

Cryptography CS 555

Lecture 15



Department of Computer Sciences
Purdue University

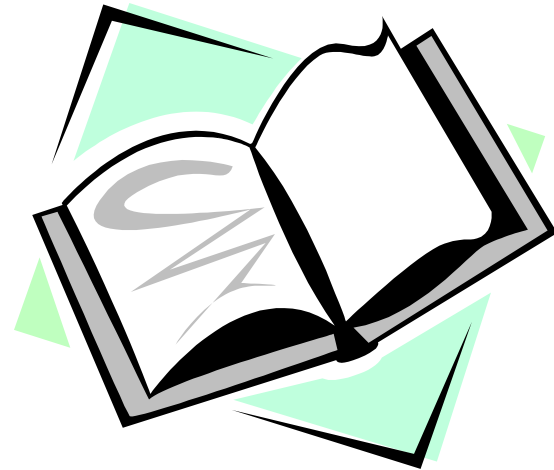
Lecture Outline

- Digital signatures
 - RSA
 - ElGamal
 - DSA
 - Schnorr
- One-time digital signatures
 - Rabin
 - Lamport
 - Merkle



Recommended Reading

- Stinson, Chapter 7 or
- HAC, Chapter 11



The Problem

- Consider the real-life example where a person pays by credit card and signs a bill; the seller verifies that the signature on the bill is the same with the signature on the card
- Contracts, they are valid if they are signed.
- Can we have a similar service in the electronic world?

Digital Signatures

- Digital Signature: a data string which associates a message with some originating entity.
- Digital Signature Scheme: for each key, there is a **SECRET signature generation algorithm** and a **PUBLIC verification algorithm**
- Services provided:
 - Authentication
 - Data integrity
 - Non-Repudiation (MAC does not p

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

Attack Models for Digital Signatures

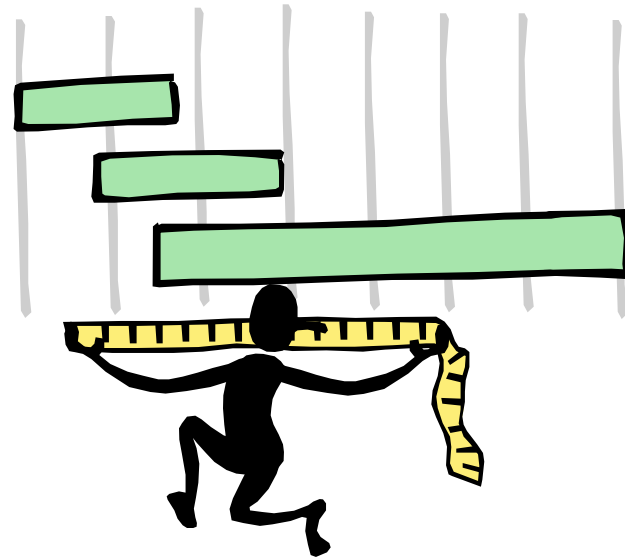
- **Key-only attack:** Adversary knows only the verification function (which is supposed to be public).
- **Known message attack:** Adversary knows a list of messages previously signed by Alice.
- **Chosen message attack:** Adversary can choose what messages wants Alice to sign, and he knows both the messages and the corresponding signatures.

Adversarial Goals

- **Total break**: adversary is able to find the secret for signing, so he can forge then any signature on any message.
- **Selective forgery**: adversary is able to create valid signatures on a message chosen by someone else, with a significant probability.
- **Existential forgery**: adversary can create a pair (message, signature), s.t. the signature of the message is valid.
- A signature scheme can not be perfectly secure; it can only be computationally secure.
- Given enough time and adversary can always forge Alice's signature on any message.

Digital Signatures and Hash

- Very often digital signatures are used with hash functions, hash of a message is signed, instead of the message.
- Hash function must be:
 - Pre-image resistant
 - Weak collision resistant
 - Strong collision resistant



RSA Signature

Key generation (as in RSA encryption):

- Select 2 large prime numbers of about the same size, p and q
- Compute $n = pq$, and $\Phi = (q - 1)(p - 1)$
- Select a random integer e , $1 < e < \Phi$, s.t. $\gcd(e, \Phi) = 1$
- Compute d , $1 < d < \Phi$ s.t. $ed \equiv 1 \pmod{\Phi}$

Public key: (e, n)

Secret key: d, p and q must also remain secret

RSA Signature (cont.)

Signing message M

- M must verify $0 < M < n$
- Use private key (d)
- compute $S = M^d \bmod n$

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

Verifying signature S

- Use public key (e, n)
- Compute $S^e \bmod n = (M^d \bmod n)^e \bmod n = M$

Note: in practice, a hash of the message is signed and not the message itself.

