CS549: Cryptography and Network Security

© by Xiang-Yang Li

Department of Computer Science,
IIT
This lecture note (Cryptography and Network Security) is prepared by Xiang-Yang Li. This lecture note has benefited from numerous textbooks and online materials. Especially the “Cryptography and Network Security” 2nd edition by William Stallings and the “Cryptography: Theory and Practice” by Douglas Stinson.

You may not modify, publish, or sell, reproduce, create derivative works from, distribute, perform, display, or in any way exploit any of the content, in whole or in part, except as otherwise expressly permitted by the author.

The author has used his best efforts in preparing this lecture note. The author makes no warranty of any kind, expressed or implied, with regard to the programs, protocols contained in this lecture note. The author shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these.
Cryptography and Network Security

Digital Signature

Xiang-Yang Li
Digital Signature

Alice

Document

To be or not to be?

0101110101011110010

11010101110101101

Document

To be or not to be?

0101110101011110010

11010101110101101

Sign

\[ S = S(A, M) \]

Alice's signature
Digital Signatures

- have looked at message authentication
  - but does not address issues of lack of trust

- digital signatures provide the ability to:
  - verify author, date & time of signature
  - authenticate message contents
  - be verified by third parties to resolve disputes

- hence include authentication function with additional capabilities
Digital Signature Properties

- must depend on the message signed
- must use information unique to sender
  - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- be practical save digital signature in storage
Securities

- A **total break** results in the recovery of the signing key.
- A **universal forgery** attack results in the ability to forge signatures for any message.
- A **selective forgery** attack results in a signature on a message of the adversary's choice.
- An **existential forgery** merely results in some valid message/signature pair not already known to the adversary.
Classification of Digital Signature

- Undeniable --- need signer to interact
- Fail-Stop
- Blind
- One-time
- Multi-party (group signature)
- (n,k)-multi-party
- Oblivious
- Multi-undeniable
Algorithm and legal concerns

- several prior requirements
  - quality algorithms. Some public key algorithms are known to be insecure, practicable attacks against them having been identified.
  - quality implementations. An implementation of a good algorithm with mistake(s) will not work. (about 1 defect per 1,000 lines).
  - the private key must remain actually secret; if it becomes known to some other party, that party can produce perfect digital signatures of anything whatsoever.
  - distribution of public keys must be done in such a way that the public key claimed to belong to Bob actually belongs to Bob, and vice versa. This is commonly done using a public key infrastructure and the public key user association is attested by the operator of the PKI (called a certificate authority). For 'open' PKIs in which anyone can request such an attestation, the possibility of mistake is non trivial.
  - users (and their software) must carry out the signature protocol properly.
  - Legal concerns
Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender’s public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using recipient's public-key
- important that sign first then encrypt message & signature
- security depends on sender’s private-key
Arbitrated Digital Signatures

- involves use of arbiter A
  - validates any signed message
  - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message
Creating and verifying a digital signature

1. Calculate hashcode

2. Encrypt the hashcode with private key of the sender

3. Digitally signed document

4. Decrypt the signature with the public key of the sender

5. If the calculated hashcode does not match the result of the decrypted signature, either the document was changed after signing, or the signature was not generated with the private key of the alleged sender.
RSA signature

- $N = p \cdot q$, where $p$ and $q$ are large primes
- Alice’s private key $(e, n)$,
- Alice’s public key $(d, n)$

- Signature of message $m$ by Alice
  - $S = H(m)^e \mod n$

- Verification of signature by Bob
  - Check if $h(m) = S^d \mod n$
Cont.

