Digital Signature

The authenticity of many legal, financial, and other documents is determined by the presence or absence of an authorized handwritten signature.

Various methods have been devised to solve this problem, but the use of ‘digital signature’ is definitely the best solution amongst them.

A digital signature is nothing but an attachment to any piece of electronic information, which represents the content of the document and the identity of the originator of that document uniquely.

- Use of signatures is recorded in the Talmud (fourth century), complete with security procedures to prevent the alteration of documents after they are signed.

- The practice of authenticating documents by affixing handwritten signatures began to be used within the Roman Empire in the year AD 439, during the rule of Valentinian III.

- It is from this Roman usage of signatures that the practice obtained its significance in Western legal tradition.

What is digital signature

- Informally, a **digital signature** is a technique for establishing the origin of a particular message in order to settle later disputes about what message (if any) was sent.
- The purpose of a digital signature is thus for an entity to bind its identity to a message.
- We use the term **signer** for an entity who creates a digital signature, and the term **verifier** for an entity who receives a signed message and attempts to check whether the digital signature is "correct" or not.
- Digital signatures have many attractive properties and it is very important to understand exactly what assurances they provide and what their limitations are.
- While data confidentiality has been the driver behind historical cryptography, digital signatures could be the major application of cryptography in the years to come.

- Hash value of a message when encrypted with the private key of a person is his digital signature on that e-Document.

- Digital Signature of a person varies from document to document thus ensuring authenticity of each word of that document.

- As the public key of the signer is known, anybody can verify the message and the digital signature.

Why Digital Signatures

- To provide Authenticity, Integrity and Non-repudiation to electronic documents

- To use the Internet as the safe and secure medium for Banking, e-Commerce and e-Governance with Security of Servers

Authenticity

- Although messages may often include information about the entity sending a message, that information may not be accurate. Digital signatures can be used to authenticate the source of messages. When ownership of a digital signature secret key is bound to a specific user, a valid signature shows that the message was sent by that user. The importance of high confidence in sender authenticity is especially obvious in a financial context.
- For example, suppose a bank's branch office sends instructions to the central office requesting a change in the balance of an account. If the central office is not convinced that such a message is truly sent from an authorized source, acting on such a request could be a grave mistake.

Integrity

- In many scenarios, the sender and receiver of a message may have a need for confidence that the message has not been altered during transmission. Although encryption hides the contents of a message, it may be possible to *change* an encrypted message without understanding it. (Some encryption algorithms, known as nonmalleable ones, prevent this, but others do not.)
- However, if a message is digitally signed, any change in the message will invalidate the signature. Furthermore, there is no efficient way to modify a message and its signature to produce a new message with a valid signature, because this is still considered to be computationally infeasible by most cryptographic hash functions.

BASIC REQUIREMENTS

- **Private Key**

  The private key is one which is accessible only to the signer. It is used to generate the digital signature which is then attached to the message.

- **Public Key**

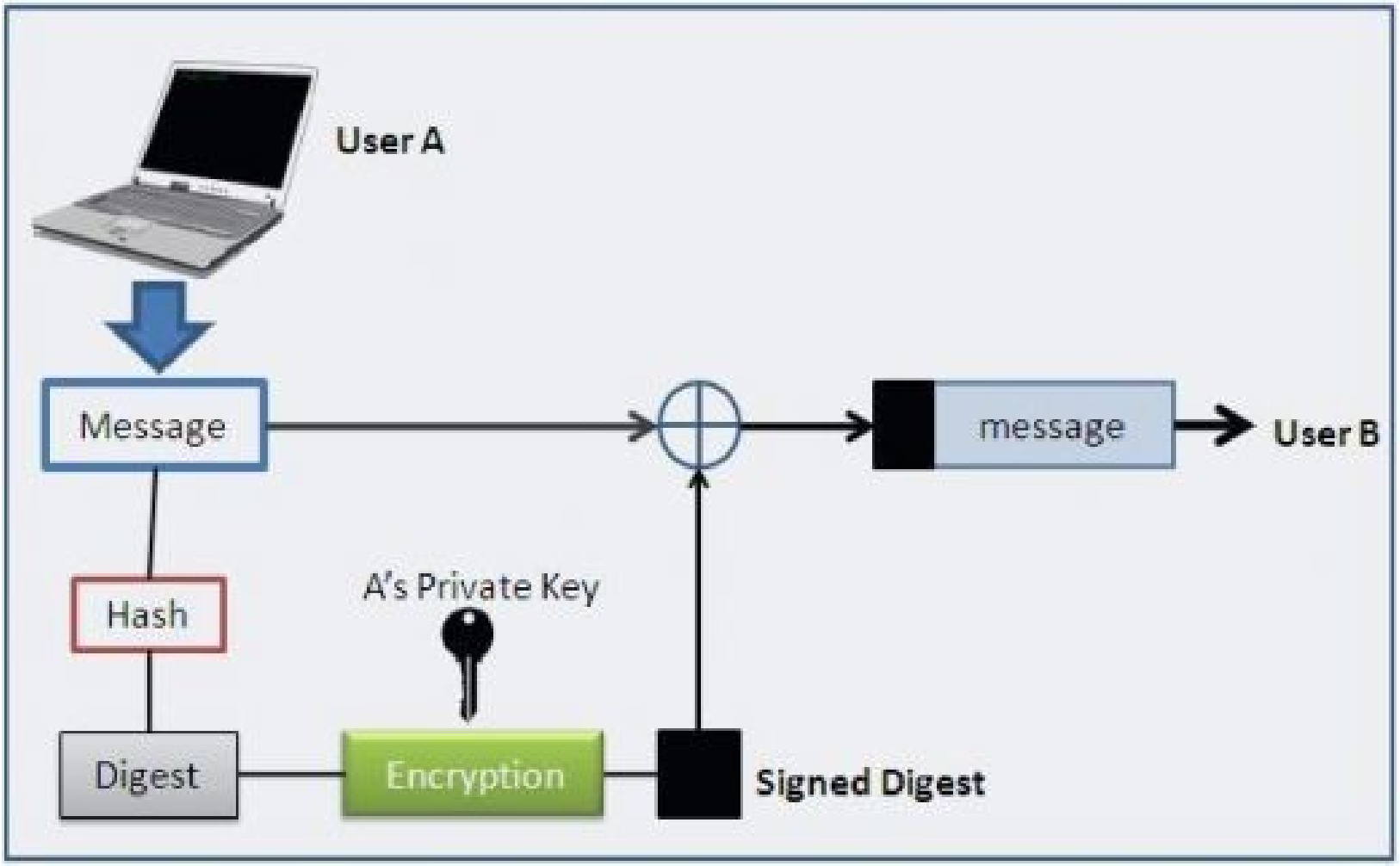
  The public key is made available to all those who receive the signed messages from the sender. It is used for verification of the received message.

- **Digital Signature Certificate**

A subscriber of the private key and public key pair makes the public key available to all those who are intended to receive the signed messages from the subscriber.

But in case of any dispute between the two sides, there must be some entity with the receiver which will allow the receiver of the message to prove that the message was indeed sent by the subscriber of the key pair. This can be done with the Digital Signature Certificate.

