Security of RSA-Timing Attacks

Timing Attacks:

- A timing attack is somewhat analogous to a burglar guessing the combination of a safe by observing how long it takes for someone to turn the dial from number to number.

- In this algorithm, modular exponentiation is accomplished bit by bit, with one modular multiplication performed at each iteration and an additional modular multiplication performed for each 1 bit.

- The attack proceeds bit-by-bit starting with the leftmost bit, bk. Suppose that the first j bits are known (to obtain the entire exponent, start with j = 0 and repeat the attack until the entire exponent is known).

- Therefore, if the observed time to execute the decryption algorithm is always slow when this particular iteration is slow with a 1 bit, then this bit is assumed to be 1. If a number of observed execution times for the entire algorithm are fast, then this bit is assumed to be 0.

Security of RSA-Timing Attacks

Counter Measures of Timing Attack

- **Constant exponentiation time**: Ensure that all exponentiations take the same amount of time before returning a result. This is a simple fix but does degrade performance.
- **Random delay**: Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.
- **Blinding**: Multiply the cipher text by a random number before performing exponentiation. This process prevents the attacker from knowing what cipher text bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack.

Security of RSA-Timing Attacks

RSA Data Security incorporates a blinding feature into some of its products. The private-key operation $M = C^d \bmod n$ is implemented as follows:

1. Generate a secret random number r between 0 and n 1.

2. Compute $C' = C(r^e) \bmod n$, where e is the public exponent.

3. Compute $M' = (C')^d \bmod n$ with the ordinary RSA implementation.

4. Compute $M = M'r^1 \bmod n$. In this equation, r^1 is the multiplicative inverse of r mod n; see Chapter 8 for a discussion of this concept. It can be demonstrated that this is the correct result by observing that $r^{ed} \bmod n = r \bmod n$.

Chosen Cipher text Attack and Optimal Asymmetric Encryption Padding

- CCA is defined as an attack in which adversary chooses a number of ciphertexts and is then given the corresponding plaintexts, decrypted with the target's private key.
- Thus, the adversary could select a plaintext, encrypt it with the target's public key and then be able to get the plaintext back by having it decrypted with the private key.

$$\mathrm{E}(PU, M_1) \times \mathrm{E}(PU, M_2) = \mathrm{E}(PU, [M_1 \times M_2])$$

We can decrypt $C = M^e$ using a CCA as follows.

1. Compute $X = (C \times 2^e) \bmod n$.
2. Submit X as a chosen ciphertext and receive back $Y = X^d \bmod n$.

But now note the following:

$$X = (C \bmod n) \times (2^e \bmod n)$$

$$= (M^e \bmod n) \times (2^e \bmod n)$$

$$= (2M)^e \bmod n$$

Therefore, $Y = (2M) \bmod n$ From this, we can deduce M.

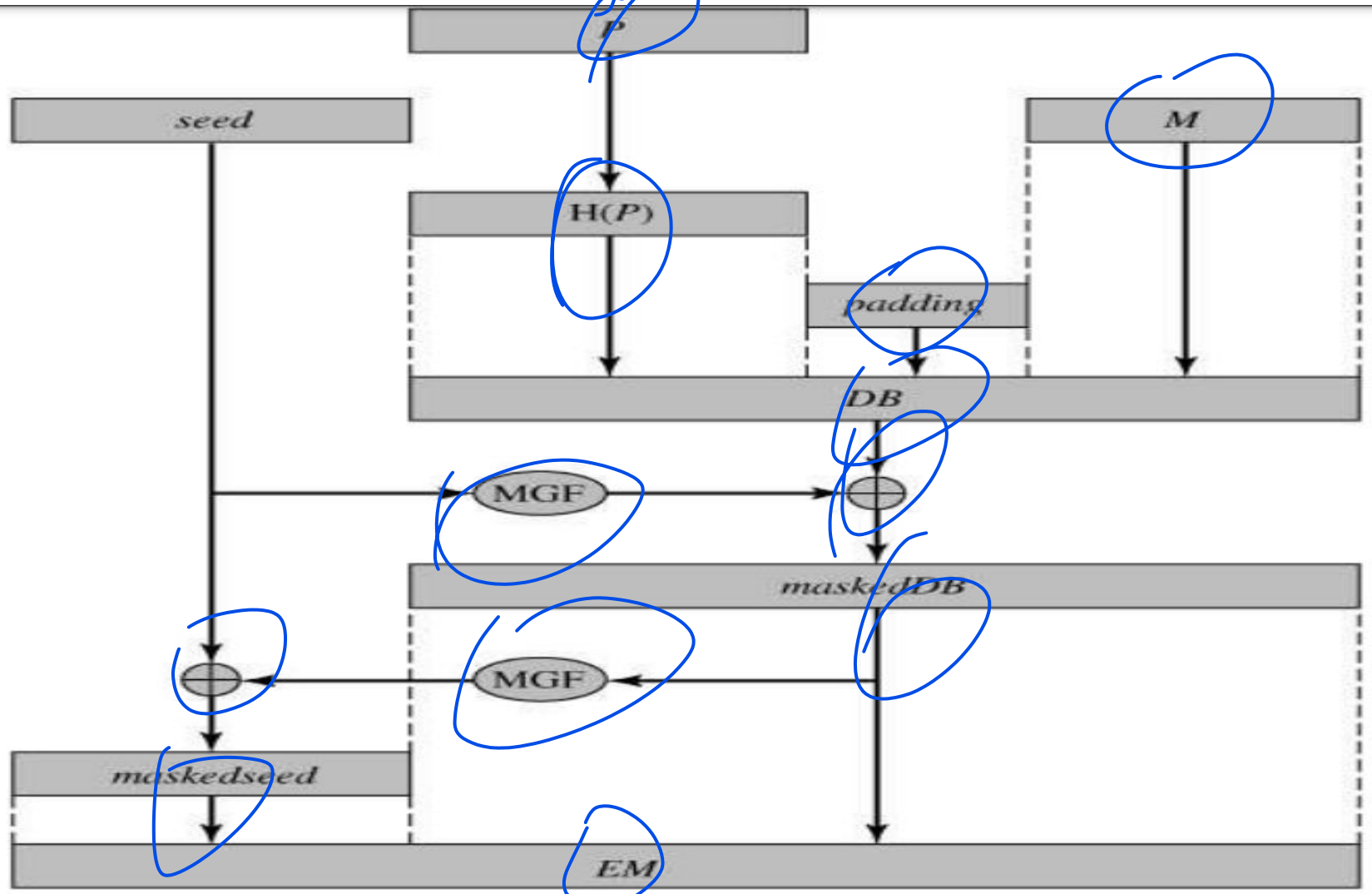


Fig: Encryption Using Optimal Asymmetric Encryption Padding (OAEP)