RSA Signature (cont.)

Example of forging

- Attack based on the multiplicative property of RSA.

$$y_1 = \text{sig}_K(x_1)$$

$$y_2 = \text{sig}_K(x_2), \text{ then}$$

$$\text{ver}_K(x_1x_2 \bmod n, y_1y_2 \bmod n) = \text{true}$$

- So adversary can create the valid signature $y_1y_2 \bmod n$ on the message $x_1x_2 \bmod n$
- This is an existential forgery using a known message attack.

El Gamal Signature

Key Generation (as in ElGamal encryption)

- Generate a large random prime p such that DLP is infeasible in Z_p and a generator g of the multiplicative group Z_p of the integers modulo p
- Select a random integer a , $1 \leq a \leq p-2$, and compute
$$y = g^a \text{ mod } p$$
- Public key is $(p; g ; y = g^a)$
- Private key is a .
- Recommended sizes: 1024 bits for p and 160 bits for a .

ElGamal Signature (cont.)

Signing message M

- Select random k , $1 \leq k \leq p-1$, $k \in \mathbb{Z}_{p-1}^*$
- Compute
$$r = g^k \bmod p$$
$$s = k^{-1}(h(M) - ar) \bmod (p-1)$$
- Signature is: (r,s)
- Size of signature is double size of p

NOTE: h is a hash function



ElGamal Signature (cont.)

Signature is: (r, s)

$$r = g^k \text{ mod } p$$

$$s = k^{-1}(h(M) - ar) \text{ mod } (p-1)$$

Verification

- Verify that r is in Z_{p-1}^* : $1 \leq r \leq p-1$
- Compute
$$v_1 = y^r r^s \text{ mod } p$$
$$v_2 = \alpha^{h(M)} \text{ mod } p$$
- Accept iff $v_1 = v_2$

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

ElGamal Signature (cont.)

Security of ElGamal signature

- Weaker than DLP
- k must be unique for each message signed
- Hash function h must be used, otherwise easy for an existential forgery attack
- $0 < r < p$ must be checked, otherwise easy to forge a signature on any message if an valid signature is available.

Schnorr Signature

Key generation (as in DSA, $h:\{0,1\}^* \rightarrow \mathbb{Z}_q$)

- Select a prime q
- Select $1 \leq a \leq q-1$
- Compute $y = g^a \text{ mod } p$

Public key: (p, q, g, y)

Private key: a

Schnorr Signature

Signing message M

- Select random secret k , $1 \leq k \leq q-1$
- Compute
$$r = g^k \bmod p,$$
$$e = h(M \parallel r)$$
$$s = ae + k \bmod q$$
- Signature is: (s, e)

Schnorr Signature

Verification

- Compute

$$v = g^{sy^{-e}} \bmod p,$$

$$e' = h(m \parallel v)$$

- Valid iff $e' = e$

Digital Signature Algorithm (DSA)

Specified as FIPS 186

Key generation

- Select a prime q of 160-bits
- Choose $0 \leq t \leq 8$
- Select $2^{511+64t} < p < 2^{512+64t}$ with $q \mid p-1$
- Select g in Z_p^* , and $\alpha = g^{(p-1)/q} \bmod p$, $\alpha \neq 1$
- Select $1 \leq a \leq q-1$
- Compute $y = \alpha^a \bmod p$

Public key: (p, q, α, y)

Private key: a

DSA

Signing message M:

- Select a random integer k , $0 < k < q$
- Compute
$$k^{-1} \bmod q$$
$$\mathbf{r = (\alpha^k \bmod p) \bmod q}$$
$$\mathbf{s = k^{-1} (h(M) + ar) \bmod q}$$
- Signature: (r, s)

Note: FIPS recommends
the use of SHA-1 as hash function.



DSA

Signature: (r, s)

$$r = (\alpha^k \bmod p) \bmod q$$

$$s = k^{-1} (h(M) + ar) \bmod q$$

Verification

- Verify $0 < r < q$ and $0 < s < q$, if not, invalid
- Compute
$$w = s^{-1} \bmod q$$
$$u_1 = wh(m) \bmod q,$$
$$u_2 = rw \bmod q$$
$$v = (\alpha^{u_1} y^{u_2} \bmod p) \bmod q$$
- Valid iff $v = r$