HOW THE TECHNOLOGY WORKS



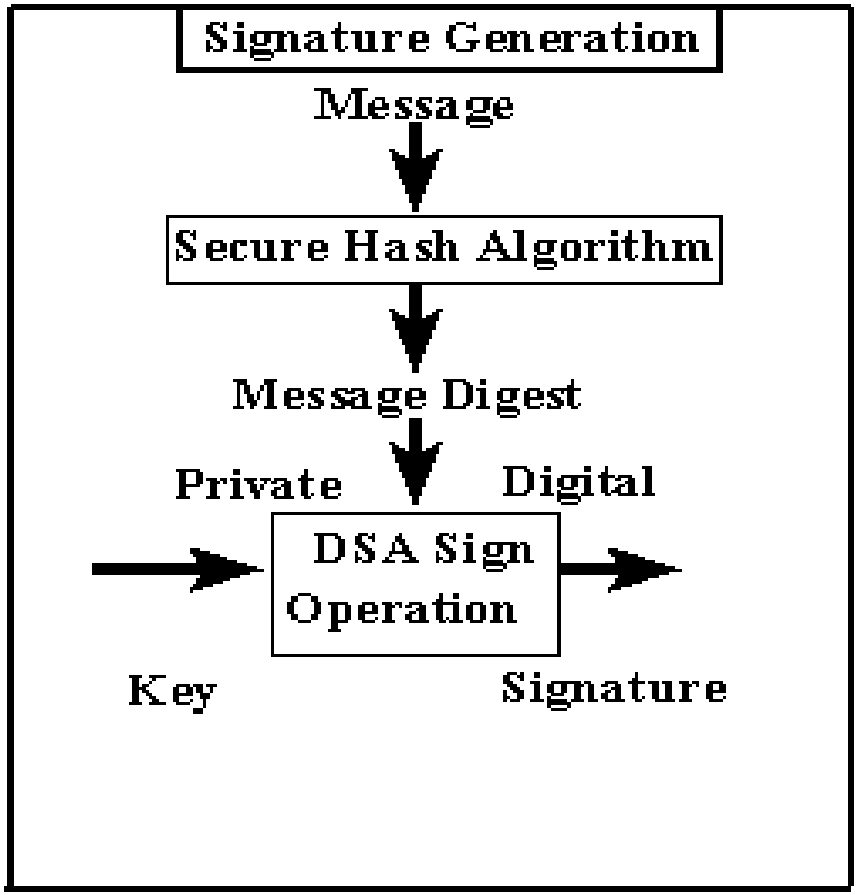
How it works

- The use of digital signatures usually involves two processes, one performed by the signer and the other by the receiver of the digital signature:

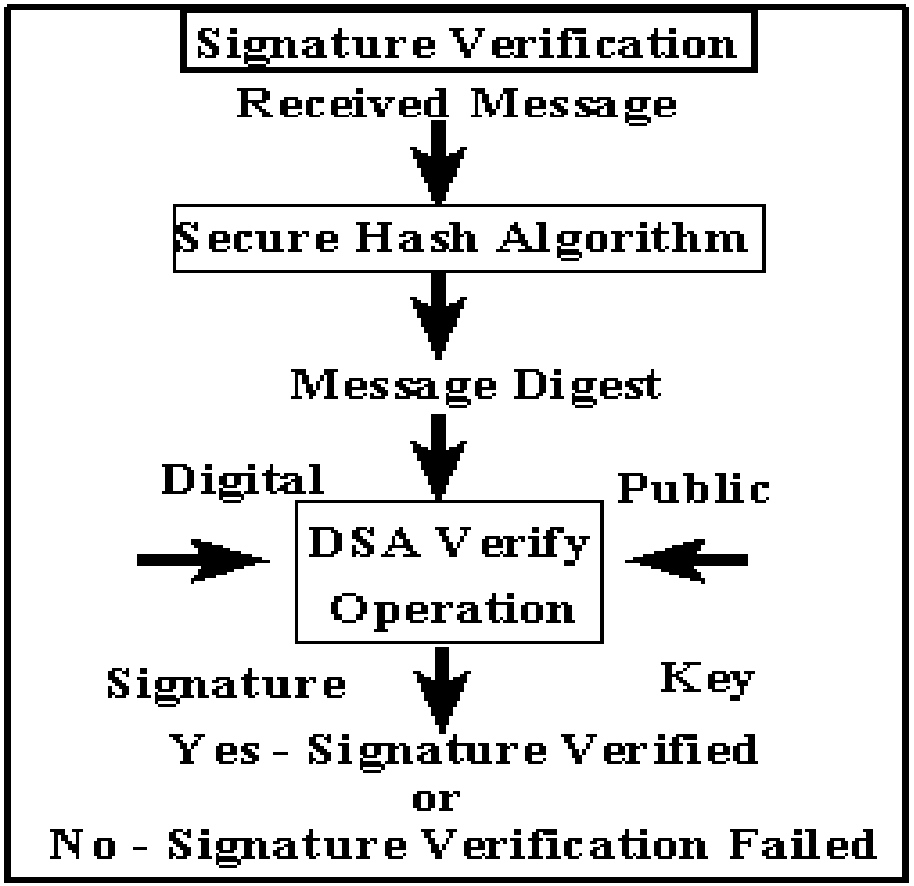
- **Digital signature** creation uses a hash result derived from and unique to both the signed message and a given private key. For the hash result to be secure, there must be only a negligible possibility that the same digital signature could be created by the combination of any other message or private key.

- **Digital signature verification** is the process of checking the digital signature by reference to the original message and a given public key, thereby determining whether the digital signature was created for that same message using the private key that corresponds to the referenced public key.

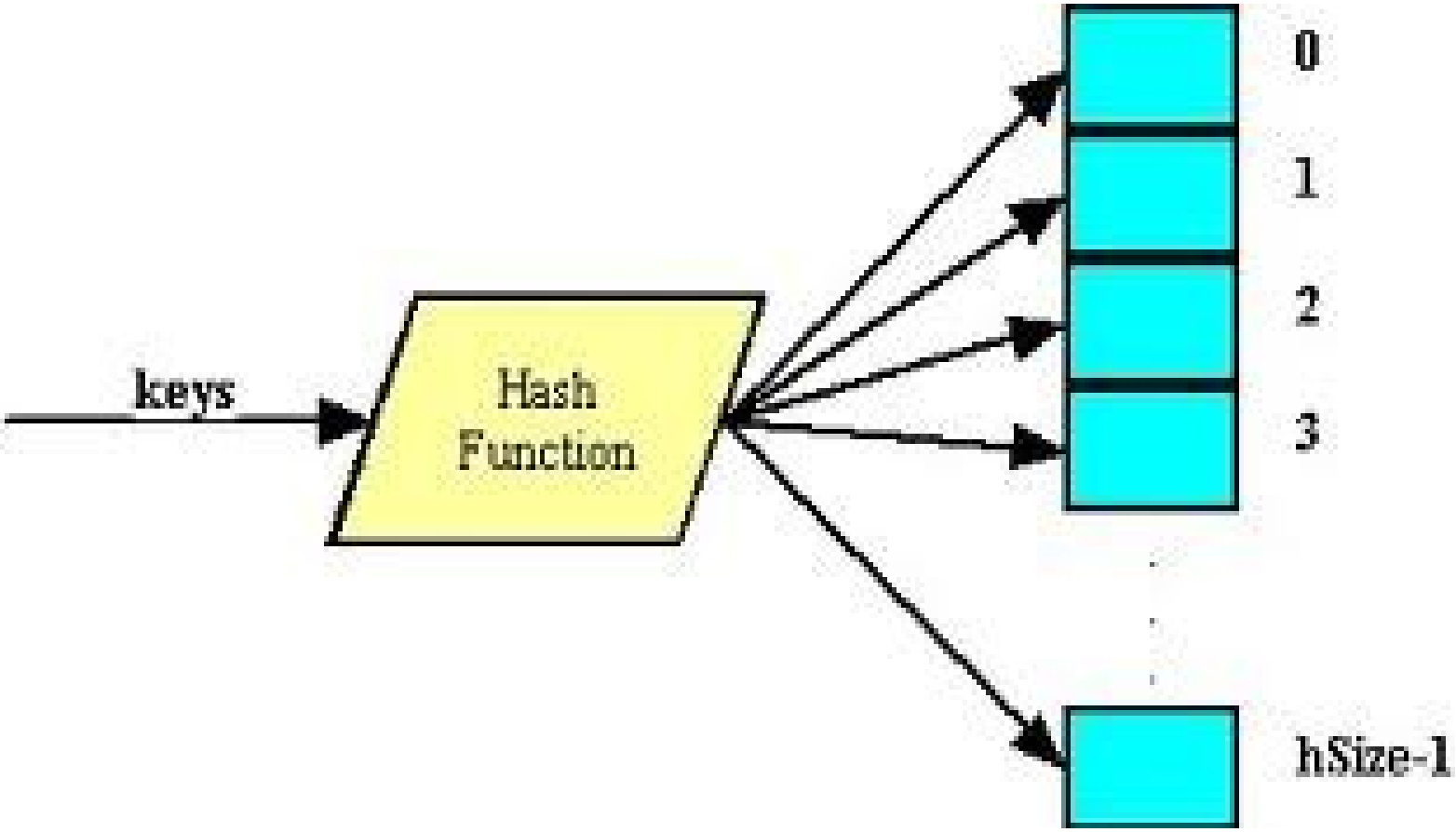
Digital Signature Generation



Digital Signature Verification



Secure Hash Algorithm



Properties of a digital signature

- **Easy for the signer to sign a message**

  There is no point in having a digital signature scheme that involves the signer needing to use slow and complex operations to compute a digital signature.