- Typically $d$ is chosen small (3 or $2^{16}+1$)
- Problem:
  - Easy to create the signature of $h(m_1)h(m_2)$

- RSA-PSS
  - Use some more randomization to enhance security
  - It was added in version 2.1 of PKCS #1 (see RFC 3447).
ElGamal Signature

- **Global public components**
  - Prime number $p$ with 512-1024 bits
  - Primitive element $g$ in $\mathbb{Z}_p$

- **Users private key**
  - Random integer $x$ less than $p$

- **Users public key**
  - Integer $y = g^x \mod p$
Elgamal

- **Signature**
  - For each message M, generates random k
  - Computes $r = g^k \mod p$
  - Computes $s = k^{-1}(H(M) - xr) \mod (p-1)$
  - Signature is $(r, s)$

- **Verifying**
  - Computes $v_1 = g^{H(M)} \mod p$
  - Computes $v_2 = y^r r^s \mod p$
  - Test if $v_1 = v_2$
Proof of Correctness

- Computes $v_2 = y^r r^s \mod q$

  - So $v_2 = y^r r^s \mod q = g^{xr} g^{ks} \mod p$
  - $= g^{xr+k(k^{-1}(H(M)-x)r) \mod (p-1)} \mod p$
  - $= g^{H(M)} \mod p = v_1$

  - Notice that here it uses Fermat theorem to show
    - That $g^{(H(M)-x)r \mod (p-1)} \mod p = g^{(H(M)-x)r} \mod p$
The main disadvantage of ElGamal is

- the need for randomness (sometimes it is good), and
- its slow speed (especially for signing).

Another potential disadvantage of the ElGamal system is that message expansion by a factor of two takes place during encryption. However, such message expansion is negligible if the cryptosystem is used only for exchange of secret keys.
Digital Signature Standard

- FIPS PUB 186 by NIST, 1991
- Final announcement 1994
- It uses
  - Secure Hashing Algorithm (SHA) for hashing
  - Digital Signature Algorithm (DSA) for signature
  - The hash code is set as input of DSA
  - The signature consists of two numbers
- DSA
  - Based on the difficulty of discrete logarithm
  - Based on Elgamal and Schnorr system
DSA

- **Global public components**
  - Prime number $p$ with 512-1024 bits
  - Prime divisor $q$ of $(p-1)$ with 160 bits
  - Integer $g = h^{(p-1)/q} \mod p$

- **Users private key**
  - Random integer $x$ less than $q$

- **Users public key**
  - Integer $y = g^x \mod p$
**DSA**

- **Signature**
  - For each message $M$, generates random $k$
  - Computes $r = (g^k \mod p) \mod q$
  - Computes $s = k^{-1}(H(M) + xr) \mod q$
  - Signature is $(r, s)$

- **Verifying**
  - Computes $w = s^{-1} \mod q$, $u_1 = H(M)w \mod q$
  - Computes $u_2 = rw \mod q, v = (g^{u_1}y^{u_2} \mod p) \mod q$
  - Test if $v = r$
Proof of Correctness

- Notice that $v=(g^{u_1}y^{u_2} \mod p) \mod q$
  - $=(g^H(M)w \mod q \ y^{rw} \mod q \mod p) \mod q$
  - $=(g^H(M)w \mod q \ g^{xrw} \mod q \mod p) \mod q$
  - $=(g^{H(M)w+xrw} \mod q \mod p) \mod q$
  - $=(g^{(H(M)+xr)w} \mod q \mod p) \mod q$
  - $=(g^{(H(M)+xr)k(H(M)+xr)^{-1}} \mod q \mod p) \mod q$
  - $=(g^{k} \mod p) \mod q$
  - $=r$
In practice (Sun Java Library)

- $g = F7E1A085D69B3DDE\ CBBCAB5C36B857B9\ 7994AFBBFA3AEA82\ F9574C0B3D078267\ 5159578EBAD4594F\ E67107108180B449\ 167123E84C281613\ B7CF09328CC8A6E1\ 3C167A8B547C8D28\ E0A3AE1E2BB3A675\ 916EA37F0BFA2135\ 62F1FB627A01243B\ CCA4F1BEA8519089\ A883DFE15AE59F06\ 928B665E807B5525\ 64014C3BFECF492A$

