

Cryptography CS 555

Lecture 21



Department of Computer Sciences
Purdue University

Review of Signature Schemes

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



Digital Signatures

- Digital Signature: a data string which associates a message with some originating entity.
- Digital Signature Scheme:
 - a secret signing key and a public verification key
- Services provided:
 - Authentication
 - Data integrity
 - Non-Repudiation (MAC does not provide this.)

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

Attack Models for Digital Signatures

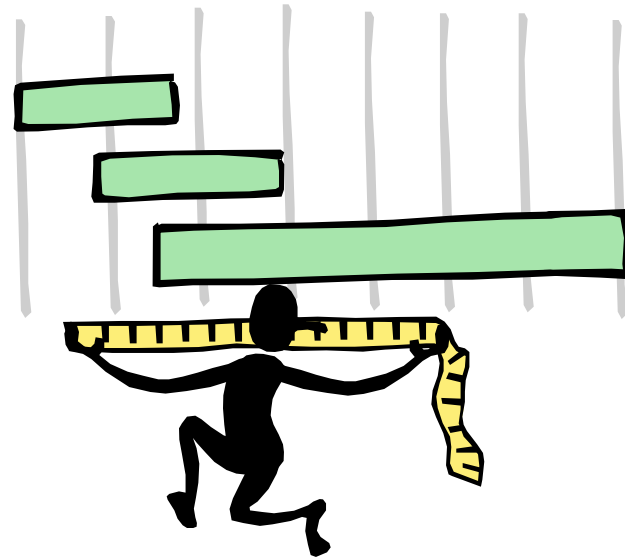
- **Key-only attack:** Adversary knows only the verification function (which is supposed to be public).
- **Known message attack:** Adversary knows a list of messages previously signed by Alice.
- **Chosen message attack:** Adversary can choose what messages wants Alice to sign, and he knows both the 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 cannot 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.

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

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

$$\text{ver}_K(x_1 x_2 \bmod n, y_1 y_2 \bmod n) = \text{true}$$

- So adversary can create the valid signature $y_1 y_2 \bmod n$ on the message $x_1 x_2 \bmod n$
- 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 DLog
- 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

$$y = g^a \text{ mod } p$$

Schnorr Signature

Signing message m

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

Schnorr Signature

Verifying that (s, e) is a signature on m

- Computes

$$r' = g^{sy^{-e}} \text{ mod } p$$

and verifies that

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

Digital Signature Algorithm (DSA)

- Variant of El Gamal
- Use a subgroup of Z_p^*

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.

One-Time Digital Signatures

- Lamport one-time signature scheme
- Merkle one-time signature scheme

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 , $1 \leq j \leq u$

Public Key Certificates

Distribution of Public Keys

- **Public announcement:** users distribute public keys to recipients or broadcast to community at large
- **Publicly available directory:** can obtain greater security by registering keys with a public directory
- Both approaches have problems, and are vulnerable to forgeries



Public-Key Certificates

- Certificates allow key exchange without real-time access to public-key authority
- A certificate binds identity (or other information) to public key
- Contents signed by a trusted Public-Key or Certificate Authority (CA)
- Can be verified by anyone who knows the public-key authorities public-key

X.509 Authentication Service

- Part of X.500 directory service standards.
- Defines framework for authentication services:
 - Defines that public keys stored as **certificates** in a public directory.
 - Certificates are **issued and signed** by an entity called **certification authority (CA)**.
- Used by numerous applications: SSL, IPSec, SET
- Started 1988

X.509 Certificates

- Certificates contain:
 - version (1, 2, or 3)
 - serial number (unique within CA) identifying certificate
 - signature algorithm identifier
 - issuer X.500 name (CA)
 - period of validity (from - to dates)
 - subject X.500 name (name of owner)
 - subject public-key info (algorithm, parameters, key)
 - issuer unique identifier (v2+)
 - subject unique identifier (v2+)
 - extension fields (v3)
 - signature (of hash of all fields in certificate)

Other Topics in Certificates

- PGP
 - different trust model
- Certificate revocation
 - CRL
 - OCS
 - CRT

Authentication

- **Entity authentication (identification)**: the process whereby one party is assured of the identity of a second party involved in a protocol and that the second has actually participated.
- **Data source authentication**: is represents an indication about the source of the data.



Basic Authentication Schemes

- Fixed passwords
- Using salts
- One-time passwords
 - shared lists of one-time passwords
 - sequentially updated one-time passwords
 - one-time password sequences based on a one-way function (Lamport)

Challenge-Response Protocols

- Goal: one entity authenticates to other entity proving the knowledge of a secret, 'challenge'
- Time-variant parameters used to prevent replay, interleaving attacks, provide uniqueness and timeliness : nonce (used only once)

Three Types of Challenges

- **Random numbers:**
 - pseudo-random numbers that are unpredictable to an adversary;
 - vulnerable to birthday attacks, use larger sample;
 - must maintain state;
 - do not prevent interleaving attacks (parallel sessions)
- **Sequences:**
 - serial number or counters;
 - long-term state information must be maintained by both parties+ synchronization
- **Timestamp:**
 - provides timeliness and detects forced delays;
 - requires synchronized clocks.