- **Easy for anyone to verify a message**

  Similarly we would like the verification of a digital signature to be as efficient as possible

- **Hard for anyone to forge a digital signature**

  It should be practically impossible for anyone who is not the legitimate signer to compute a digital signature on a message that appears to be valid. By “appears to be valid” we mean that anyone who attempts to verify the digital signature is led to believe that they have just successfully verified a valid digital signature on a message.

- ✓ **Institutional overhead:**

  The cost of establishing and utilizing certification authorities, repositories, and other important services, as well as assuring quality in the performance of their functions.

- ✓ **Subscriber and Relying Party Costs:**

  A digital signer will require software, and will probably have to pay a certification authority some price to issue a certificate. Hardware to secure the subscriber's private key may also be advisable.

APPLICATIONS

Electronic Mail

Data storage

Electronic funds transfer

Software Distribution

eGovernance Applications

DRAWBACKS

- The private key must be kept in a secured manner.
- The process of generation and verification of digital signature requires considerable amount of time.
- For using the digital signature the user has to obtain private and public key, the receiver has to obtain the digital signature certificate also.

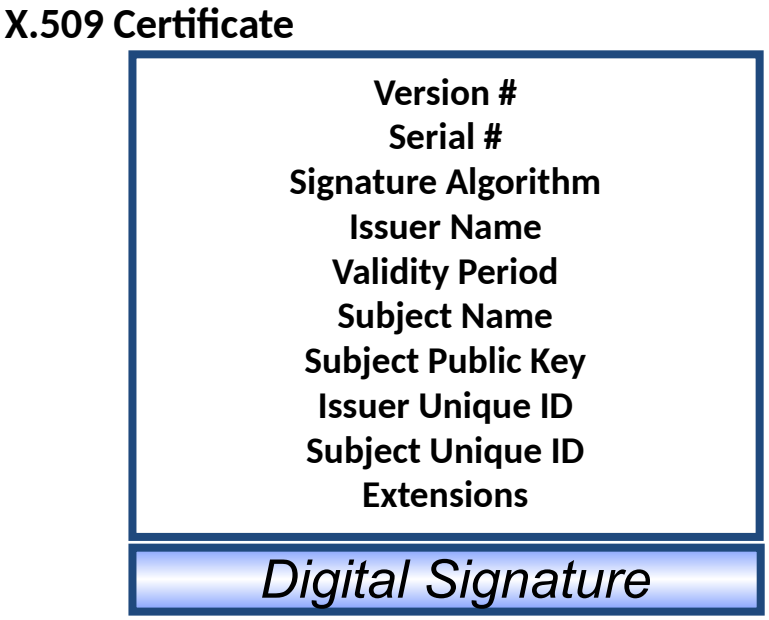
Public Key Infrastructure

- Public Key Infrastructure = PKI
- PKIs bind an identity to a public key
- Certificate Authority Based – Most familiar ← Web of Trust Based – PGP's model
- PKIs enable encryption and sender authentication for email, authentication of servers to browsers, authentication of users to applications, etc.

Digital Certificates

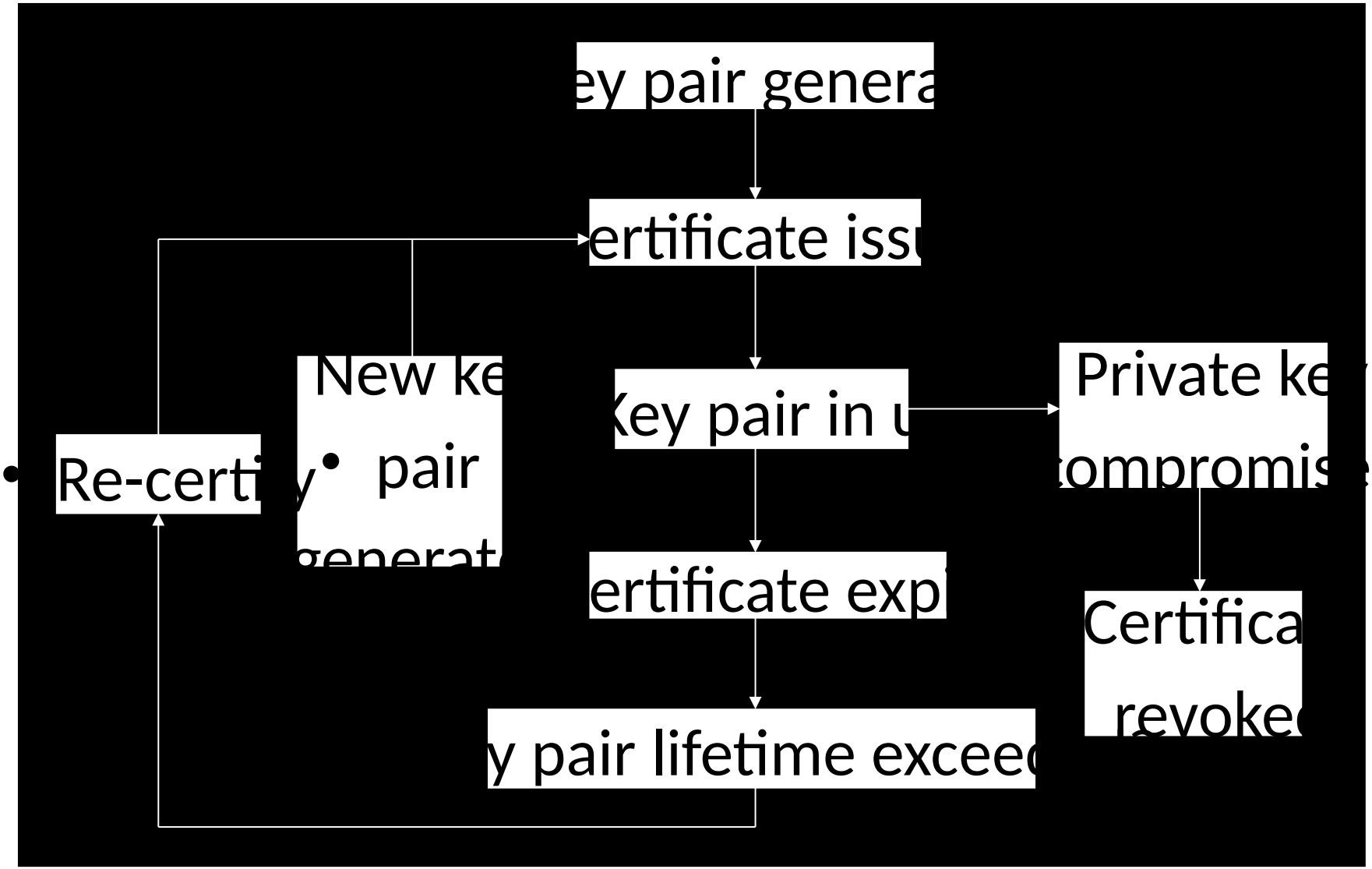
- Before two parties exchange data using Public Key cryptography, each wants to be sure that the other party is authenticated
- Before B accepts a message with A's Digital Signature, B wants to be sure that the public key belongs to A and not to someone masquerading as A on an open network
- One way to be sure, is to use a trusted third party to authenticate that the public key belongs to A. Such a party is known as a **Certification Authority (CA)**
- Once A has provided proof of identity, the Certification Authority creates a message containing A's name and public key. This message is known as a **Digital Certificate**.