- $p = FD7F53811D751229\ 52DF4A9C2EECE4E7\ F611B7523CEF4400\ C31E3F80B6512669\ 455D402251FB593D\ 8D58FABFC5F5BA30\ F6CB9B556CD7813B\ 801D346FF26660B7\ 6B9950A5A49F9FE8\ 047B1022C24FBBBA9\ D7FEB7C61BF83B57\ E7C6A8A6150F04FB\ 83F6D3C51EC30235\ 54135A169132F675\ F3AE2B61D72AEFF2\ 2203199DD14801C7$

- $q = 9760508F15230BCC\ B292B982A2EB840B\ F0581CF5$

Here $g$ and $p$ have 1024 bits, while $q$ has 160 bits. They fulfill the requirement that $g^q = 1 \mod p$. 

Cryptography and Network Security
Note

- Can we use the random number k twice?
  - What will happen if k used twice?
  - We have $r = (g^k \mod p) \mod q$
  - $s_1 = k^{-1}(H(M_1)+xr) \mod q$ and $s_2 = k^{-1}(H(M_2)+xr) \mod q$
  - We have $s_1 - s_2 = k^{-1}(H(M_1)-H(M_2)) \mod q$

- Another attack (for OpenPGP)
  - Replace p and g
  - http://www.tigertools.net/board/?topic=topic4&msg=14
  - http://www.orlingrabbe.com/DSAflaw_OpenPGP.htm
Cont.

- We cannot use small k
Non-deterministic

- Non-determined signatures
  - For each message, many valid signatures exist
  - DSA, Elgamal

- Deterministic signatures
  - For each message, one valid signature exists
  - RSA
Comparisons

- **Speed**
  - DSS has faster signing than verifying
  - RSA could have faster verifying than signing
  - Message be signed once, but verified many times
    - This prefers the faster verification
  - But the signer may have limited computing power
    - Example: smart card
    - This prefers the faster signing
Blind Signature (digital cash)

- first introduced by Chaum, allow a person to get a message signed by another party without revealing any information about the message to the other party.
- Suppose Alice has a message $m$ that she wishes to have signed by Bob, and she does not want Bob to learn anything about $m$.
  - Let $(n,e)$ be Bob's public key and $(n,d)$ be his private key.
  - Alice generates a random value $r$ such that $\gcd(r, n) = 1$ and sends $x = (r^e m) \mod n$ to Bob. The value $x$ is "blinded" by the random value $r$; hence Bob can derive no useful information from it.
  - Bob returns the signed value $t = x^d \mod n$ to Alice.
  - Since $x^d \equiv (r^e m)^d \equiv r^m \mod n$,
  - Alice can obtain the true signature $s$ of $m$ by computing $s = r^{-1} t \mod n$. 
Security Concerns

- GnuPG permits creating ElGamal keys
  - are usable for both encryption and signing.
  - It is even possible to have one key (the primary one) used for both operations.
  - This is not considered good cryptographic practice, but is permitted by the OpenPGP standard.

- signature is much larger than a RSA or DSA signature
  - verification and creation takes far longer and the use of ElGamal for signing has always been problematic due to a couple of cryptographic weaknesses when not done properly.
Applications of Blind Signature

- In an online context the blind signature works as follows.
  - Voters encrypt their ballot with a secret key and then blinds it.
  - Then the voter signs the encrypted vote and sends it to the validator.
  - The validator checks to see if the signature is valid (the signature acts as a I.D. tag and will have to be registered with the voter before the voting process has started) and if it is the validator signs it and returns it to the voter.
  - The voter removes the blinding encryption layer, which then leaves behind an encrypted ballot with the validator's signature.
Cont.

- This is then sent to the tallier who checks to make sure the validator's signature is present on the votes.
- He then waits until all votes have been collected and then publishes all the encrypted votes so that the voters can verify their votes have been received.
- The voters then send their keys to the tallier to decrypt their ballots.
- Once the vote has been counted the tallier publishes the encrypted votes and the decryption keys so that voters can then verify the results.
- Next we illustrate the transfer of ballots between the various parties.
Cont.