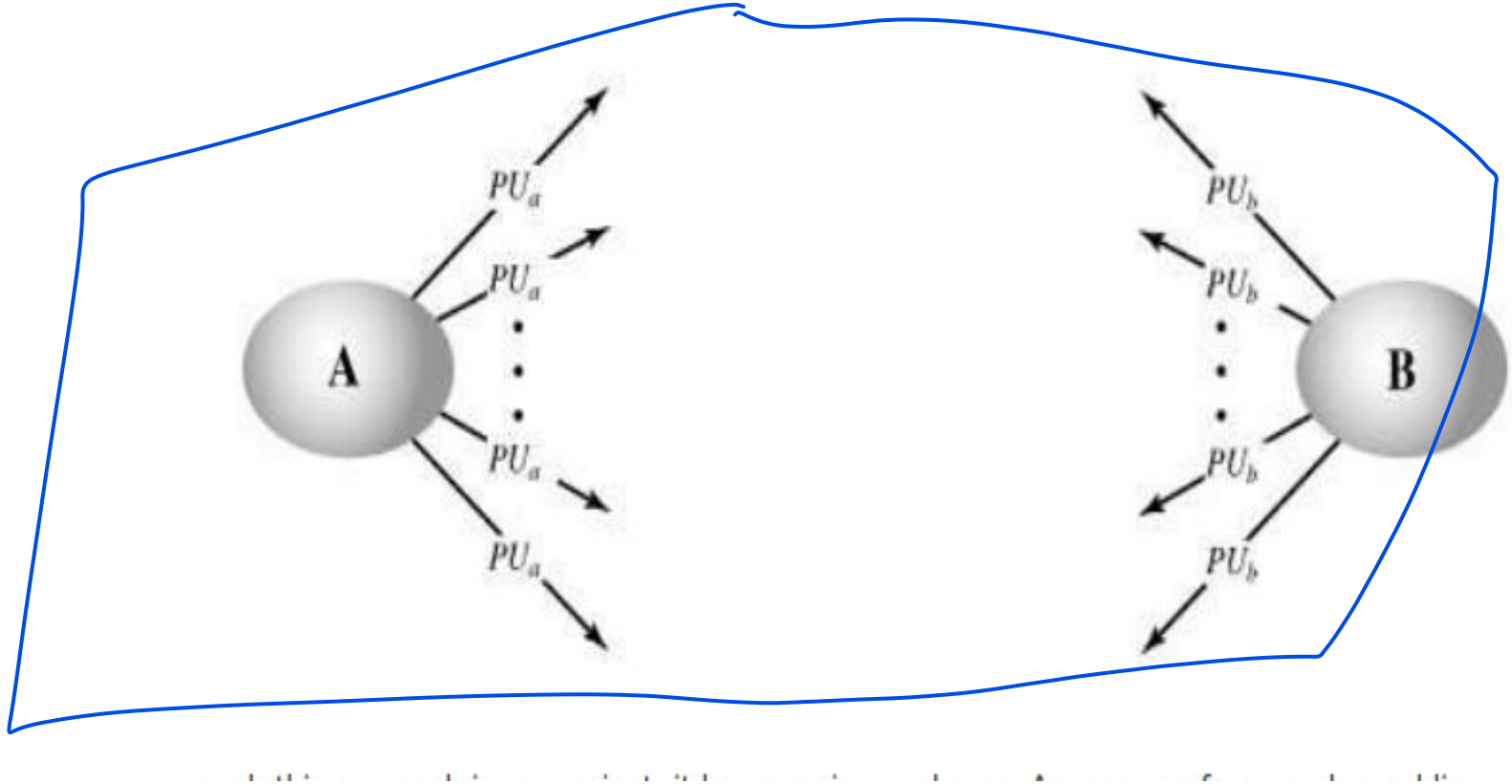
Key Management

1. **The distribution of public keys**
2. **The use of public-key encryption to distribute secret keys**

Distribution of Public Keys Several techniques have been proposed for the distribution of public keys. :

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

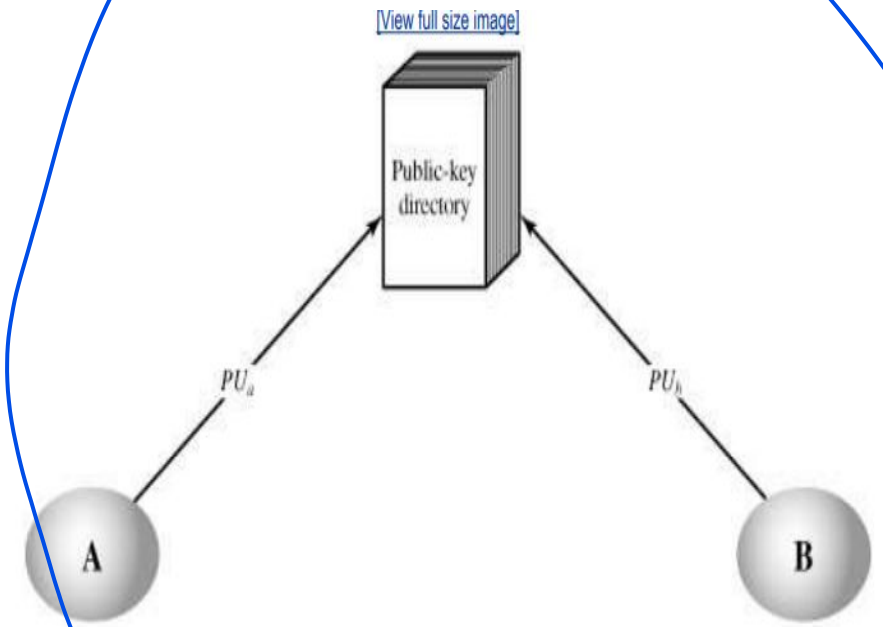
Public announcement



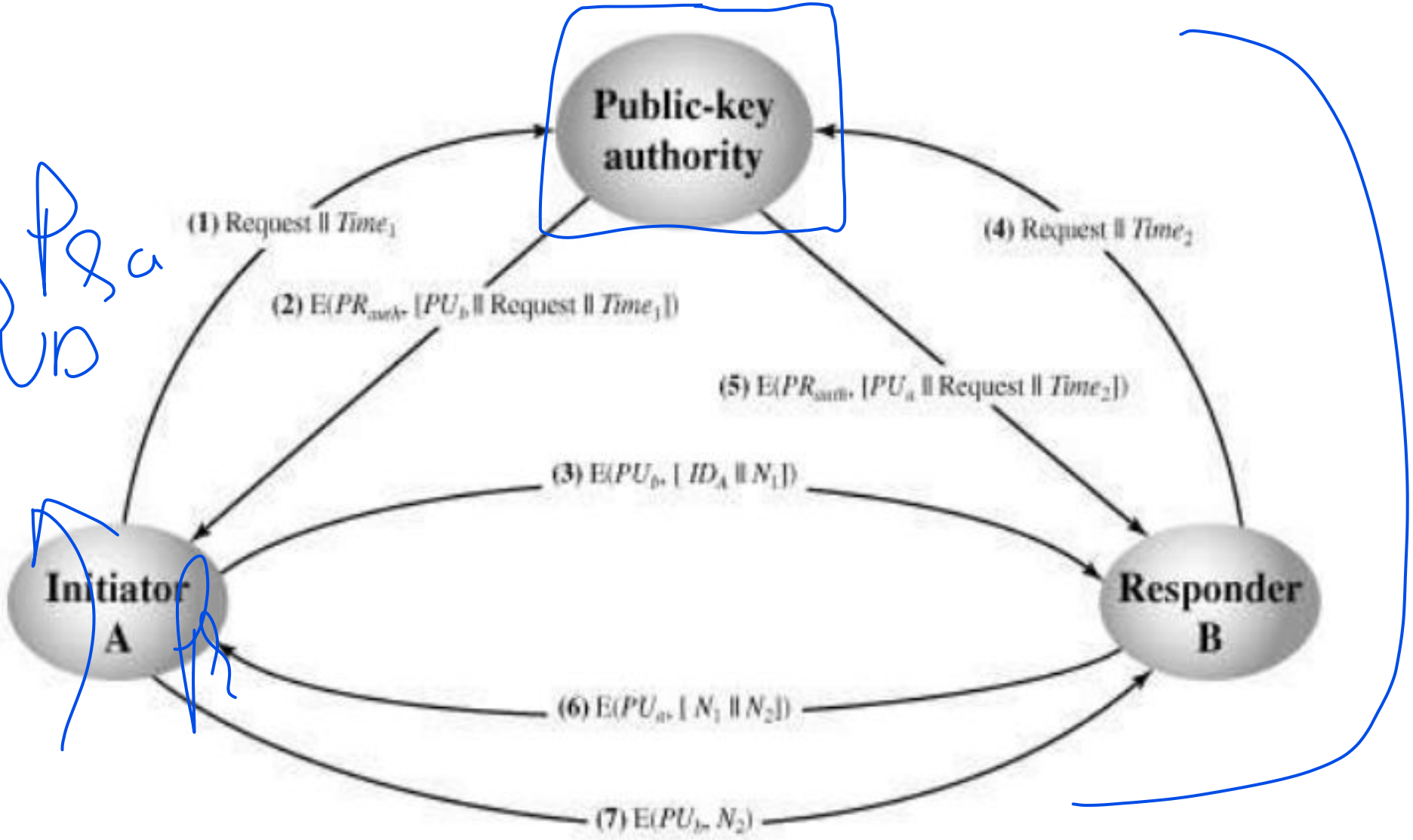
Key Management

Public Available Directory

- The authority maintains a directory with a {name, public key} entry for each participant.
- Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
- A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
- Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory



Public Key Authority



Key Management

Public Key Authority

1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key, *PRauth* Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority.