- A Digital Certificate is simply an X.509 defined data structure with a Digital Signature. The data represents who owns the certificate, who signed the certificate, and other relevant information



- When the signature is generated by a Certification Authority (CA), the signature can be viewed as trusted.
- Since the data is signed, it can not be altered without detection.
- Extensions can be used to tailor certificates to meet the needs of end applications.

Certificate Life Cycle



Certificate Authority Model

- A Certificate Authority (CA) sits at the top of a trust hierarchy
- CAs issue digital certificates that contain identity information about the subject, expiration and revocation information, and the subject's public key
- CAs sign digital certificates they issue. If you trust a CA, you trust any certificate they sign that hasn't expired or been revoked
- CAs can be internal to a business, government, or organization or they can be they can be large forprofit multi-national corporations

- 1. Key Generation
- 2. Registration
- 3 Verification

4. Certificate Creation

PKI Players

- Registration Authority (RA) to identity proof users
- Certification Authorities (CA) to issue certificates and CRL's
- Repositories (publicly available databases) to hold certificates and CRLs

Certification Authority (CA)

- ***Certification Authority***
  - Trusted (Third) Party
  - Enrolls and Validates Subscribers
  - Issues and Manages Certificates
  - Manages Revocation and Renewal of Certificates
  - Establishes Policies & Procedures

What's Important

- Operational Experience
- High Assurance Security Architecture
- Scalability
- Flexibility
- Interoperability
- Trustworthiness

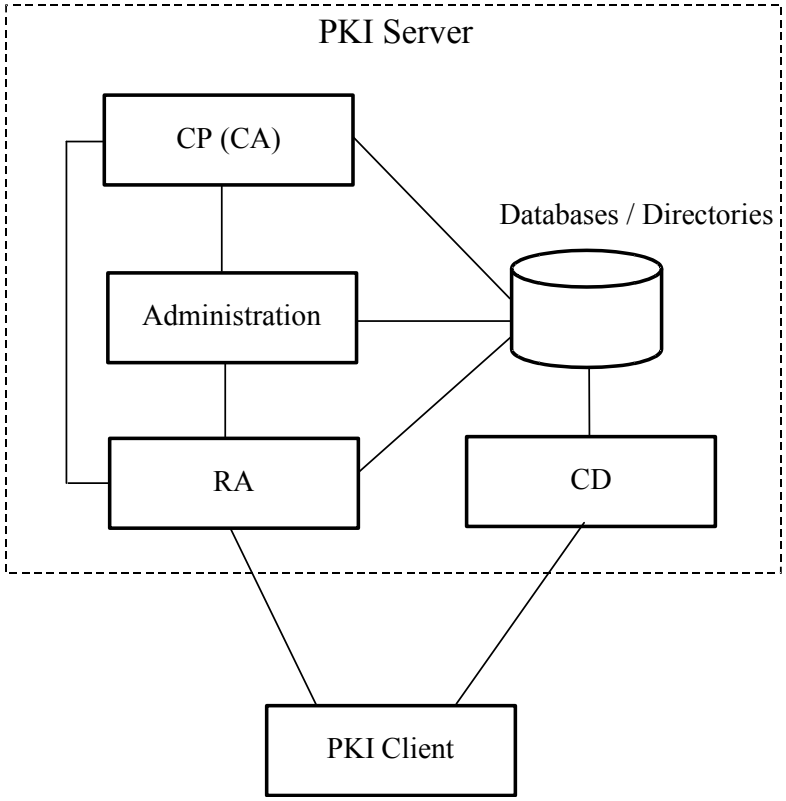
Registration Authority (RA)

- Enrolling, de-enrolling, and approving or rejecting requested changes to the certificate attributes of subscribers.
- Validating certificate applications.
- Authorizing requests for key-pair or certificate generation and requests for the recovery of backed-up keys.
- Accepting and authorizing requests for certificate revocation or suspension.
- Physically distributing personal tokens to and recovering obsolete tokens from people authorized to hold and use them.

Certificate Policy (CP) is

- the basis for trust between unrelated entities
- not a formal “contract” (but implied)
- a framework that both informs and constrains a PKI implementation
- a statement of what a certificate means
- a set of rules for certificate holders
- a way of giving advice to Relying Parties

Certificate Processor/Authority



Registration Authority

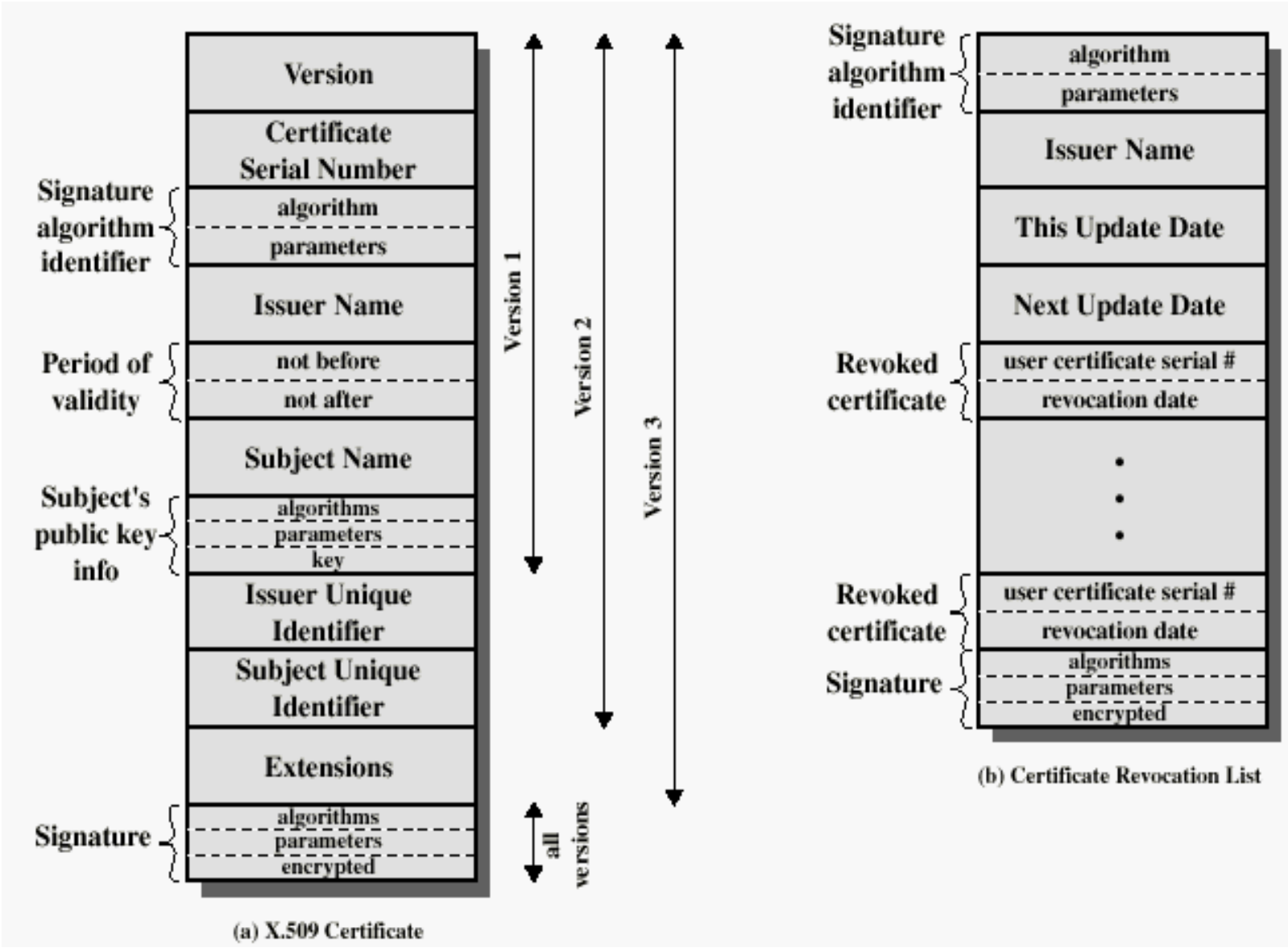
Certificate Distribution

Architecture of a typical organization-wide PKI

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 `CA<<A>>`
  - denotes certificate for A signed by CA

X.509 Formats



Revocation of Certificates

- Reasons for revocation:
  - The user's private key is assumed to be compromised.
  - The user is no longer certified by this CA.
  - The CA's certificate is assumed to be compromised.