Voter  

Blinded ballots

Signed ballots

Validator  

Unblinded ballots

Tallier  

Ballots

Keys

Keys and ballots
This protocol has been implemented used in reality and has been found that the entire voting process can be completed in a matter of minutes despite the complex nature of the voting procedure.

Most of the tasks can be automated with the only user interaction needed being the actual vote casting.

Encryption, blinding and all the verification needed can be performed by software in the background.

Of course we'd have to trust this software to handle the voting procedures correctly and accurately and to assume it has not been compromised in some way.
Cryptography and Network Security

Certificate

Xiang-Yang Li
Certificate

A public-key certificate is a digitally signed statement from one entity, saying that the public key (and some other information) of another entity has some specific value.
More terms

- **Digitally Signed**
  - If some data is *digitally signed* it has been stored with the "identity" of an entity, and a signature that proves that entity knows about the data. The data is rendered unforgeable by signing with the entity's private key.

- **Identity**
  - A known way of addressing an entity. In some systems the identity is the public key, in others it can be anything from a Unix UID to an Email address to an X.509 Distinguished Name.

- **Entity**
  - An entity is a person, organization, program, computer, business, bank, or something else you are trusting to some degree.
More about CA

Why need it

- In a large-scale networked environment it is impossible to guarantee that prior relationships between communicating entities have been established or that a trusted repository exists with all used public keys. Certificates were invented as a solution to this public key distribution problem. Now a Certification Authority (CA) can act as a Trusted Third Party. CAs are entities (e.g., businesses) that are trusted to sign (issue) certificates for other entities. It is assumed that CAs will only create valid and reliable certificates as they are bound by legal agreements. There are many public Certification Authorities, such as VeriSign, Thawte, Entrust, and so on. You can also run your own Certification Authority using products such as the Netscape/Microsoft Certificate Servers or the Entrust CA product for your organization.
Who uses Certificate?

- Probably the most widely visible application of X.509 certificates today is in web browsers (such as Netscape Navigator and Microsoft Internet Explorer) that support the SSL protocol.
  - SSL (Secure Socket Layer) is a security protocol that provides privacy and authentication for your network traffic. These browsers can only use this protocol with web servers that support SSL.

- Other technologies that rely on X.509 certificates include:
  - Various code-signing schemes, such as signed Java Archives, and Microsoft Authenticode.
  - Various secure E-Mail standards, such as PEM and S/MIME.
  - E-Commerce protocols, such as SET.
How to create certificate?

- There are two basic techniques used to get certificates:
  - you can create one yourself (using the right tools, such as [keytool](#))
    - Not everyone will accept self-signed certificates, 😊
  - you can ask a Certification Authority to issue you one (either directly or using a tool such as [keytool](#) to generate the request).

- The main inputs to the certificate creation are:
  - Matched public and private keys, generated using some special tools (such as keytool), or a browser.
  - Information about the entity being certified (e.g., you). This normally includes information such as your name and organizational address. If you ask a CA to issue a certificate for you, you will normally need to provide proof to show correctness of the information.
business

- Many companies sale the service of creating the certificate (such as SSL certificate)
  - Comodo
  - Verisign
  - Thawte
  - Entrust
  - Geotrust
X.509 Authentication Service

- Public key certificate associated with user
  - The certificates are created by Trusted Authority
  - Then placed in the directory by TA or user
  - Itself is not responsible for creating certificate
  - It includes
    - Version, serial number, signature algorithm identifier, Issuer name, issuer identifier, validity period, the user, user identifier, user's public key, extensions, signature by TA
  - The signature by TA guarantees the authority
  - Certificates can be used to certify other TAs
  - Y<<X>>: certificate of user X issued by TA Y
What is inside X.509 certificate?

- **Version**
  - Thus far, three versions are defined.