The message includes the following:

- B's public key, *PUb* which A can use to encrypt messages destined for B
- The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
- The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key

Public Key Authority

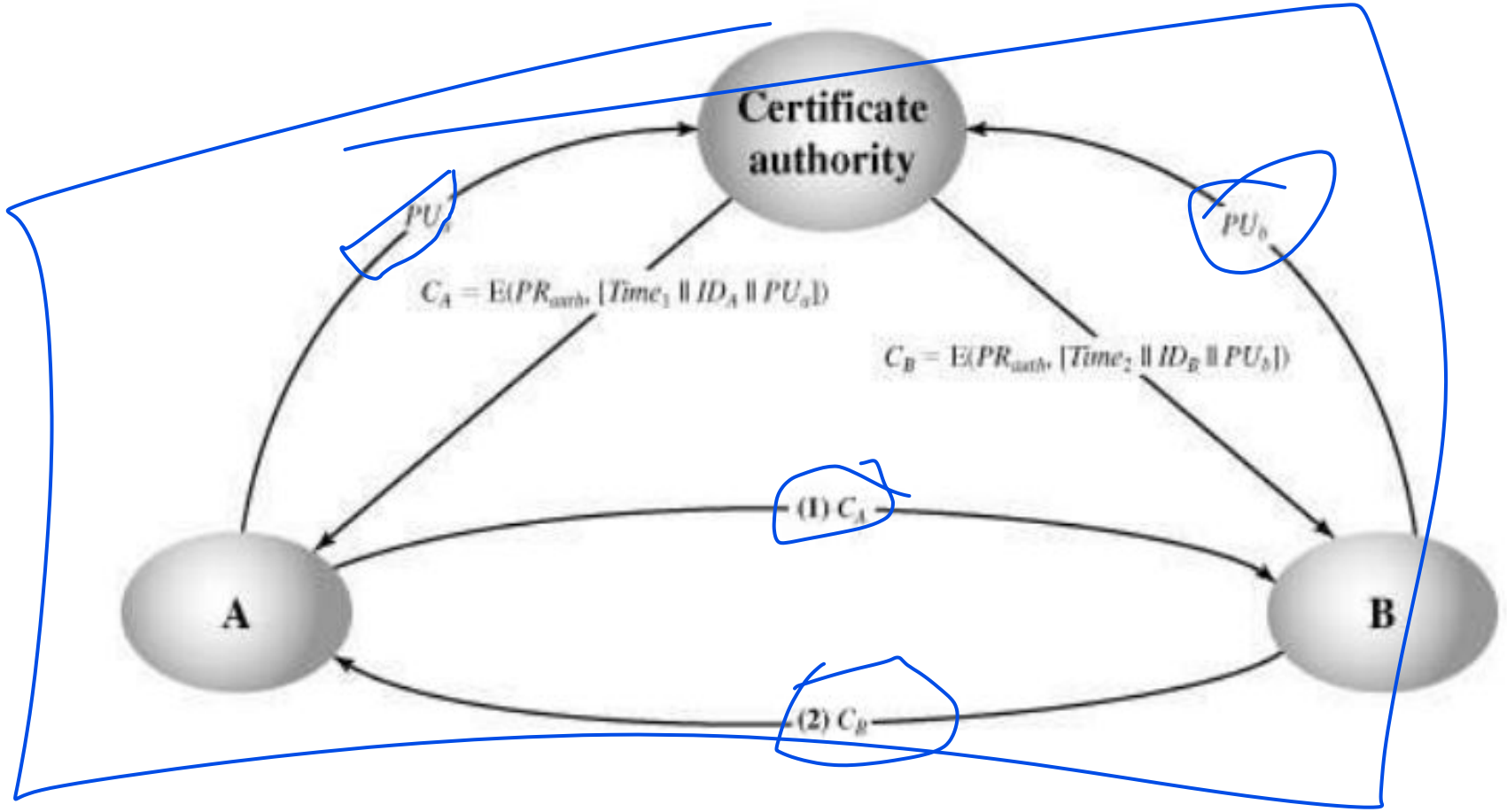
3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (*IDA*) and a nonce (*N*1), which is used to identify this transaction uniquely.

4, 5. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.
At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

6. B sends a message to A encrypted with *PUa* and containing A's nonce (*N*1) as well as a new nonce generated by B (*N*2) Because only B could have decrypted message (3), the presence of *N*1 in message (6) assures A that the correspondent is B.

7. A returns *N*2, encrypted using B's public key, to assure B that its correspondent is A.

Public Key certificates



Key Management

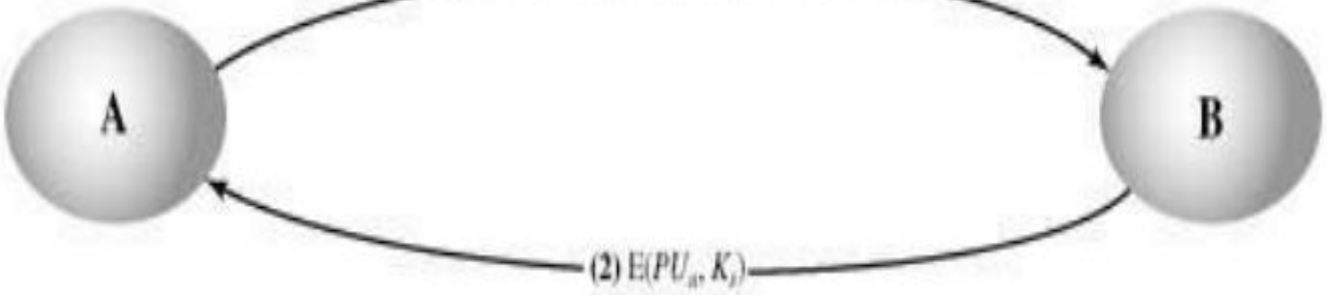
Public Key certificates

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.

Distribution of Secret Keys using public key

Simple Secret Key Distribution

[View full size image]



Distribution of Keys using public key cryptography

Simple Secret Key Distribution

1. A generates a public/private key pair {*PUa*, *PR*a} and transmits a message to B consisting of *PUa* and an identifier of *A*, *IDA*.

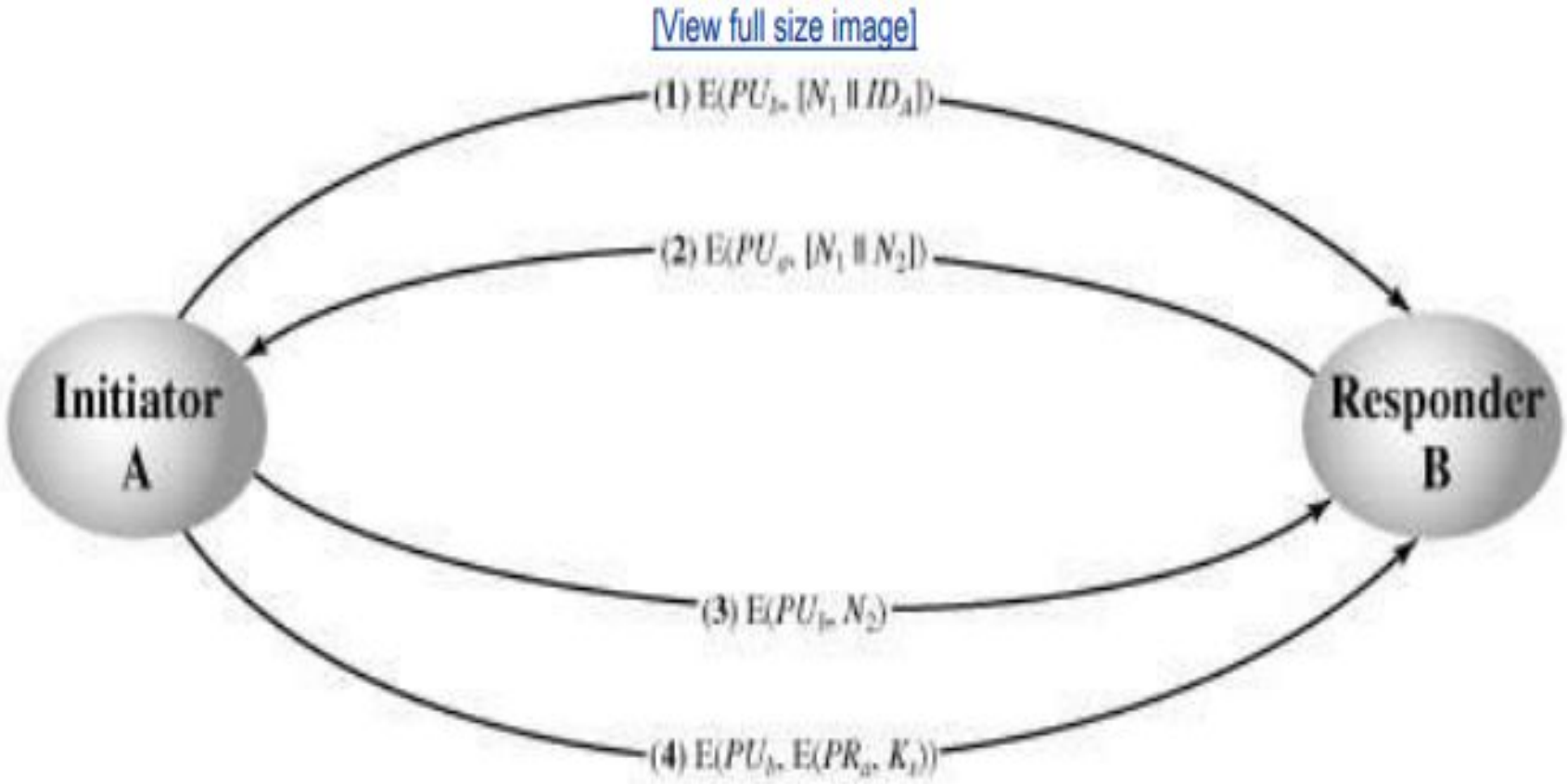
2. B generates a secret key, *Ks*, and transmits it to A, encrypted with A's public key.