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
  - Timestamp (t_A)
    - Message generation and expiration time
  - Nonce (r_A)
    - Value must be unique within the expiration time
    - Used to detect replay attacks
- may include additional info for B
  - Eg. session key

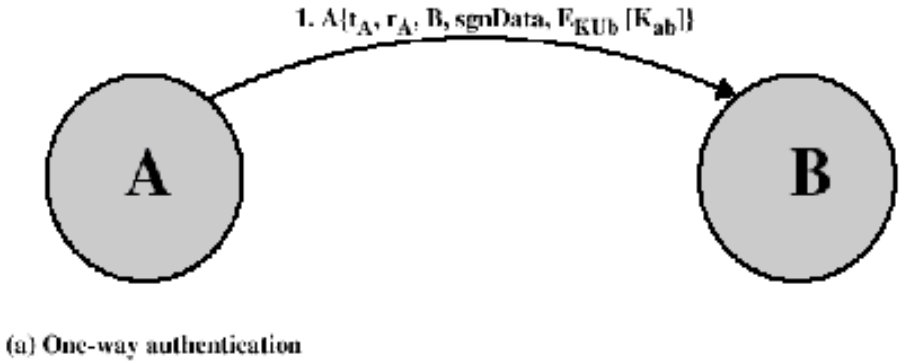
Two-Way Authentication

- 2 messages (A->B, B->A) which also establishes:
  - 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
- may include additional info for A

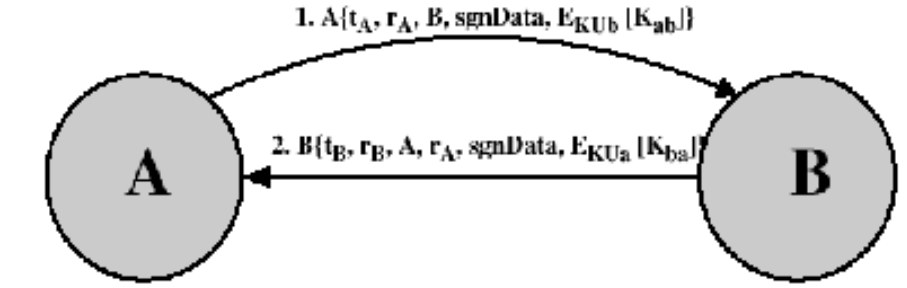
Three-Way Authentication

- 3 messages (A->B, B->A, A->B) to enable 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

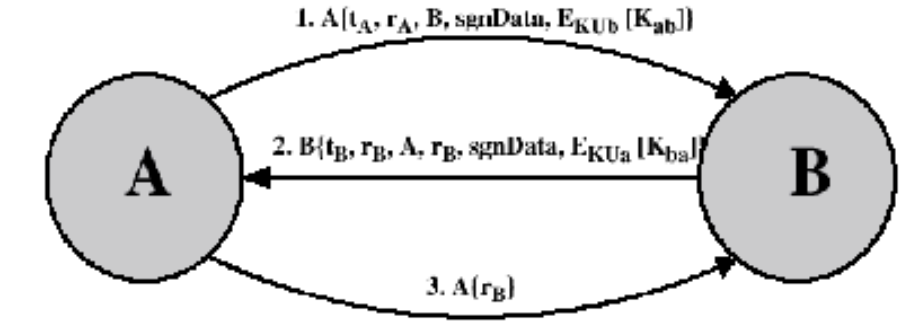
Authentication Procedures



(a) One-way authentication



(b) Two-way authentication



(c) Three-way authentication

Figure 4.5 X.509 Strong Authentication Procedures

Secure Socket Layer

A protocol developed by Netscape.

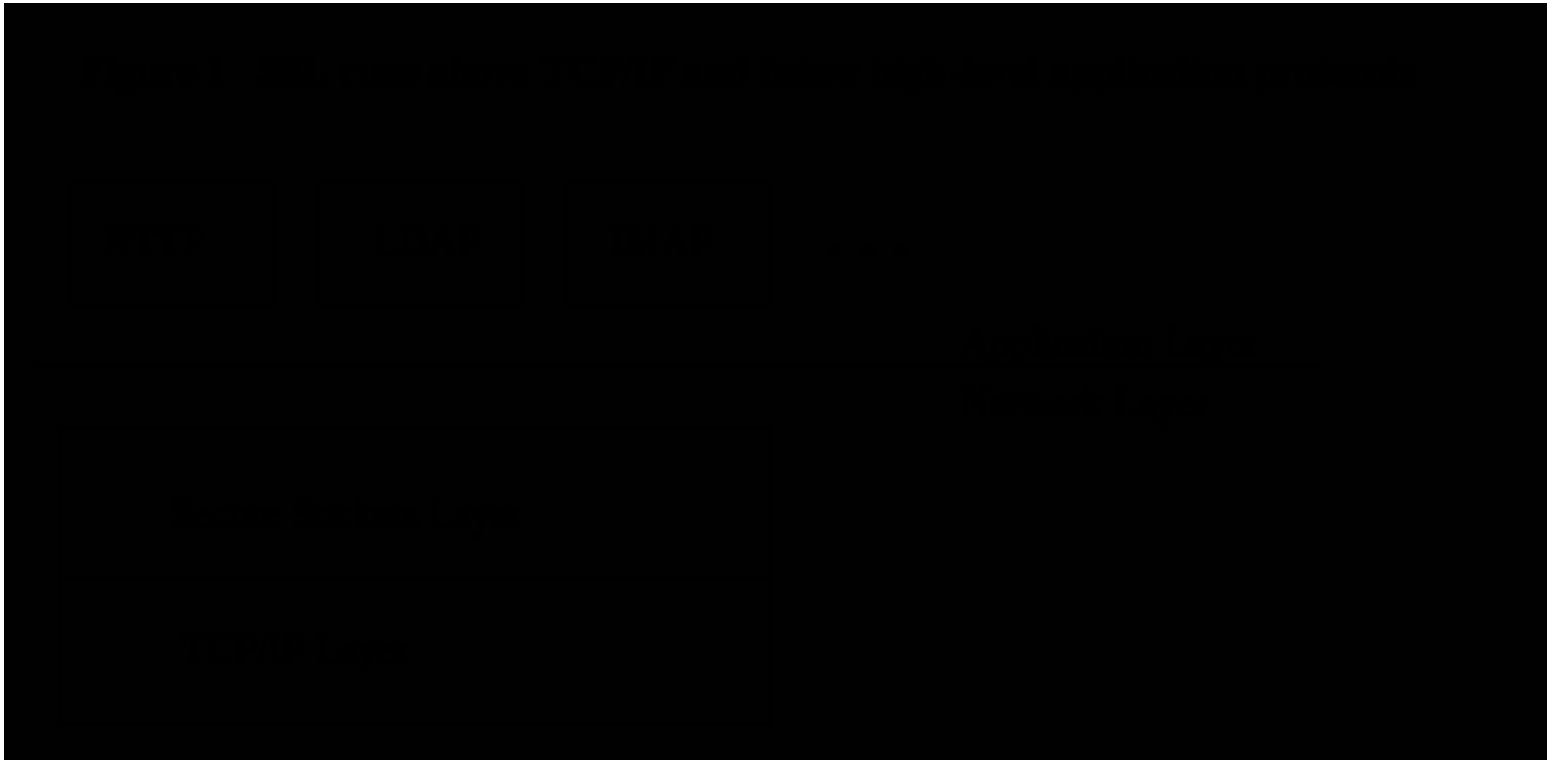
SSL version 3.0 has been implemented in many web browsers (e.g., Netscape Navigator and MS Internet Explorer) and web servers and widely used on the Interne

SSL v3.0 was specified in an Internet Draft (1996)

It is a whole new layer of protocol which operates above the Internet TCP protocol and below high-level application protocols

Encrypt the web traffic between two sites, so no one can listen in and get credit card numbers

Uses something called “Secure Sockets Layer” (SSL)



What Can SSL Do?

