CS 408
Cryptography & Internet Security

Lecture 19:

Digital signatures,

RSA signature,

PKI,

Hybrid schemes

Last time

- Cryptographic Hash Functions
- Message Authentication Codes

What Do We Have in Our Toolbox?

- Confidentiality
  - Symmetric-key encryption (block ciphers: AES, DES)
  - Public-key encryption (RSA, ElGamal)
- Integrity
  - Message Authentication Codes (HMAC-SHA1)
- Authentication
  - Message Authentication Codes (HMAC-SHA1)
- Non-Repudiation?

Digital Signatures: 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 Signature

Each entity has:

- a public key (Pub), which is made public
- a private key (Priv), which is kept secret

(Pub_A, $Priv_A$)

Alice

M, Sig →

Bob

generate signature
$Sig = S_{PrivA}(M)$

verify signature

$V_{PubA}(M, Sig) \stackrel{?}{=}$ "valid"

Digital Signature (continued)

- Entities don't need to establish a secret key or a trust relationship ahead of time
- A public key signature scheme is a collection of three algorithms (G, S, V)
  - Key generation algorithm G: generates a pair of keys (Pub, Priv)
  - Signature generation algorithm S: $Sig = S_{Priv}(M)$
  - Signature verification algorithm V:
    "result" = $V_{Pub}(M,Sig)$, where "result" is "valid" or "invalid"
- The following should always hold true:
  - $V_{Pub}(M, Sig))$ = "valid", if $Sig = S_{Priv}(M)$
    = "invalid", otherwise

Digital Signature (continued)

- It is infeasible to derive the private key from the public key
- Provides: authentication, integrity, non-repudiation
- Does not provide: confidentiality
- It is a keyed cryptographic primitive
- Example: RSA signature, ElGamal signature, DSA signature

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.
- **Existential forgery**: adversary can create a pair (message, signature), s.t. the signature of the message is valid.

Attack Models for Digital Signatures

- **Key-only attack**: Adversary knows only the public verification key.
- **Known message attack**: Adversary knows a list of messages previously signed by Alice (and their corresponding signatures).
- **Chosen message attack**: Adversary can choose what messages Alice will sign, and he knows both the messages and the corresponding signatures.

Two Flavors of Digital Signatures

- Digital signatures with appendix
  - A computes Sig = $S_{PrivA}(M)$
    A sends to B: (M, Sig)
    B verifies if Sig is a valid signature on M
  - Example: Schnorr signature scheme
- Sigital signatures with message recovery
  - A computes Sig = $S_{PrivA}(M)$
    A sends to B: Sig
    B uses Sig to recover M and also verifies the validity of the signature in the process
  - Example: RSA signature scheme

Digital Signatures and Hash Functions

- For efficiency reasons, digital signatures are usually used in combination with cryptographic hash functions:
  - To sign a message, first compute a hash of the message, and then sign the hash (instead of the message)
    1. A computes Sig = $S_{PrivA}(h(M))$
    2. A sends to B: M, Sig
    3. B computes h(M) and checks if $V_{PubA}(h(M), Sig)$ = "valid"

- Cryptographic hash functions must have:
  - Pre-image resistance (first pre-image resistance)
  - Weak collision resistance (second pre-image resistance)
  - Strong collision resistance (collision resistance)

RSA Digital Signature

Key generation (as in RSA encryption):

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

Public key: (n, e)
Private key: d

Note: p and q must remain secret

RSA Digital Signature

Signature Generation:

Goal: generate a digital signature for message M

- Represent the message as an integer M , $0 < M < n$
- Compute $S = M^d \bmod n$
- Send S to recipient

Signature Verification:

- Obtain the sender's public key (n,e)
- Compute $M = S^e \bmod n$

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

Security of RSA Signature

Example of forging

- Attack based on the multiplicative property of RSA:

  $y_1 = sig_K(x_1) = x_1^d \bmod n$

  $y_2 = sig_K(x_2) = x_2^d \bmod n$, then

  $ver_K(x_1x_2 \bmod n, y_1y_2 \bmod n) = true$

  $sig(x_1x_2) = (x_1x_2)^d = (x_1)^d (x_2)^d = y_1y_2 \bmod n$

- 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.