3. A computes D(*PRa*, E(*PUa*, *Ks*)) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of *Ks*.

4. A discards *PUa* and *PRa* and B discards *PUa*.

Distribution of Keys using public key cryptography

Secret Key Distribution with Confidentiality and Authentication



Distribution of Keys using public key cryptography

Secret Key Distribution with Confidentiality and Authentication

1. A uses B's public key to encrypt a message to B containing an identifier of A (*IDA*) and a nonce (*N*1), which is used to identify this transaction uniquely.

2. B sends a message to A encrypted with *PUa* and containing A's nonce (*N*1) as well as a new nonce generated by B (*N*2) .Because only B could have decrypted message (1), the presence of *N*1 in message (2) assures A that the correspondent is B.

3. A returns *N*2 encrypted using B's public key, to assure B that its correspondent is A.

4. A selects a secret key *Ks* and sends $M = E(PUb, E(PRa, Ks))$ to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.

5. B computes $D(PUa, D(PRb, M))$ to recover the secret key.

Distribution of Keys using public key cryptography

Hybrid Scheme

This scheme retains the use of a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key.

A public key scheme is used to distribute the master keys. The following rationale is provided for using this three-level approach:

Performance: There are many applications, especially transaction-oriented applications, in which the session keys change frequently. Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption. With a three-level hierarchy, public-key encryption is used only occasionally to update the master key between a user and the KDC.

Backward compatibility: The hybrid scheme is easily overlaid on an existing KDC scheme, with minimal disruption or software changes.

Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid.

Message Authentication Requirements:
In the context of communications across a network, following attacks can be identified.

1. Disclosure
2. Traffic Analysis
3. Masquerade
4. Content Modification
5. Sequence Modification
6. Timing Modification
7. Source Nonrepudiation
8. Destination Nonrepudiation

Message Authentication Requirements

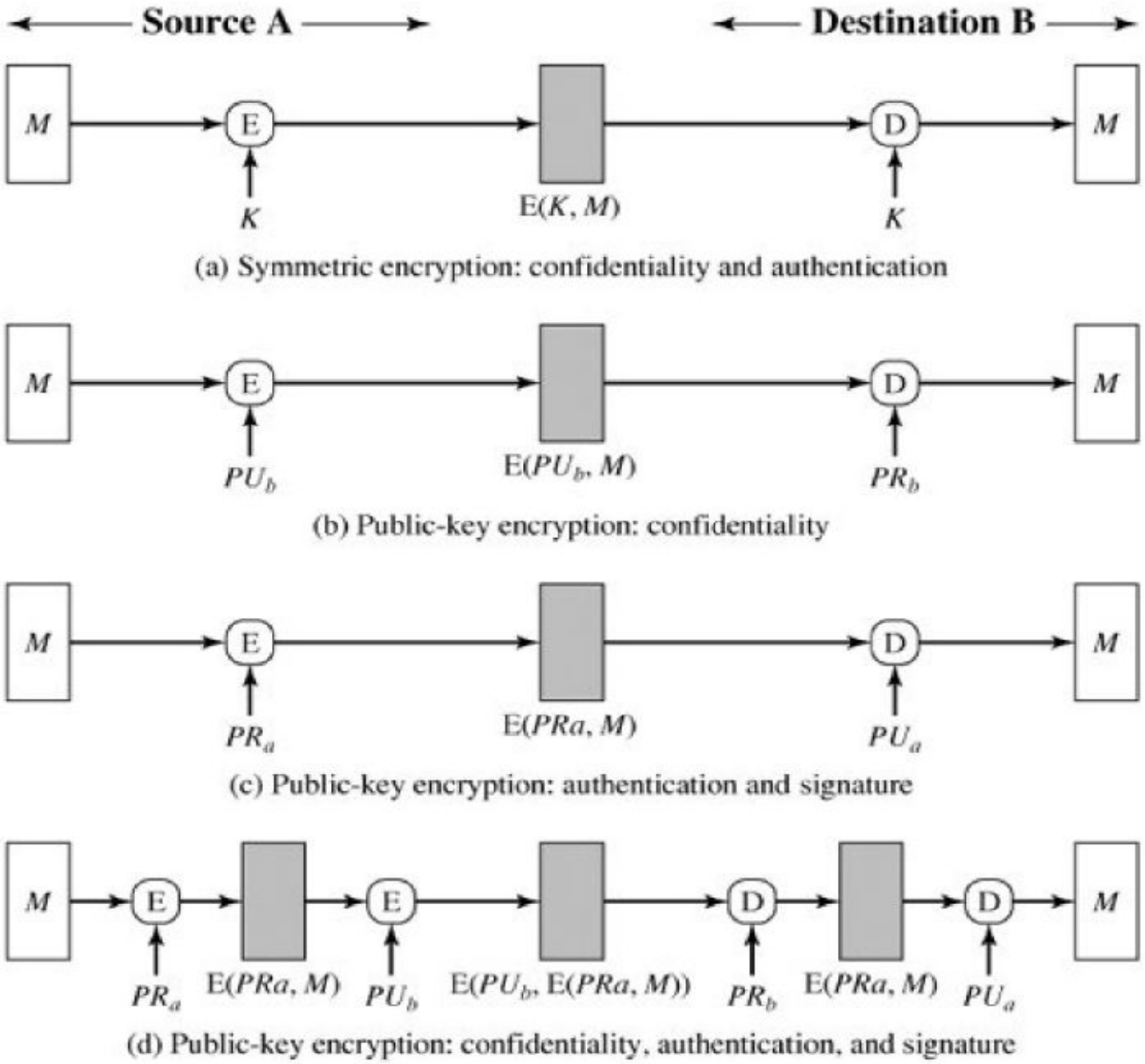
1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.
4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
7. **Source repudiation:** Denial of transmission of message by source.
8. **Destination repudiation:** Denial of receipt of message by destination.