- SSL uses TCP/IP on behalf of the higher-level protocols.
- Allows an SSL-enabled server to authenticate itself to an SSL-enabled client;
- Allows the client to authenticate itself to the server;
- Allows both machines to establish an encrypted connection.

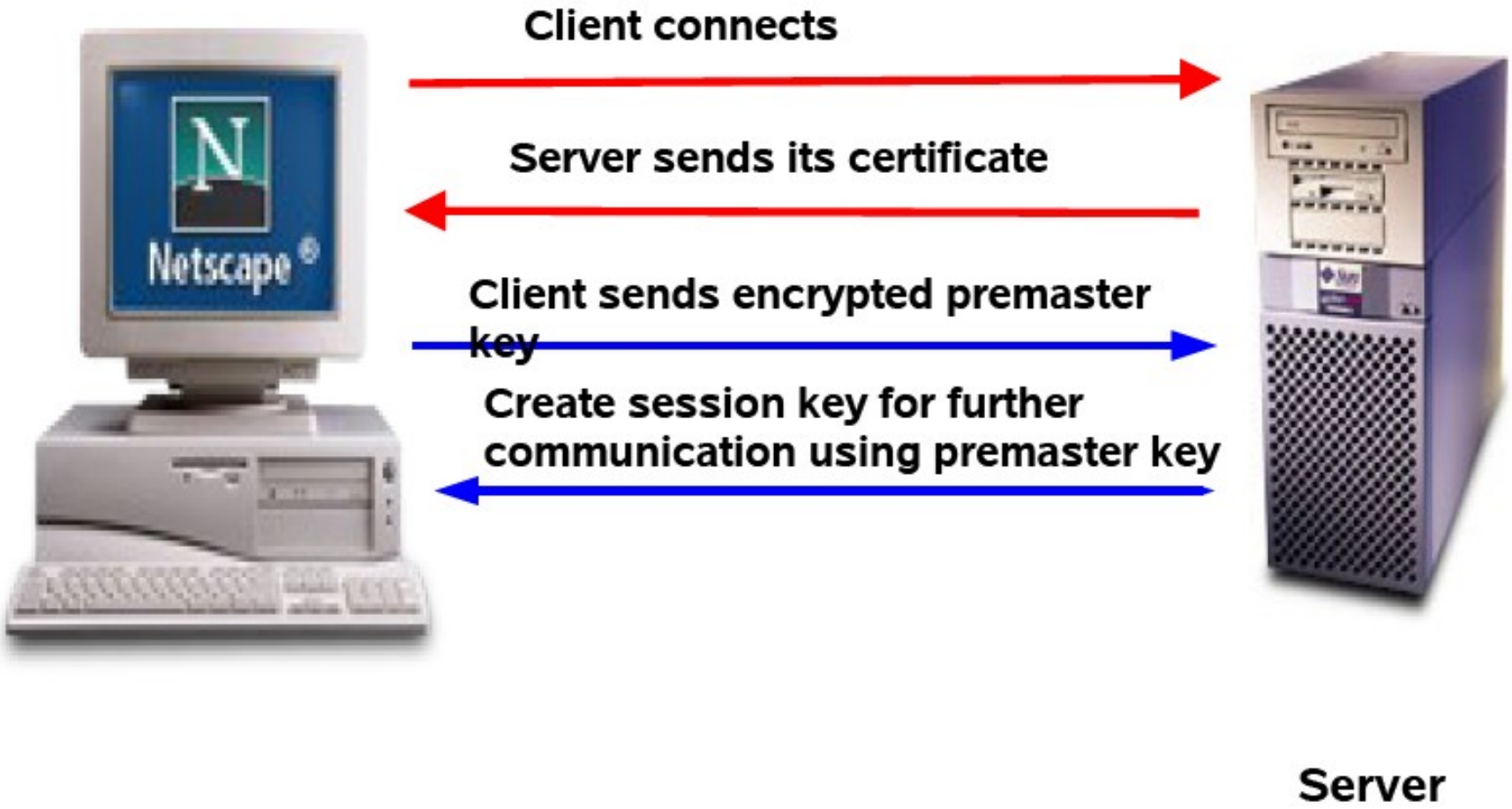
What Does SSL Concern?

- SSL server authentication.
- SSL client authentication. (optional)
- An encrypted SSL connection or Confidentiality. This protects against electronic eavesdropper.
- Integrity. This protects against hackers.

- Confidentiality (Privacy)
- Data integrity (Tamper-proofing)
- Server authentication (Proving a server is what it claims it is)
  - Used in typical B2C transaction
- Optional client authentication
  - Would be required in B2B (or Web services environment in which program talks to program)

SSL and Security Keys

- Uses public/private key (asymmetric) scheme to create secret key (symmetric)
- Secret key is then used for encryption of data
  - SSL operation is optimized for performance: Using symmetric key for encryption is a lot faster than using asymmetric keys


Client connects
Server sends its certificate
Client sends encrypted premaster key
Create session key for further communication using premaster key
Netscape
Server