Security of RSA Signature

- To avoid the forgery attack, we must use a secure padding scheme that encodes the message in order to provide some specific structure to the message:

  A computes Sig = $(P(M))^d \bmod n$

  B recovers P(M) and checks if it has a specific format

  Current standards for secure padding schemes: PKCS#1 (provides padding for signatures, similar with the OAEP padding for public-key encryption)

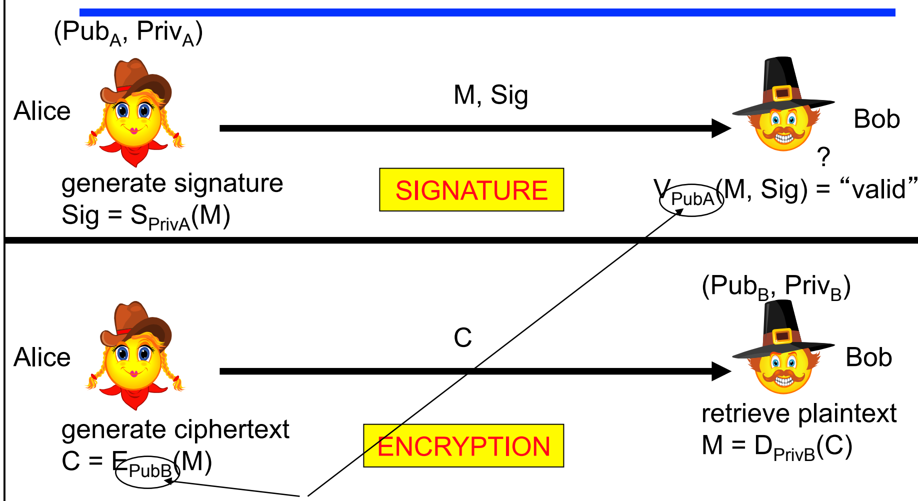
Our Toolbox (revisited)

- Confidentiality
  - Symmetric-key encryption (block ciphers: AES, DES)
  - Public-key encryption (RSA, ElGamal)
- Integrity
  - Digital Signatures (RSA signature)
  - Message Authentication Codes (HMAC-SHA1)
- Authentication
  - Message Authentication Codes (HMAC-SHA1)
  - Digital Signatures (RSA signature)
- Non-Repudiation
  - Digital Signatures (RSA signature)

Our Toolbox (another view)

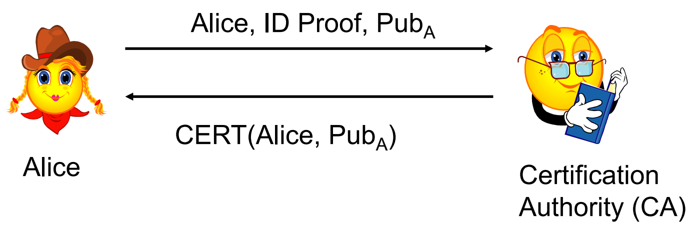
- Symmetric-key encryption
  - Confidentiality
- Public-key encryption
  - Confidentiality
- Message Authentication Codes
  - Integrity and Authentication
- Digital Signatures
  - Integrity, Authentication, Non-Repudiation

Public-key Cryptographic Primitives



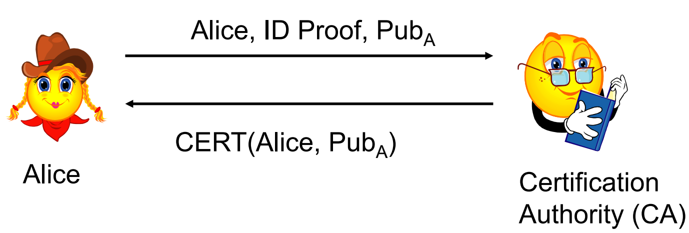
How to ensure the authenticity of the other's party public key?