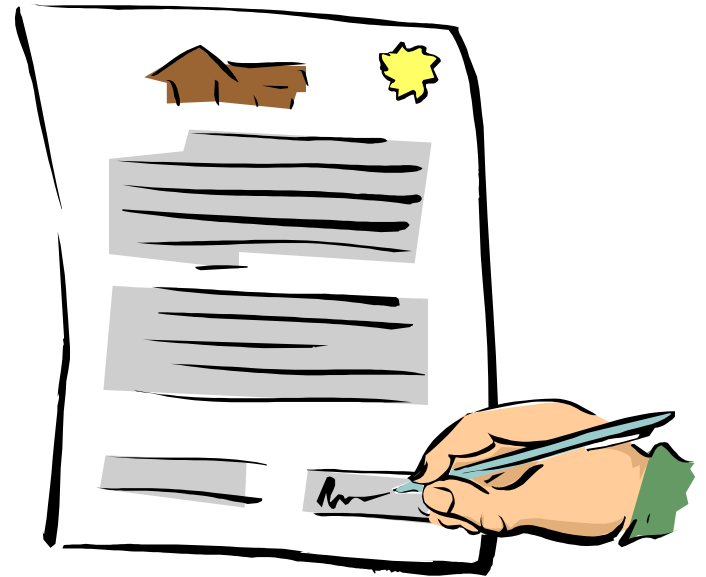
One-Time Digital Signatures

- One-time digital signatures: digital schemes used to sign, at most one message; otherwise signature can be forged.
- A new public key is required for each signed message.
- Advantage: signature generation and verification are very efficient and is useful for chipcards, where low computation complexity is required.

Lamport One-time Signature

To sign one bit:

- Choose as secret keys x_1, x_2
 - x_1 represents '0'
 - x_2 represents '1'
- public key:
 - $y_1 = h(x_1)$,
 - $y_2 = h(x_2)$.
- Signature is $h(x_1)$ if the message is x_1 or $h(x_2)$ for x_2



Rabin One-Time Signature

Key generation

- Select a symmetric key encryption scheme E
- Generate $2n$ random secret strings k_1, k_2, \dots, k_{2n} , each of bit length l
- Compute

$$y_i = E_{k_i}(M_0(i)), \quad i \in [1, 2n], \text{ where}$$

$$M_0(i) \text{ is } 0^{l-e} \parallel b_{e-1} \dots b_1 b_0,$$

$b_{e-1} \dots b_1 b_0$ is the binary representation of i

Public key: $(y_1, y_2, \dots, y_{2n})$,

Private key: $(k_1, k_2, \dots, k_{2n})$.

$n = 80, l = 64$

Rabin One-Time Signature

Signing message m

- Compute $h(m)$
- Compute
$$s_i = E_{k_i}(h(m)), i \in [1, 2n]$$
- Signature is: $(s_1, s_2, \dots, s_{2n})$

Rabin One-Time Signature

Verifying

- Compute $h(m)$
- Select n distinct random number r_j , $r_j \in [1, 2n]$, with $j \in [1, n]$
- Request from signer, the keys k_{r_j} , $j \in [1, n]$
- Verify received n keys:
 - $z_j = E_{k_{r_j}}(M_0(r_j))$ and
 - $z_j = y_{r_j}$
- Verify all
 - $s_{r_j} = E_{k_{r_j}}(h(m))$, $j \in [1, n]$

Merkle One-Time Signature

Key generation: to sign a n -bit message

- Select $t = n + 1 + \lfloor \lg n \rfloor$ and random secret strings k_1, k_2, \dots, k_t each of bitlength l

- Compute

$$v_i = h(k_i), \quad 1 \leq i \leq t,$$

h is a pre-image resistant hash:

$$\{0,1\}^* \rightarrow \{0,1\}^l$$

- Public key: (v_1, v_2, \dots, v_t)
- Private key: (k_1, k_2, \dots, k_t)

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

Merkle One-Time Signature

Sign message m of bit-length n

- Compute c , the binary representation of number of 0's in m ;
 $w = m \parallel c = (a_1 a_2 \dots a_t)$
- Determine the coordinate positions $i_1 < i_2 \dots < i_u$ in w , s.t. $a_{i_j} = 1$, $1 \leq j \leq u$
- Let $s_j = k_{i_j}$, $1 \leq j \leq u$
- Signature is: (s_1, \dots, s_u)

Merkle One-Time Signature

Verification

- Compute c , the binary representation of number of 0's in m ;
 $w = m \parallel c = (a_1 a_2 \dots a_t)$
- Determine the coordinate positions $i_1 < i_2 \dots < i_u$ in w , s.t. $a_{i_j} = 1$
- Let $s_j = k_{i_j}$, $1 \leq j \leq u$
- Valid iff $v_{i_j} = h(s_j)$ for all j , $1 \leq j \leq u$

Summary

- Digital signatures consist of a private algorithm and a public verifying algorithm
- Main difference between digital signatures and HMAC is non-repudiation



Next Lecture...

- Authentication & Key Exchange Protocols