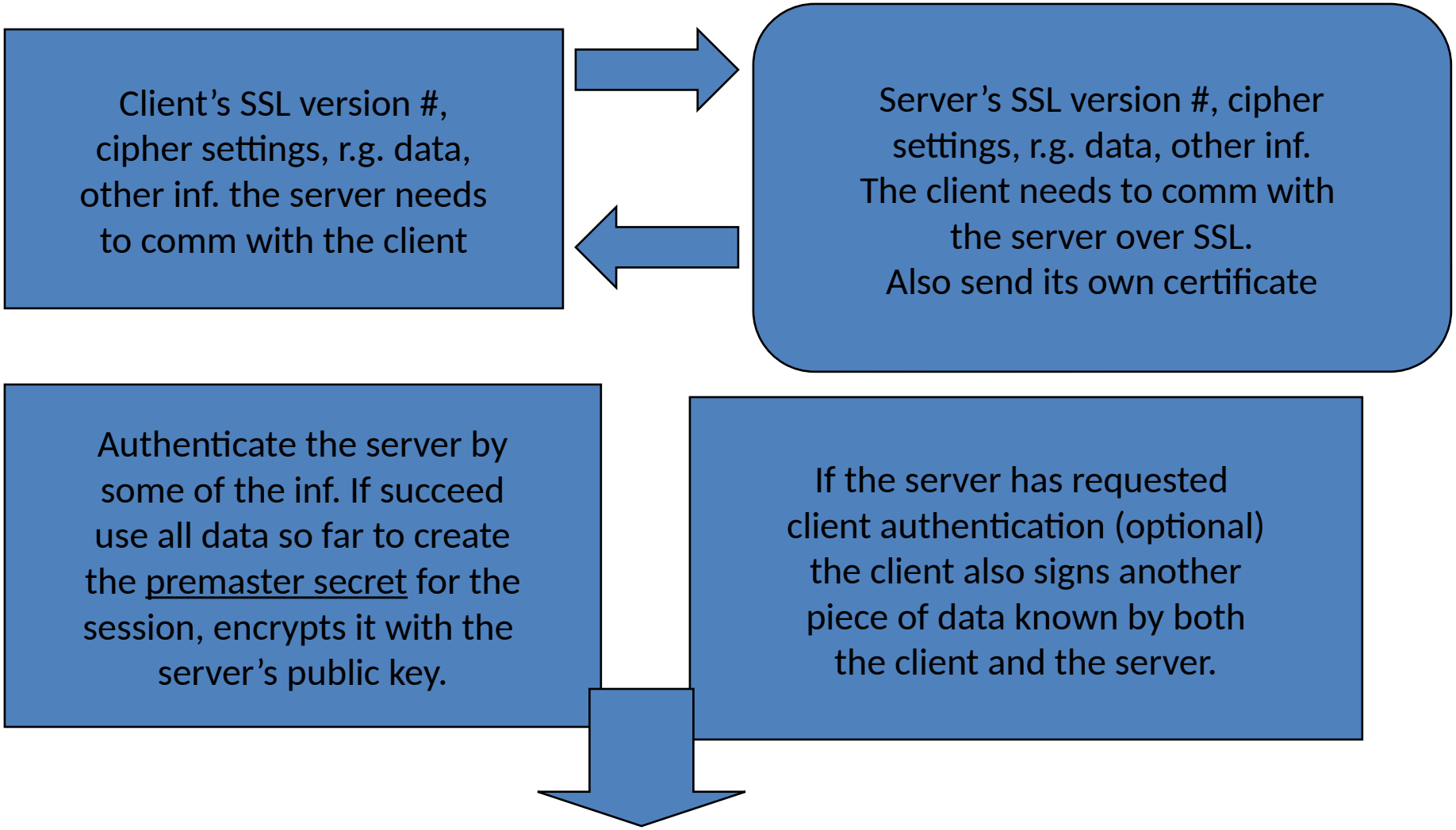
SSL Key Exchange Steps

1. SSL client connects to an SSL server
2. Server then sends its own certificate that contains its public key
3. Client then creates a random key (premaster key) and uses server's public key to encrypt it
4. Client then sends encrypted premaster key to the server
5. Server then decrypts it (only the server that has the matching private key can decrypt it) and uses decrypted premaster key to create secret session key
6. Now both client and server uses secret session key for further communication

SSL components

- SSL Handshake Protocol
  - negotiation of security algorithms and parameters
  - key exchange
  - server authentication and optionally client authentication
- SSL Record Protocol
  - fragmentation
  - compression
  - message authentication and integrity protection
  - encryption
- SSL Alert Protocol
  - error messages (fatal alerts and warnings)
- SSL Change Cipher Spec Protocol
  - a single message that indicates the end of the SSL handshake

How Does SSL Work?



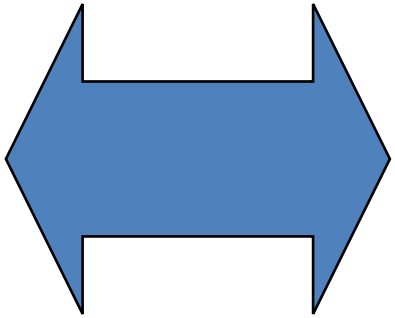
If the server has requested client authen., the server attempts to authen the client. If succeed, uses its private key decrypt the premaster secret, then perform a series of steps to generate the master secret
Use the master secret to generate the session keys.

Also performs a series of steps, starting from the same premaster secret to generate the master secret.
Use the master secret to generate the session keys

- Session keys are used to encrypt and decrypt information exchange during the SSL session and to verify its integrity
- Master secrets protect session keys in transit.

Informing the server that the future message from here will be encrypted with the session key.

Then sends a separate (encrypted) message indicating that the client portion of handshake is finished.



Informing the client that the future message from here will be encrypted with the session key.

Then sends a separate (encrypted) message indicating that the server portion of handshake is finished.

Implementation

- The secure web site includes a digital certificate signed by some certificate authority. The certificate includes the server name, its public key, IP number, and an expiration date. It is typically signed with a 1024 bit key by the CA

- The list of certificate authorities that you trust to identify people is available in Netscape by clicking on the lock icon at top; in IE, Internet Options->Content

How It Works

- The browser reads the site certificate; if it is signed by one of the trusted certificate authorities, browser accepts the certificate as valid

- If the certificate is signed by some unknown certificate authority, Netscape will ask you if you want to trust the guy who signed it

How It Works (Basic Protocol)

- The browser negotiates a secure session using something like the following protocol:

1: A->B: hello
2: B->A: Hi, I'm Bob, bobs-certificate
3: A->B: prove it
4: B->A: Alice, This Is bob
{ digest[Alice, This Is Bob] } bobs-private-key
5: A->B: ok bob, here is a secret {secret} bobs-public-key
6: B->A: {some message}secret-key

How It Works

Step 1: your browser introduces itself to the secure server

Step 2: the server responds by sending back a message with the certificate included

Step 3: Your browser tells the secure site to prove its identity, that it really is who it says it is.

Step 4: The secure server proves who it is by creating a message for the browser, generating a “fingerprint” of that message, and encrypting the “fingerprint” with the private key that is matched to the public key in the certificate. The browser generates the “fingerprint” for the message itself, then decrypts the “fingerprint” generated by the server using the public key provided in the certificate.

How It Works

- At this point the browser is sure that the server is how it says it is. It can send it secret messages encrypted with the public key provided in the certificate. The server (and only the server) can decrypt these messages, because only it has the private key.

- At this point what typically happens is that the browser generates a *session key* using a completely different encryption algorithm. A new session key is generated for every connection; this does not have to be a public key algorithm. You can use any encryption algorithm you like; usually a faster conventional, non-PK algorithm is used. This is usually 40 or 128 bits long in Netscape.

- You’ll use a completely different key for encrypting traffic to the web site every time you connect. This makes cracking communication more difficult; you need to discover the keys for every session rather than just one key.

SSL-Handshake Protocol

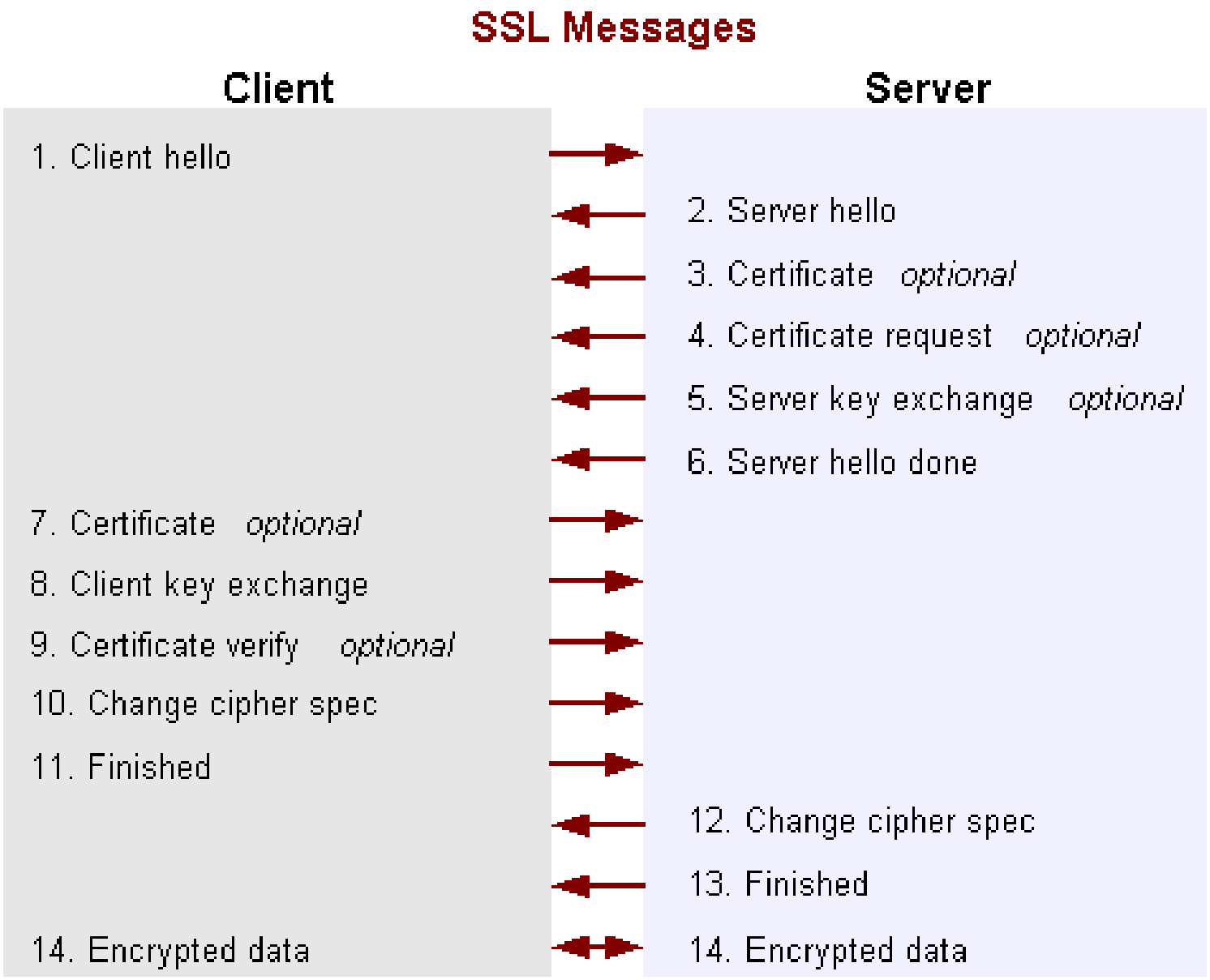
- This is the most complex part of SSL and allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record.

- This protocol is used before any application data is sent. It consists of a series of messages exchanged by the client and server. Each message has three fields:

1. Type (1 byte): Indicates one of 10 messages such as “hello request”.

2. Length (3 bytes): The length of the message in bytes.

3. Content(≥ 0 byte): The parameters associated with this message such version of SSL being used.


SSL Messages
Client
Server
1. Client hello
2. Server hello
3. Certificate optional
4. Certificate request optional
5. Server key exchange optional
6. Server hello done
7. Certificate optional
8. Client key exchange
9. Certificate verify optional
10. Change cipher spec
11. Finished
12. Change cipher spec
13. Finished
14. Encrypted data
14. Encrypted data

- The Handshake Protocol consists of four phases:

1. Establish security capabilities including protocol version, session ID, cipher suite, compression method and initial random numbers. This phase consists of the client hello and server hello messages which contain the following (this is for the client however it's a little different for the server)

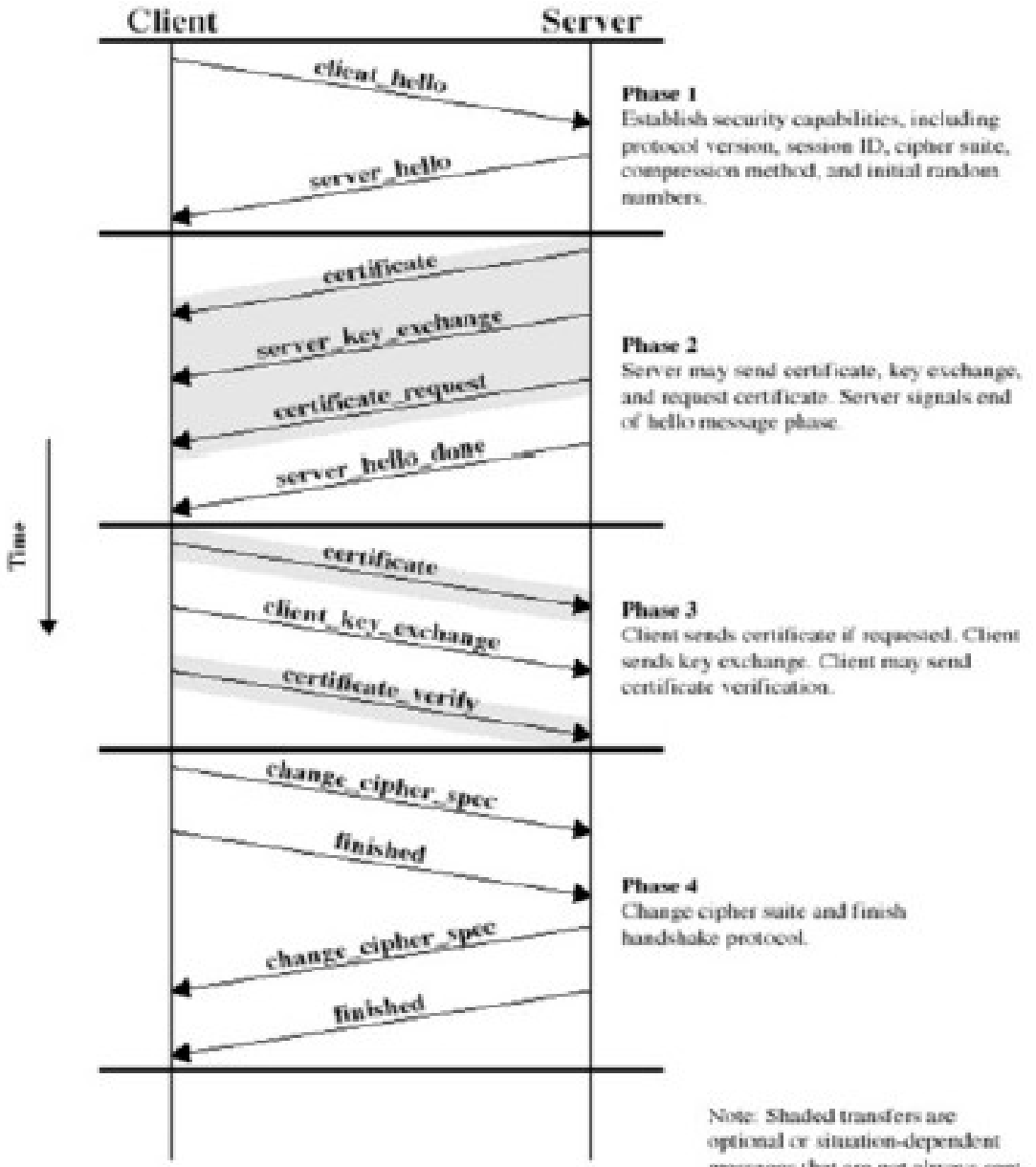
   Version: The highest SSL version understood by client

- Random: 32-bit timestamp and 28 byte nonce.
- Session ID: A variable length session identifier.
- CipherSuite: List of cryptoalgorithms supported by client in decreasing order of preference. Both key exchange and CipherSpec (this includes fields such as CipherAlgorithm, MacAlgorithm, CipherType, HashSize, Key Material and IV Size) are defined.
- Compression Method: List of methods supported by client.

2. Server may send certificate, key exchange, and request certificate it also signals end of hello message phase. The certificate sent is one of a chain of X.509 certificates discussed earlier in the course. The server key exchange is sent only if required. A certificate may be requested from the client if needs be by certificate request.

3. Upon receipt of the server done message, the client should verify that the server provided a valid certificate, if required, and check that the server hello parameters are acceptable. If all is satisfactory, the client sendback to the server. The client sends certificate if requested (if none available then it sends a no certificate alert instead). Next the client sends client key exchange message . Finally, the client may send certificate verification.

- Change cipher suite and finish handshake protocol. The secure connection is now setup and the client and server may begin to exchange application layer data.


Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.