Authentication Function

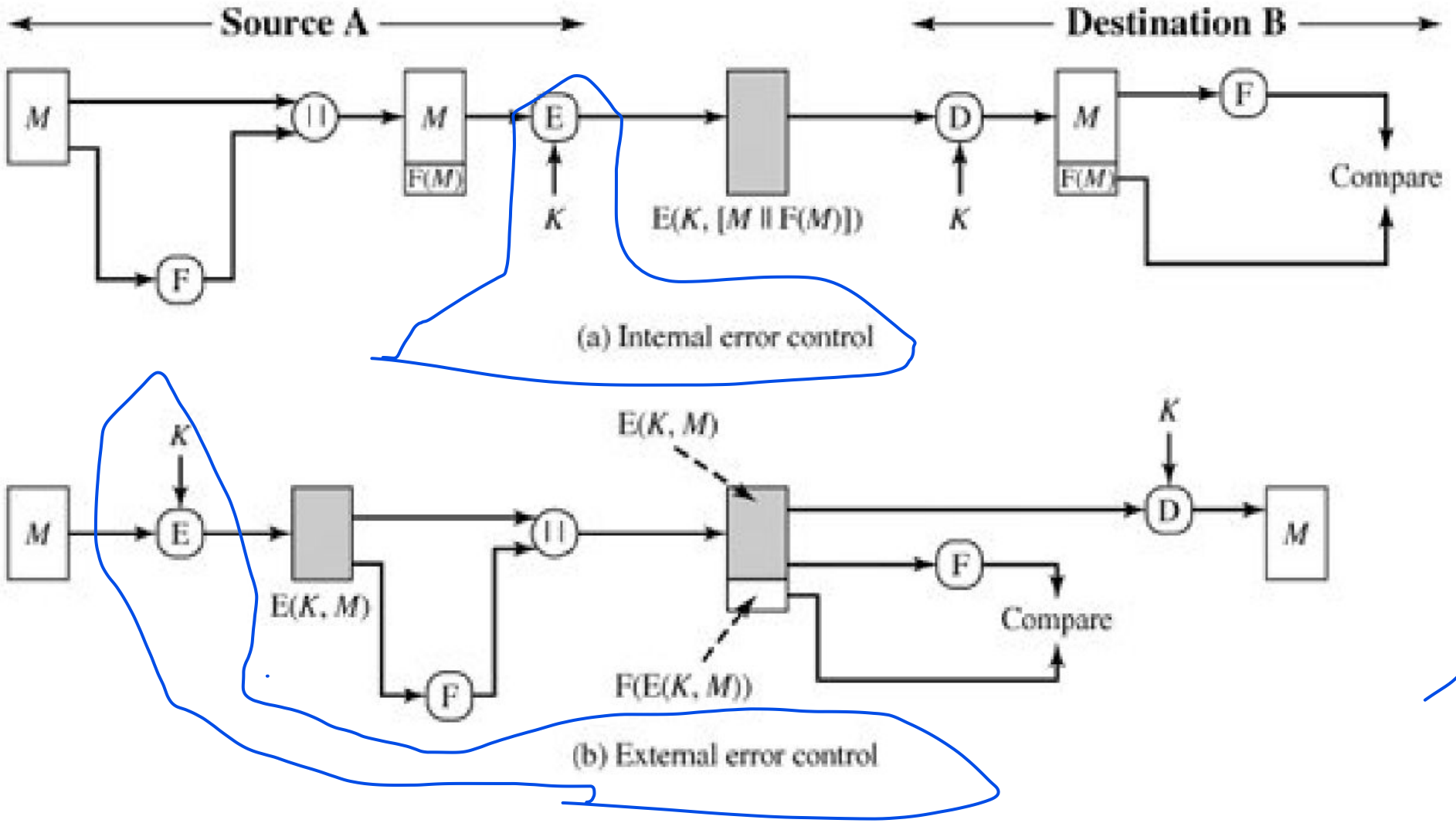
- **Message Encryption**: The ciphertext of the entire text serves as its authenticator.
- **Message Authentication Code**:- A function of the message and a secret key that produces a fixed length value that serves as the authenticator.
- **Hash Functions**:- A function that maps a message of any length into a fixed hash value, which serves as authenticator.

Message Encryption

Fig: Basic Uses of Message Encryption



Internal and External Error Control

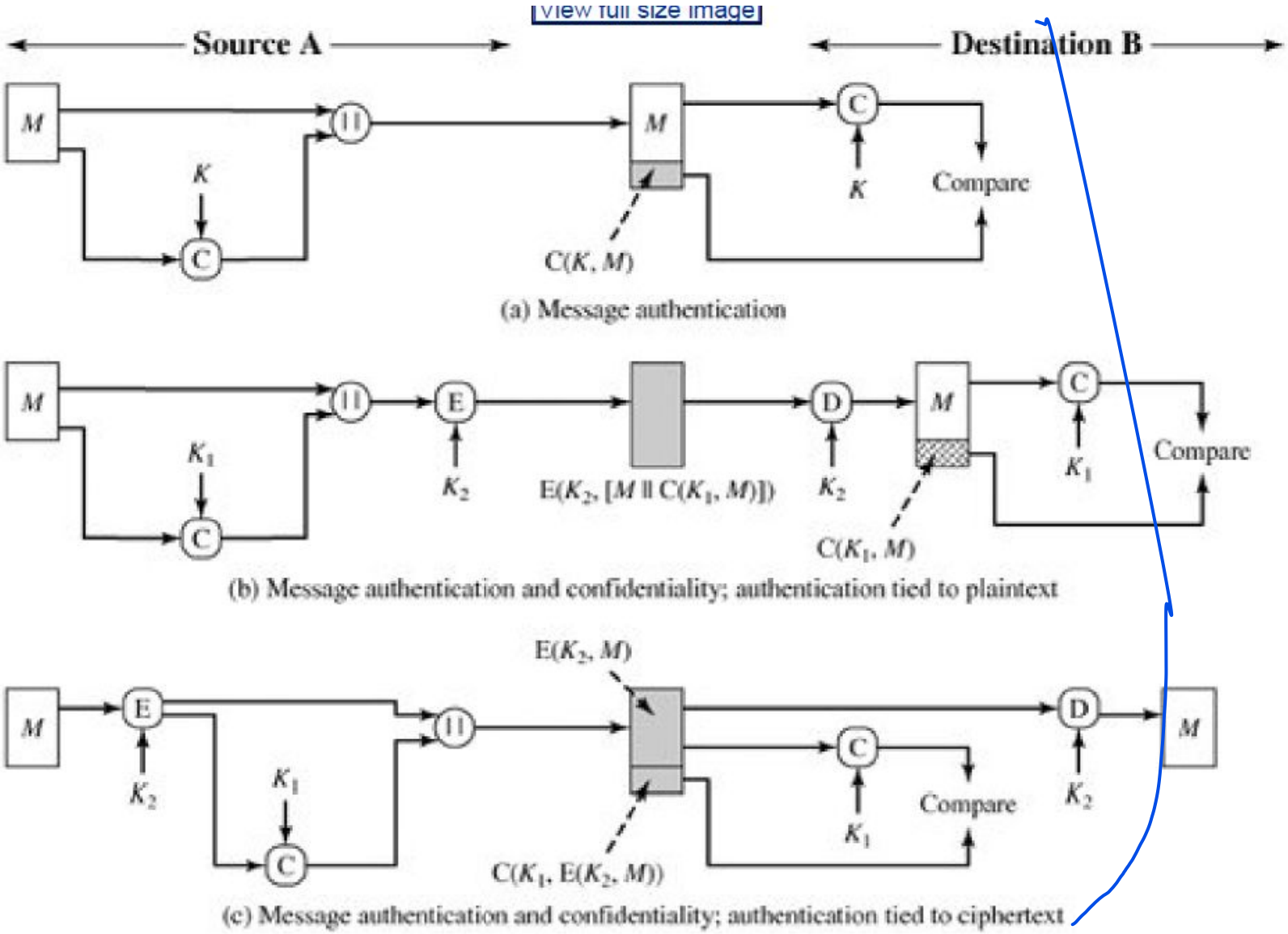


Message Authentication Code

Message Authentication code involves the use of a secret key to generate a small fixed block of data known as **Cryptographic Checksum or MAC.**

MAC=C(K,M)

Where M->Input Message
C= MAC function
K=Shared Secret Key
MAC= Message authentication code.



Message Authentication Code

- The Message plus MAC are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key to generate new MAC.
- The received MAC is compared to the calculated MAC.

If we assume that only receiver and sender know the identity of the secret key, if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with proper MAC.

3. If the message includes sequence number, then the receiver can be assured of the proper sequence.

Message Authentication Code- Usage

1. There are a number of applications in which the same message is broadcast to a number of destinations. Examples are notification to users that the network is now unavailable or an alarm signal in a military control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated message authentication code. The responsible system has the secret key and performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.

2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, messages being chosen at random for checking.

3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication code were attached to the program, it could be checked whenever assurance was required of the integrity of the program.

Message Authentication Code- Usage

4. For some applications, it may not be of concern to keep messages secret, but it is important to authenticate messages. An example is the Simple Network Management Protocol Version 3 (SNMPv3), which separates the functions of confidentiality and authentication. For this application, it is usually important for a managed system to authenticate incoming SNMP messages, particularly if the message contains a command to change parameters at the managed system. On the other hand, it may not be necessary to conceal the SNMP traffic.

5. Separation of authentication and confidentiality functions affords architectural flexibility. For example, it may be desired to perform authentication at the application level but to provide confidentiality at a lower level, such as the transport layer.

6. A user may wish to prolong the period of protection beyond the time of reception and yet allow processing of message contents. With message encryption, the protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.