- **Serial Number**
  - Distinguish it from other certificates it issues. This information is used in numerous ways, for example when a certificate is revoked its serial number is placed in a Certificate Revocation List (CRL).

- **Signature Algorithm Identifier**
  - This identifies the algorithm used by the CA to sign the certificate.

- **Issuer Name**
  - The X.500 name of the entity that signed the certificate. This is normally a CA. Using this certificate implies trusting the entity that signed this certificate. *root or top-level CA certificates, the issuer signs its own certificate.*
Validity Period
- This period is described by a start date and time and an end date and time, and can be as short as a few seconds or almost as long as a century. It depends on a number of factors, such as the strength of the private key used to sign the certificate or the amount one is willing to pay for a certificate. This is the expected period that entities can rely on the public value, if the associated private key has not been compromised.

Subject Name
- The name of the entity whose public key the certificate identifies. This name uses the X.500 standard, so it is intended to be unique across the Internet.

Subject Public Key Information
- together with an algorithm identifier
Certificate Revocation

- Need the private key together with the certificate to revoke it
- The revocation is recorded at the directory
- Each time a certificate is arrived, check the directory to see if it is revoked
X.509 Authentication Service

- part of CCITT X.500 directory service standards
  - distributed servers maintaining some info database
- defines framework for authentication services
  - directory may store public-key certificates
  - with public key of user
  - signed by certification authority
- also defines authentication protocols
- uses public-key crypto & digital signatures
  - algorithms not standardised, but RSA recommended
X.509 Certificates

- issued by a Certification Authority (CA), containing:
  - 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)

- notation $\text{CA}<<\text{A}>>$ denotes certificate for A signed by CA
X.509 Certificates

(a) X.509 Certificate

(b) Certificate Revocation List
Obtaining a Certificate

- any user with access to CA can get any certificate from it
- only the CA can modify a certificate
- because cannot be forged, certificates can be placed in a public directory
CA Hierarchy

- if both users share a common CA then they are assumed to know its public key
- otherwise CA's must form a hierarchy
- use certificates linking members of hierarchy to validate other CA's
  - each CA has certificates for clients (forward) and parent (backward)
- each client trusts parents certificates
- enable verification of any certificate from one CA by users of all other CAs in hierarchy
CA Hierarchy Use
Certificate Revocation

- certificates have a period of validity
- may need to revoke before expiry, eg:
  1. user's private key is compromised
  2. user is no longer certified by this CA
  3. CA's certificate is compromised
- CA's maintain list of revoked certificates
  - the Certificate Revocation List (CRL)
- users should check certs with CA's CRL
Authentication Procedures

- X.509 includes three alternative authentication procedures:
  - One-Way Authentication
  - Two-Way Authentication
  - Three-Way Authentication
- all use public-key signatures
One-Way Authentication

- 1 message (A→B) used to establish
  - the identity of A and that message is from A
  - message was intended for B
  - integrity & originality of message

- message must include timestamp, nonce, B’s identity and is signed by A
Two-Way Authentication

- 2 messages (A→B, B→A) which also establishes in addition:
  - the identity of B and that reply is from B
  - that reply is intended for A
  - integrity & originality of reply

- reply includes original nonce from A, also timestamp and nonce from B
Three-Way Authentication

- 3 messages (A→B, B→A, A→B) which enables above authentication without synchronized clocks
- has reply from A back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon
X.509 Version 3

- has been recognised that additional information is needed in a certificate
  - email/URL, policy details, usage constraints
- rather than explicitly naming new fields defined a general extension method
- extensions consist of:
  - extension identifier
  - criticality indicator
  - extension value
Certificate Extensions

- **key and policy information**
  - convey info about subject & issuer keys, plus indicators of certificate policy

- **certificate subject and issuer attributes**
  - support alternative names, in alternative formats for certificate subject and/or issuer

- **certificate path constraints**
  - allow constraints on use of certificates by other CA’s