Different kinds of responses

- Using symmetric encryption
- Using digital signatures

Zero-Knowledge Protocols

- **Zero-knowledge protocols:** allows a prover to prove that it possesses a secret without revealing any information of use to the verifier.
- **Motivation:**
 - more secure (multiple uses don't degrade the security)
 - the verifier cannot prove to 3rd party
 - sometimes more efficient than signatures

Two Kinds of Zero-Knowledge Proofs

- ZK proof of a statement
 - convincing the verifier that a statement is true without yielding any other information
 - example of a statement, a propositional formula is satisfiable
- ZK proof of knowledge of a secret
 - convincing the verifier that one knows a secret, e.g., one knows the square root modulo $N=pq$

Properties Zero-Knowledge Proofs

- Properties of ZK Proofs:
 - completeness
 - honest prover who knows the secret convinces the verifier with overwhelming probability
 - soundness
 - no one who doesn't know the secret can convince the verifier with nonnegligible probability
 - zero knowledge
 - the proof does not leak any additional information

Typical ZK Protocols

- Multiple rounds
- Each round consists of 3 steps
 - commit
 - challenge
 - respond
- If challenge can be predicted, then cheating is possible.
 - cannot convince a third party (even if the party is online)
- If respond to more than one challenge with one commit, then the secret is revealed.

Formalizing ZK property

- A simulator exists
 - taking what the verifier knows before the proof, can generate a communication transcript that is indistinguishable from one generated during ZK proofs
- Three kinds of indistinguishability
 - perfect (information theoretic)
 - statistical
 - computational

Different kinds of ZK

- Perfect ZK
- Statistical ZK
- Computational ZK
- An orthogonal angle
 - Honest verifier ZK

Fiat-Shamir Identification Protocol

Setup:

- 1) A trusted server T makes public $n = pq$, keeping secrets p and q (selected as in RSA)
- 2) Every claimant A selects $0 < s < n$, $\gcd(s, n) = 1$, computes $v = s^2 \bmod n$ and registers v with the server T

Protocol run t rounds, proof accepted if all t rounds succeed

1. A selects r , $0 < r < n$, and sends to B $x = r^2 \bmod n$
2. B randomly selects $e = 0$ or $e = 1$, and sends e to A
3. If $e = 0$, A sends $y = r$, else sends $y = rs \bmod n$
4. B rejects $y = 0$, otherwise accept if $y^2 \equiv x v^e \bmod n$

Schnorr Identification Protocol

Like DSA select public (p, q, g) , and t ,
 t defines the security of the system

Setup: a claimant A has an identifier I_A , private a , $0 < a < q$,
 $v = g^{-a} \bmod p$, obtain certificate that provides
authenticity of v , cert_A

Protocol:

A selects $0 < r < q$, computes $x = g^r \bmod p$,

B returns challenge $1 \leq e \leq 2^t \leq q$

A sends $y = (ae + r) \bmod q$

B accepts if $x = g^y v^e \bmod p$

Converting Interactive ZK to Non-interactive ZK

- The only interactive role played by the verifier is to generate random challenges
 - challenges not predictable by the prover
- The same thing can be done using one-way hash functions

Interactive ZK Implies Signatures

- Given a message M , replace the random challenge of the verifier by the one-way hash $c=h(x||M)$

Key Transport vs. Key Agreement

- **Key establishment**: process to establish a shared secret key available to two or more parties;
 - **key transport**: one party creates, and securely transfers it to the other(s).
 - **key agreement**: key establishment technique in which a shared secret is derived by two (or more) parties

Key Pre-distribution vs. Dynamic Key Establishment

- **Key establishment**
 - **Key pre-distribution**: established keys are completely determined a priori by initial keying material
 - generally in the form of key agreement
 - **Dynamic shared key establishment**: protocols that keys established between a fixed group of users varies in different sessions
 - also known as session key establishment
 - could be key transport or key agreement

Notions of Authentication

- **Entity authentication:** identity of a party, and aliveness at a given instant
- **Implicit key authentication:** one party is assured that no other party aside from a specifically identified second party may gain access to a particular secret key.
- **Key confirmation:** one party is assured that a second party actually has possession of a particular secret key.
- **Explicit key authentication:** both (implicit) key authentication and key confirmation hold.

Assumptions and Adversaries

- Assumption: Protocol messages are transmitted over open networks
- An adversary may
 - deduce a session key using eavesdropping.
 - altering messages to be able to deduce the key
 - deceive a legitimate party regarding the identity of the party with which is shares a key
 - initiate one or more protocol execution (possibly simultaneously) and combine messages from one with another)

Effects of Key Compromise

- **Perfect forward secrecy**: compromise of long-term key does not compromise past session keys.
- **Known-key attack**: compromise of past session keys allows either a passive adversary to compromise future session keys, or impersonation by an active adversary in the future.

Protocols

- AKEP2
- Shamir's no-key algorithm
- Needham-Schroeder shared key
- Needham-Schroeder public key
- Diffie-Hellman
- Authenticated Diffie-Hellman
- MTI
- STS

Other Topics covered

- Mental poker protocol
- Different security requirements for public-key encryption
- Secure multi-party function evaluation
 - oblivious transfer
 - Yao's generic protocol