Message Authentication Code Requirements

- **Round 1**

  Given: M_1, $MAC_1 = C(K, M_1)$

  Compute $MAC_i = C(K_i, M_1)$ for all 2^k keys

  Number of matches $\approx 2^{(k-n)}$

- **Round 2**

  Given: M_2, $MAC_2 = C(K, M_2)$

  Compute $MAC_i = C(K_i, M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1

  Number of matches $\approx 2^{(k-2xn)}$

Message Authentication Code Requirements

1. If an opponent observes M and $C(K, M)$, it should be computationally infeasible for the opponent to construct a message M' such that $C(K, M') = C(K, M)$.
2. $C(K, M)$ should be uniformly distributed in the sense that for randomly chosen messages M and M', the probability that $C(K, M) = C(K, M')$ is 2^n, where n is the number of bits in the MAC.
3. Let M' be equal to some known transformation on M. That is, $M' = f(M)$. For example, f may involve inverting one or more specific bits. In that case, $Pr[C(K, M) = C(K, M')] = 2^n$.

Hash Functions

A hash value *h* is generated by a function H of the form

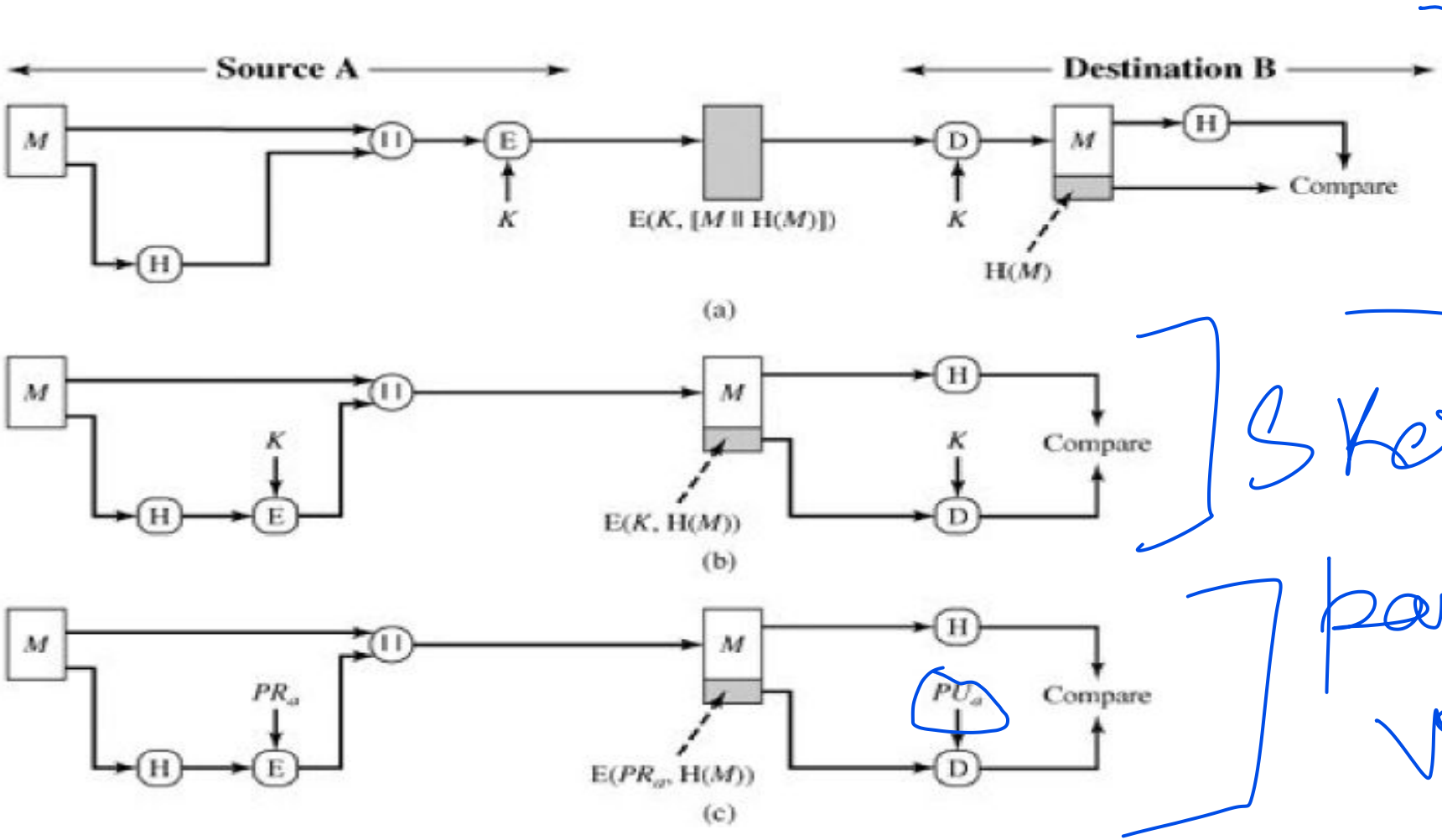
$h = H(M)$

- where *M* is a variable-length message and H(*M*) is the fixed-length hash value.
- The hash value is appended to the message at the source at a time when the message is assumed or known to be correct.
- The receiver authenticates that message by recomputing the hash value.

Hash Functions

- A variation on the message authentication code is the one-way hash function.
- As with the message authentication code, a hash function accepts a variable-size message *M* as input and produces a fixed-size output, referred to as a **hash code** H(*M*).
- Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a **message digest** or **hash value**.
- The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.

