) \)
Open Problems

- Do "AND" in a clean way, so that $\sigma(A)$ can be computed from $\sigma(B)$ and $\sigma(C)$.
  (But $\sigma(B)$ is not computable from $\sigma(A)$ and $\sigma(C')$.)

- Do "OR"

- Do formulae & circuits built from AND's and OR's.
**Signing Graphs**

- A graph $G = (V, E)$ has a finite set $V$ of vertices and a set $E \subseteq V \times V$ of edges. (May be directed or undirected)

- Graphs are widely used representation.

- We are interested in secure (authenticated, signed) representation of graphs when graph has certain properties.
Transitive Closure

- A graph is transitive if
  \[(u, v) \in E \quad \Rightarrow \quad \exists (v, w) \in E \quad \Rightarrow \quad (u, w) \in E\]

- Many graphs are naturally closed transitively (and reflexively):
  administrative domains
  chain of command
Transitive Signature Schemes

- A transitive signature scheme is a way of signing vertices ($\sigma(v)$) and edges ($\sigma(u,v)$) such that given $\sigma(u,v)$ and $\sigma(v,w)$, one can compute $\sigma(u,w)$.

- Imagine some issuer signs various vertices and edges over time...

- Inferred signature $\sigma(u,w)$ should be indistinguishable from an original issuer sig.

- Provides efficiency for issuer & verifier.
TSS for undirected graphs

- \( p = 2q + 1 \) large prime
- \( g, h \) elements of \( \mathbb{Z}_p^* \) of order \( q \)
- \( \log_g(h) \) infeasible to compute (DLP)

- For vertex \( i \), issuer computes
  \[
  v_i = g^{x_i} h^{y_i} \pmod{p}
  \]
  \( v_i \) public (assigned), \( x_i \) & \( y_i \) secret, random

- For edge \((i, j)\), issuer's sig is
  \[
  (\Delta x, \Delta y) = (x_i - x_j, y_i - y_j) \mod q
  \]
- Verify edge:
  \[
  \frac{v_i}{v_j} = g^{\Delta x} h^{\Delta y} \pmod{p}
  \]
Transitivity

\[ \sigma(i, k) = \sigma(i, j) + \sigma(j, k) \]
Security

Theorem: Assuming that discrete logarithm problem is hard, an adversary cannot forge a signature on an edge not already signed and not implied by transitivity, even if he can adaptively request edge signatures first.

Proof sketch: Given DLP instance \( \log_g (h) \mod p \), simulate adversaries view. Can answer all signature requests with knowing \( \alpha = \log_g (h) \). Representations unknown to adversary by multiplof \((-\alpha,1)\), since \( g^{-\alpha}h = 1 \). But

\[
g^{\Delta x} h^{\Delta y} = g^{\Delta x'} h^{\Delta y'} \Rightarrow \alpha = \frac{\Delta x - \Delta x'}{\Delta y' - \Delta y}
\]
Open Problem 1

Find a secure directed transitive signature scheme.
Open Problem II

Assume vertices = public keys

Find a TSS such that only B can create $\sigma(A,C)$ from $\sigma(A,B)$ and $\sigma(B,C)$.

(Delegation; SPKI/SDSI tuple reduction)