Public Key Infrastructure (PKI)



- CERT(Alice, Pub_A) is Alice's public key certificate, which binds Alice's identity to her public key
  - signed by the CA (using the CA's private key)
- Anyone can verify authenticity of $CERT_A$ by using the CA's public key
- The CA's public key is readily available in a ***root certificate***
  - Included in the browser, published online, or in a newspaper, or on a CD etc.
  - The root certificate is a ***self-signed certificate*** (signed with the private key corresponding to the actual public key contained in the certificate)

Public Key Infrastructure (PKI)



- To verify a signature from Alice:
  - Bob retrieves Alice's certificate $CERT_A$ = CERT(Alice, Pub_A)
  - Bob can verify $CERT_A$ by using the CA's public key
  - Bob can verify the signed message using Pub_A

PKI

- A public key certificate contains several fields:
  - The identity of the public key's owner
  - The public key
  - Serial number
  - Expiration date
  - Other useful fields

Public Key Infrastructure

- When Alice needs Bob's public key, she retrieves Bob's certificate: CERT(Bob, Pub_B)
  - Alice has the authentic public key of the CA, so she can verify the authenticity of Bob's certficate
  - This validates the authenticity of Bob's public key, Pub_B, which is contained inside Bob's certificate
- A Root Certificate acts as an ***anchor point*** in the *chain of trust*
  - They are used to validate certificates lower in the PKI hierarchy
- PKI = the entire infrastructure needed to support public key cryptography
  - Includes organizations (CAs), principals, and their interactions

Problems with PKI?

- Public key certificates have expiration dates
  - Short term (a few months), or longer term (a few years)
- What happens when a certificate needs to be revoked before the expiration date?
  - a company goes out of business
  - a web domain changes ownership
  - an employee changes affiliation (e.g., leaves a company)
  - the private key gets compromised (!)
  - the private key is lost (!)

Problems with PKI?

- Certificate Revocation Lists (CRL) are lists of certificates (serial numbers more precisely) that have been revoked
  - Published periodically by the CA that issued the corresponding certificate
- Best practices require to always check if a certificate has not been revoked before expiration
  - This means, that in order to PKI effectively, one needs to always check the list of current CRLs
  - Thus, there is a need for an entity which is always available ("on-line") and provides CRLs
  - This requirement of "on-line validation" negates one of the original major advantages of PKI over symmetric key cryptography
  - The need for verifying CRL before certification validation also raises the possibility of DoS attacks against the PKI

Public key cryptography

- Advantages over symmetric key crypto
  - Key management
    - Key establishment: does not require secure channel to transmit secret keys
    - Key distribution: does not require $O(n^2)$ keys to be managed to communicate with n entities
    - Can provide non-repudiation
- Disadvantages over symmetric key crypto
  - Slower (orders of magnitude)
- Is not meant to completely replace symmetric key cryptography, but to supplement it
  - See hybrid schemes

Hybrid Schemes

- Alice wants to send a large secret message to Bob over an insecure channel
- How should Alice encrypt the message for Bob?

First attempt

$P_{Bob}(m)$

(where P_{Bob} is public-key encryption with Bob's public key)

- This is very inefficient for a large message m!

Second attempt

$E_k(m)$, $P_{Bob}(k)$

(where E_k is symmetric-key encryption with key k,

P_{Bob} is public-key encryption with Bob's public key)

- But $E_k(m)$ doesn't guarantee integrity of m!

Hybrid Schemes

1. $E_{k1}(m)$, $HMAC_{k2}(m)$, $P_{Bob}(k1 \| k2)$

2. $E_{k1}(m)$, $HMAC_{k2}(E_{k1}(m))$, $P_{Bob}(k1 \| k2)$

3. $E_{k1}(m \| HMAC_{k2}(m))$, $P_{Bob}(k1 \| k2)$

4. $E_k(m \| S_{Alice}(m))$, $P_{Bob}(k)$
   (where S_{Alice} is a signature with Alice's private key)

Authenticated Encryption

- A symmetric-key block cipher (such as AES) can also be used in a mode of operation that provides both confidentiality and authentication/integrity:
  authenticated encryption

- Examples of such modes of operation:
  OCB, GCM, CCM, CWC, EAX

Computational Costs of Various Cryptographic Primitives

(results obtained with OpenSSL on MacBook 2 Ghz Intel Core 2 Duo with 2 GB memory)

1024-bit blocks:

sha1 :	4.31×10^{-6} seconds
hmac(md5):	9.19×10^{-6} s
aes-128-cbc:	26×10^{-6} s

1024-bit keys:

RSA sign:	7.22×10^{-3} s
RSA verify:	0.31×10^{-3} s
DSA sign:	3.09×10^{-3} s
DSA verify:	3.72×10^{-3} s