Uses/Applications of Hash Functions

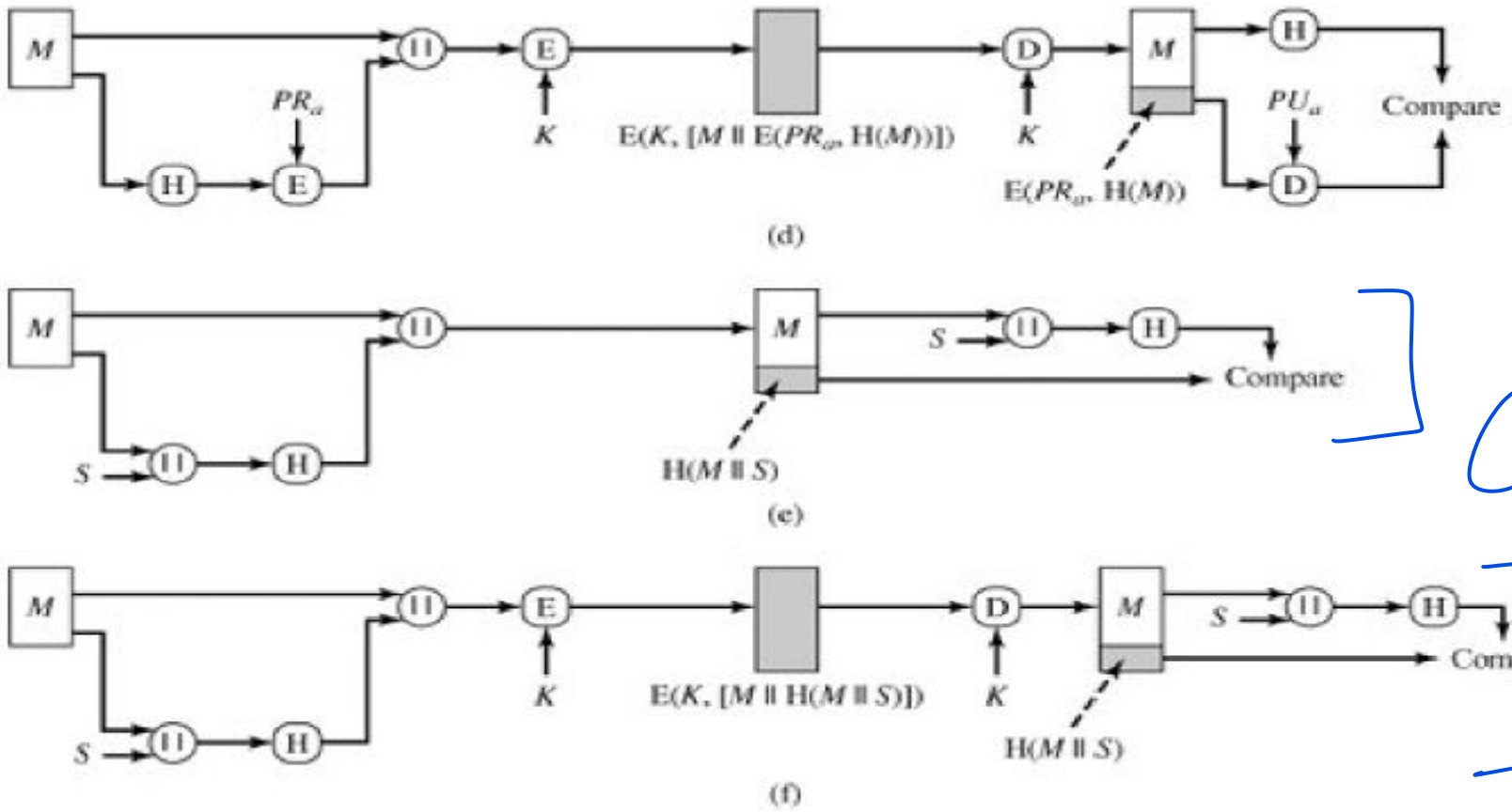


HMAC

Sym Key Conf

pass verif

Uses/Applications of Hash Functions



Passwords + Key Config

Challenge Response + Enhanced Version

Security of Hash Functions and Macs

Two categories of hacks are possible

1. **Brute Force Attacks**
2. **Cryptanalysis**

Security of Hash Functions and Macs

Hash function has the 3 desirable properties

- **One-way:** For any given code h, it is computationally infeasible to find x such that $H(x) = h$.
- **Weak collision resistance:** For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
- **Strong collision resistance:** It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

For a hash code of length n, the level of effort required, as we have seen is proportional to the following:

One way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^{n/2}$

Message Authentication Codes

To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows:

- **Computation resistance:** Given one or more text-MAC pairs $[x_i, C(K, x_i)]$, it is computationally infeasible to compute any text-MAC pair $[x, C(K, x)]$ for any new input $x \neq x_i$.

Digital Signature

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other.

Several forms of dispute between the two are possible.

- Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.
- John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the **digital signature**

Digital Signature Properties

- It must verify the author and the date and time of the signature.
- It must to authenticate the contents at the time of the signature.
- It must be verifiable by third parties, to resolve disputes.

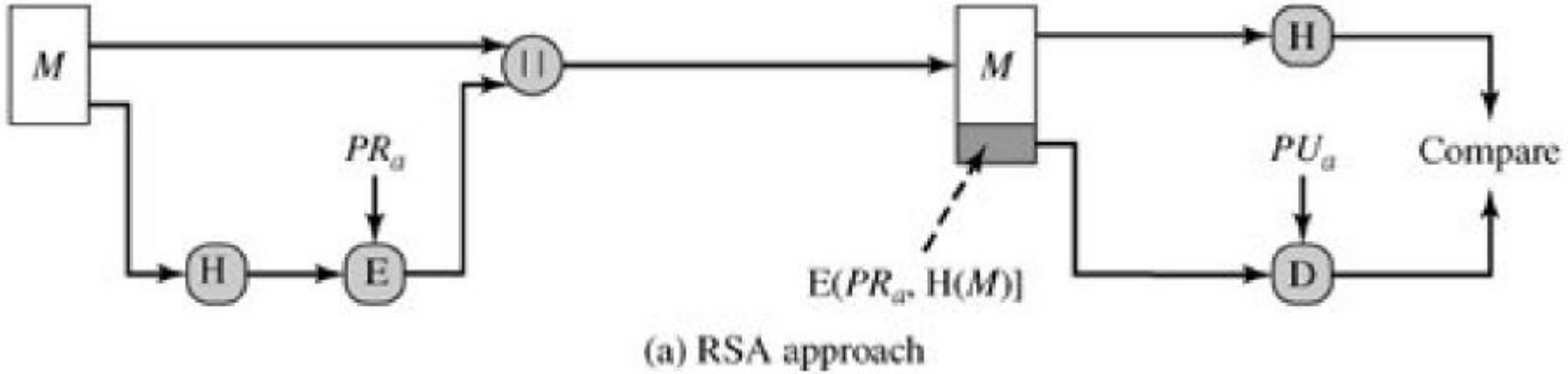
Digital Signature Requirements

1. The signature must be a bit pattern that depends on the message being signed.
2. The signature must use some information unique to the sender, to prevent both forgery and denial.
3. It must be relatively easy to produce the digital signature.
4. It must be relatively easy to recognize and verify the digital signature.
5. It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
6. It must be practical to retain a copy of the digital signature in storage.

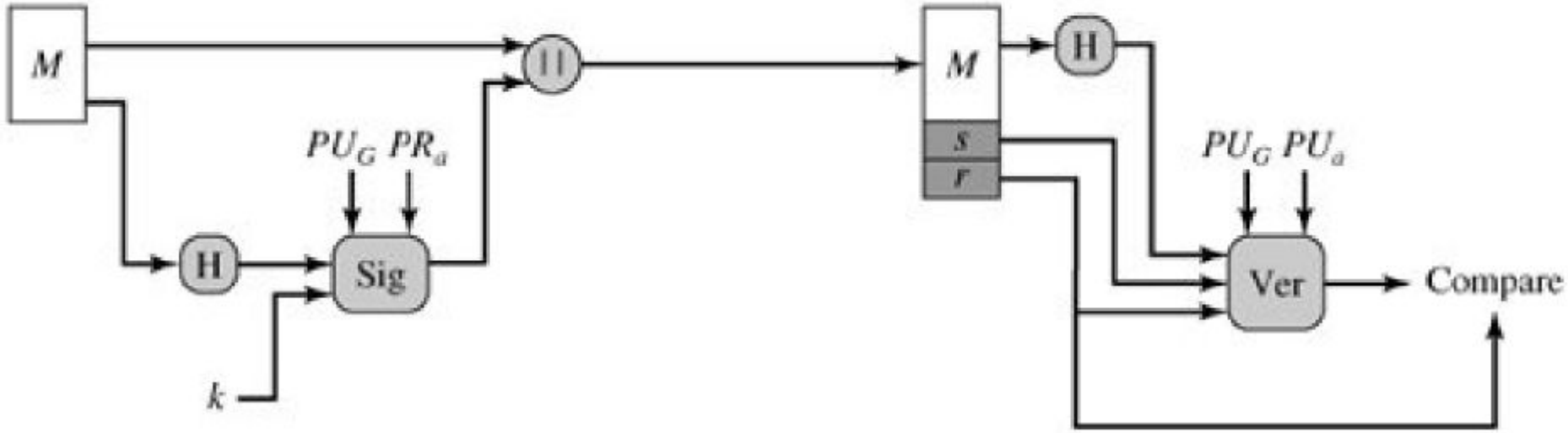
Direct Digital Signature

- The direct digital signature involves only the communicating parties (source, destination).
- It is assumed that the destination knows the public key of the source. A digital signature may be formed by encrypting the entire message with the sender's private key.
- Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key. Here Signature function is put first and then outer layer is confidentiality.
- If the signature is calculated on an encrypted message, then third party also needs access to the decryption key to read the original message.
- The validity of the scheme depends on the security of the sender’s private key.
- If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature.

DSS Approach



(a) RSA approach



(b) DSS approach

Digital Signature Algorithm

1. The DSA (Digital Signature Algorithm) approach involves using of a hash function to create a hash code, same as RSA.
2. This hash code is combined with a randomly generated number k as an input to a signature function. The signature function depends on the sender's private key (PRa) as well as a set of parameters that are known to a group of communicating principals.
3. This set can be considered as a global public key (PUG). The output of the signature function is a signature with two components, s and r. When an incoming message is received, a hash code is generated for the message.
4. This hash code is then combined with the signature and input into a verification function. The verification function depends on the global public key as well as the sender's public key (PUa) which is paired with the sender's private key.
5. The output of the verification function returns a value equal to the signature's component r, if the signature is valid. The signature function is designed in such a way that only the sender, with knowledge of the private key, can produce a valid signature.

Digital SignatureAlgorithm

Components:-

- M = Message or Plaintext
- H = Hash Function
- || = bundle the plaintext and hash function
- E = Encryption Algorithm
- D = Decryption Algorithm
- PUa = Public key of sender
- PRa = Private key of sender
- Sig = Signature function
- Ver = Verification function
- PUG = Global public Key

- **Signing (Sig)**: Signing involves creating a digital signature with the help of a user's private key. In case of DSA, this process requires mathematical operations to be performed on the message that should be signed using a given private key in order to generate a unique signature for that message.
- **Verifying (Ver):** Verifying is the process of verifying whether or not a digital signature has been forged using its corresponding public key. In DSA, this involves comparing the messages hash against the verification value through mathematical operations between two binary strings – one representing an encrypted data and another one representing plain-text original

Steps to perform Digital Signature Algorithm

Global Public Components:-

There are 3 components that are public and can be shared to a set of users.

