Signature Schemes

CSG 252 Fall 2006

Riccardo Pucella

Signatures

- Signatures in “real life” have a number of properties
  - They specify the person “responsible” for a document
    - E.g. that it has been produced by the person, or that the person agrees with the document
  - Attached to a particular document
  - Easily verifiable by third parties

- We want a similar mechanism for digital documents
- Some difficulties:
  - Need to bind signature to document
  - Need to ensure verifiability (and avoid forgeries)

Formal Definition

A signature scheme is a tuple (P,A,K,S,V) where:

- P is a finite set of possible messages
- A is a finite set of possible signatures
- K (the keyspace) is a finite set of possible keys
- For all k, there is a signature algorithm sig_k in S and a verification algorithm ver_k in V such that
  - $sig_k : P \rightarrow A$
  - $ver_k : P \times A \rightarrow \{true, false\}$
  - $ver_k(x,y)$ = true iff $y = sig_k(x)$

- A pair $(x,y) \in P \times A$ is called a signed message

Example: RSA Signatures

- The RSA cryptosystem (in fact, most public key cryptosystems) can be used as a signature scheme
- Take:
  - $sig_k(x) = d_k(x)$
  - $ver_k(x,y) = (x =? e_k(y))$
  - Only user can sign (because decryption is private)
  - Anyone can verify (because encryption is public)

Signing and Encrypting

- Suppose you want to sign and encrypt a piece of data
  - Where encryption is public key (why is this important?)
  - Public key cryptography does not say anything about the sender

- Two possibilities:
  - First encrypt, then sign: $x \rightarrow (e_{ke}(x), sig_{ks}(e_{ke}(x)))$
    - But adversary could replace by $sig_{ke'}(e_{ke}(x))$ making it seem the message came from someone else
  - First sign, then encrypt: $x \rightarrow (e_{ke}(x), sig_{ks}(x))$
    - Better make sure signature does not leak info!

Possible Attacks

- (Alice is the signer, Oscar the attacker)
- Key-only attack
  - Oscar possesses Alice's public verification algorithm
- Known message attack
  - Oscar possesses a list of signed messages (x_i, y_i)
- Chosen message attack
  - Oscar queries Alice for the signatures of a list of messages x_i

Possible Adversarial Goals

- Total break
  - Oscar can derive Alice's private signing algorithm
- Selective forgery
  - Oscar can create a valid signature on a message chosen by someone else, with some non-negligible probability
- Existential forgery
  - Oscar can create a valid signature for at least one message

Some Comments

- Cannot have unconditional security, only computational or provable security
- Attacks above are similar to those against MACs
  - We mostly concentrated on existential forgeries against chosen message attacks
  - What Graham was asking was for total break against (some) attacks
- Existential forgeries against chosen message attacks:
  - Least damage against worst attacker
  - The minimum you should ask for

Security of RSA Signatures

- Existential forgery using a key-only attack:
  - Choose a random y
  - Compute $x = e_k(y)$
  - We have $y = sig_k(x)$, a valid signature of x
- Existential forgery using a known-message attack:
  - Suppose $y = sig_k(x)$ and $y' = sig_k(x')$
  - Can check $e_k(y\, y' \bmod n) = x\, x' \bmod n$
  - So $y\, y' \bmod n = sig_k(x\, x' \bmod n)$
- Existential forgery using a chosen message attack:
  - To get a signature for x, find $x_1 x_2 = x \bmod n$
  - Query for signatures of x_1 and x_2
  - Apply previous attack

Signatures and Hashing

- The easiest way to get around the above problems is to use a cryptographic hash function
  - Given message x
  - Produce digest h(x)
  - Sign digest h(x) to create $(x, sig_k(h(x)))$

- To verify:
  - Get (x,y)
  - Compute h(x)
  - Check $ver_k\,(h(x),y)$

Use of Hashing for Signatures

- Existential forgery using a chosen messge attack
  - Oscar finds x,x′ s.t. h(x)=h(x′)
  - He gives x to Alice and gets her to sign h(x)
  - Then (x′,sig_K(h(x))) is a valid signed message
  - Prevented by having h collision resistant

- Existential forgery using a known message attack
  - Oscar starts with (x,y), where y = sig_K(h(x))
  - He computes h(x) and tries to find x′ s.t. h(x′) = h(x)
  - Prevented by having h second preimage resistant

- Existential forgery using a key-only attack
  - (If signature scheme has existential forgery using a key-only attack)
  - Oscar chooses message digest and finds a forgery z for it
  - Then tries to find x s.t. h(x)=z
  - Prevented by having h preimage resistant

Example: ElGamal Signature Scheme

- Let p be a prime s.t. discrete log in Z_p is hard
- Let a be a primitive element in Z_p^*
- $P = Z_p^*$, $A = Z_p^* \times Z_{p-1}$
- $K = \{(p,\alpha,a,\beta) \mid \beta = \alpha^a \pmod{p}\}$

- For $k = (p,\alpha,a,\beta)$ and $t \in Z_{p-1}^*$
  - $\gamma = \alpha^t \bmod p$
  - $sig_k(x,t) = (\gamma, (x-a\gamma)t^{-1} \pmod{p-1})$

  - $ver_k(x,(\gamma,\delta)) = (\ \beta^\gamma \gamma^\delta =?\ \alpha^x \pmod{p}\ )$

Security of ElGamal Scheme

- Forging a signature (γ,δ) without knowing a
  - Choosing γ and finding corresponding δ amounts to finding discrete log
  - Choosing δ and finding corresponding γ amounts to solving $\beta^{\gamma}\gamma^{\delta} = \alpha^{x} \pmod{p}$
    - No one knows the difficulty of this problem (believed to be hard)
  - Choosing γ and δ and solving for the message amounts to finding discrete log
  - Existential forgery with a key-only attack:
    - Sign a random message by choosing γ, δ and message simultaneously (p.289)

Variant 1: Schnorr Signature Scheme

- ElGamal requires a large modulus p to be secure
- A 1024 bit modulus leads to a 2048 bit signature
  - Too large for some uses of signatures (smartcards)

- Idea: use a subgroup of Z_p of size q (q << p)
- Let p be a prime s.t. discrete log is hard in Z_p^*
- Let q be a prime that divides p-1
- Let α in Z_p^* be a q-th root of 1 mod p
- Let $h : \{0,1\}^* \rightarrow Z_q$ be a secure hash function
- $P = \{0,1\}^*$, $A = Z_q \times Z_q$
- $K = \{(p,q,\alpha,a,\beta) \mid \beta = \alpha^a \pmod{p}\}$
- For $k=(p,q,\alpha,a,\beta)$ and $1 \leq t \leq q-1$:
  - $\gamma = h(x \,\|\, \alpha^t \bmod p)$
  - $sig_k(x,t) = (\gamma, t+a\gamma \bmod q)$
  - $ver_k(x,(\gamma,\delta)) = (h(x \,\|\, \alpha^\gamma\beta^{-\gamma} \bmod p) =?\ \gamma$

Variant 2: DSA

- Let p be a prime s.t. discrete log is hard in Z_p
  - bitlength of p = 0 (mod 64), 512 ≤ bitlength ≤ 1024
- Let q be a 160 bit prime that divides p-1
- Let α in Z_p^* be a q-th root of 1 mod p
- Let $h : \{0,1\}^* \to Z_q$ be a secure hash function
- $P = \{0,1\}^*$, $A = Z_q^* \times Z_q^*$
- $K = \{(p,q,\alpha,a,\beta) \mid \beta = \alpha^a \pmod{p}\}$
- For $k=(p,q,\alpha,a,\beta)$ and $1 \le t \le q-1$:
  - $\gamma = (\alpha^t \bmod p) \bmod q$
  - $sig_k(x,t) = (\gamma, (SHA1(x)+a\gamma)t^{-1} \bmod q)$
  - $ver_k(x,(\gamma,\delta)) = (\alpha^{e1}\beta^{e2} \bmod p) \bmod q =? \gamma$
    - $e1 = SHA1(x)\delta^{-1} \bmod q$
    - $e2 = \gamma\delta^{-1} \bmod q$

Variant 3: Elliptic Curve DSA

- Modification of the DSA to use elliptic curves
- Instead of choosing α, β, use A and B two points on an elliptic curve over Z_p
- Roughly speaking, instead of: $(\alpha^t \bmod p) \bmod q$
  use the x coordinate of the point tA, mod q
- The rest of the computation is as before

Provably Secure Signature Schemes

- The previous examples were (to the best of our knowledge) computationally secure signature scheme
- Here is a provably secure signature scheme
  - As long as only one message is signed

- Let m be a positive integer
- Let $f : Y \rightarrow Z$ be a one-way function
- $P = \{0,1\}^m$, $A = Y^m$
- Choose $y_{i,j}$ in Y at random for $1 \le i \le m$, $j=0,1$
- Let $z_{i,j} = f(y_{i,j})$
- A key = 2m y's and 2m z's (y's private, z's public)
  - $sig_k(x_1,\ldots,x_m) = (y_{1,x1},\ldots,y_{m,xm})$
  - $ver_k((x_1,\ldots,x_m),(a_1,\ldots,a_m)) = (f(a_i) =? z_{i,xi})$ for all i

Argument for Security

- Argument for provable security:
  - Existential forgeries using a key-only attack
    - Assume that f is a one-way function
    - Show that if there is an existential forgery using a key-only attack, then there is an algorithm that finds preimage of random elements in the image of f with probability at least 1/2
- We need the restriction to one signature only
  - If the attacker gets two messages signed with the same key, then can easily construct signatures for other messages
  - (0,1,1) and (1,0,1) can give signatures for (0,0,1), (1,1,1)

Undeniable Signature Schemes

- Introduced by Chaum and van Antwerpen in 1989
  - A signature cannot be verified without the signer
  - Prevent signer from disavowing signature

- Let p,q primes, $p = 2q+1$, and discrete log hard in Z_p^*
- Let α in Z_p^* be an element of order q
- G = multiplicative subgroup of Z_p^* of order q
- P = A = G
- $K = \{(p,\alpha,a,\beta\} \mid \beta = \alpha^a \bmod p\}$
- For key $k=(p,\alpha,a,\beta)$ and x in G:
  - $sig_k(x) = x^a \bmod p$
- To verify (x,y): pick e_1,e_2 at random in Z_q
  - Compute $c = y^{e1}\beta^{e2}$
  - Signer computes $d = c^{inv(a) \bmod q} \bmod p$ (where $inv(a) = a^{-1}$)
  - y is a valid signature iff $d = x^{e1}\alpha^{e2} \bmod p$

Disavowal Protocol

- Can prove that Alice cannot fool Bob into accepting a fraudulent signature (except with very small probability = 1/q)

- What if Bob wants to make sure that a claimed forgery is one?
  1. Bob chooses e_1, e_2 at random in Z_q^*
  2. Bob computes $c = y^{e1}\beta^{e2}$ mod p; sends it to Alice
  3. Alice computes $d = c^{inv(a) \bmod q}$ mod p; sends it to Bob
  4. Bob verifies $d \neq x^{e1}\alpha^{e2}$ mod p
  5. Bob chooses f_1, f_2 at random, in Z_q^*
  6. Bob computes $C = y^{f1}\beta^{f2}$ mod p; sends it to Alice
  7. Alice computes $D = C^{inv(a) \bmod q}$ mod p; sends it to Bob
  8. Bob verifies $D \neq x^{f1}\alpha^{f2}$ mod p
  9. Bob concludes y is a forgery iff $(d\alpha^{-e2})^{f1} = (D\alpha^{-f2})^{e1}$ mod p

Why Does This Work?

- Alice can convince Bob that an invalid signature is a forgery
  - If $y \neq x^a \bmod p$ and Alice and Bob follow the protocol, then the check in last step succeeds
- Alice cannot make Bob believe that a valid signature is a forgery except with a very small probability
  - Intuition: since she cannot recover e_1, e_2, f_1, f_2, she will have difficulty coming up with d and D that fail steps 4 and 8, but still pass step 9
  - See Stinson for details