- A prime number **p** is chosen with a length between 512 and 1024 bits such that q divides (p – 1). So, p is prime number where $2^{L-1} < p < 2^{L}$ for 512<= L<=1024 and L is a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits.
- Next, an N-bit prime number **q** is selected. So, q is prime divisor of (p – 1), where $2^{N-1} < q < 2^{N}$ i.e., bit length of N bits.
- Finally, g is selected to be of the form h(p-1)/q mod p, where h is an integer between 1 and (p – 1) with the limitation that g must be greater than 1. So, g is = h(p – 1)/q mod p, where h is any integer with $1 < h < (p – 1)$ such that $h^{(p-1)/q} \bmod p > 1$.

Steps to perform Digital Signature Algorithm

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

k = random or pseudorandom integer with $0 < k < q$

Signing

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

$s = [k^{-1} (H(M) + xr)] \bmod q$

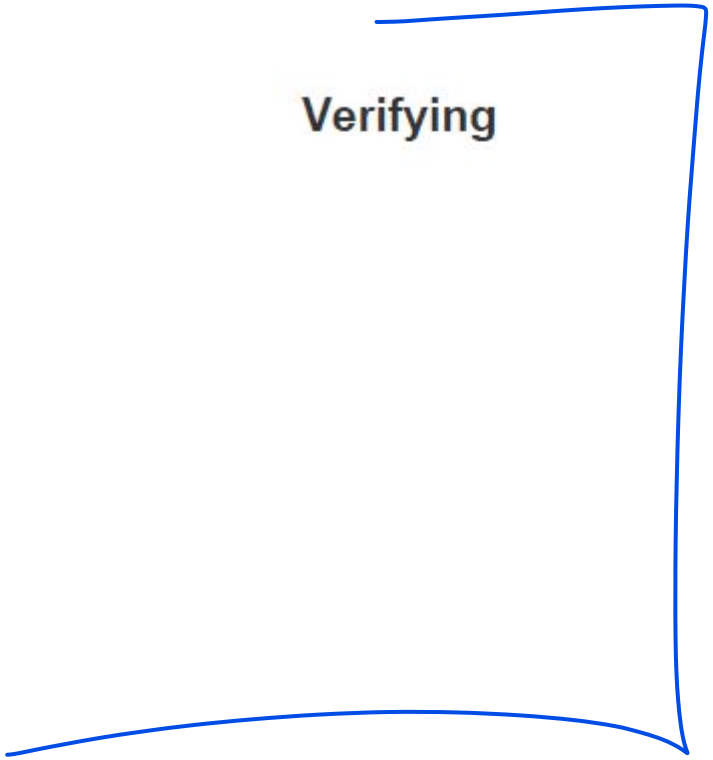
Steps to perform Digital Signature Algorithm

$w = (s')^{-1} \bmod q$

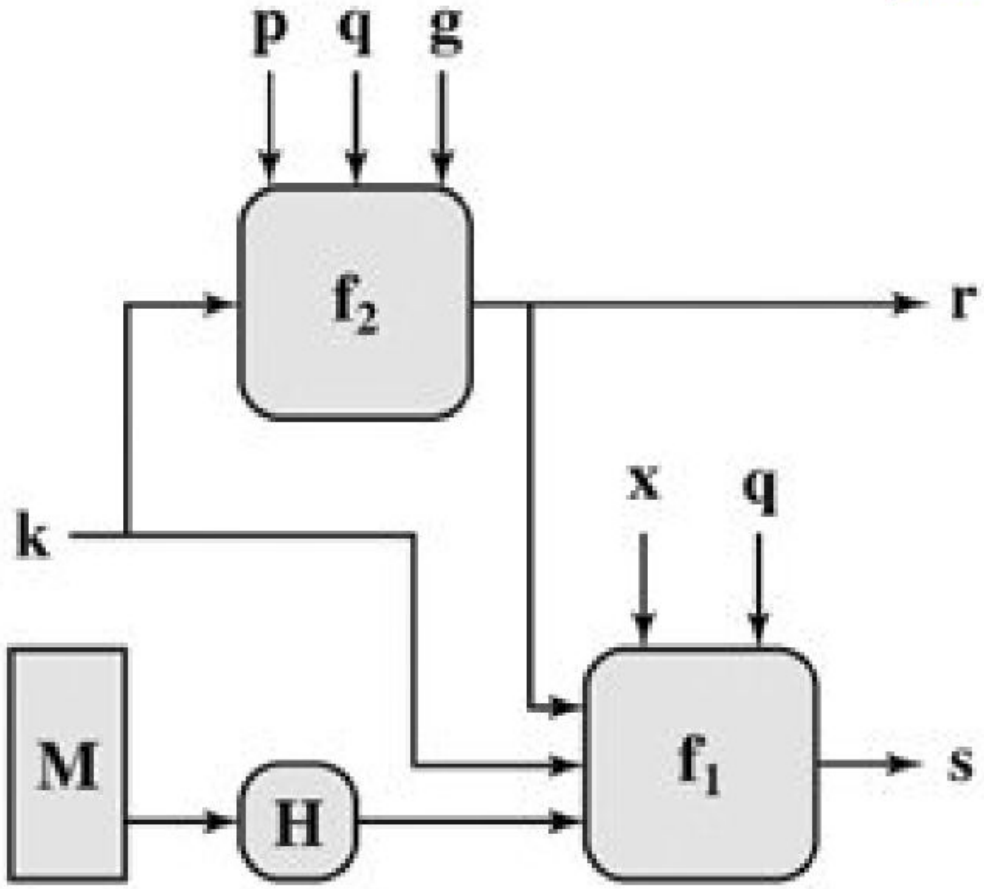
$u1 = [\mathbf{H(M')}w] \bmod q$

$u2 = (r')w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$



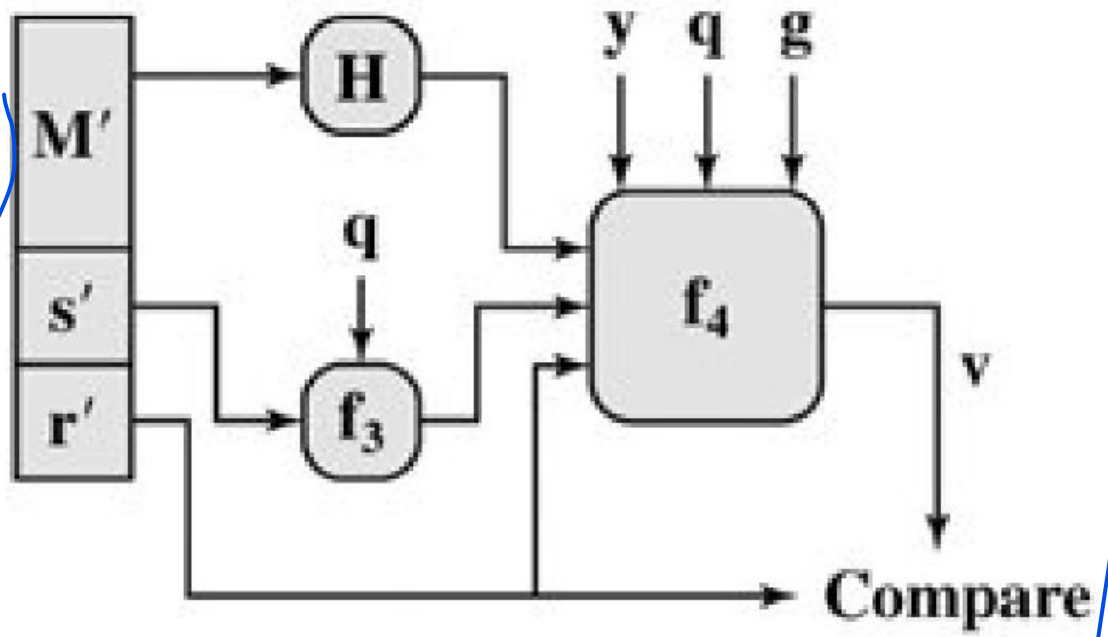
[View full size image]



$s = f_1(H(M), k, x, r, q) = (k^{-1}(H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing



$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying