Chapter 9

Public Key Cryptography
Hemant Ghayvat

Table 9.1 Terminology Related to Asymmetric Encryption

Asymmetric Keys
Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

Public Key Certificate
A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

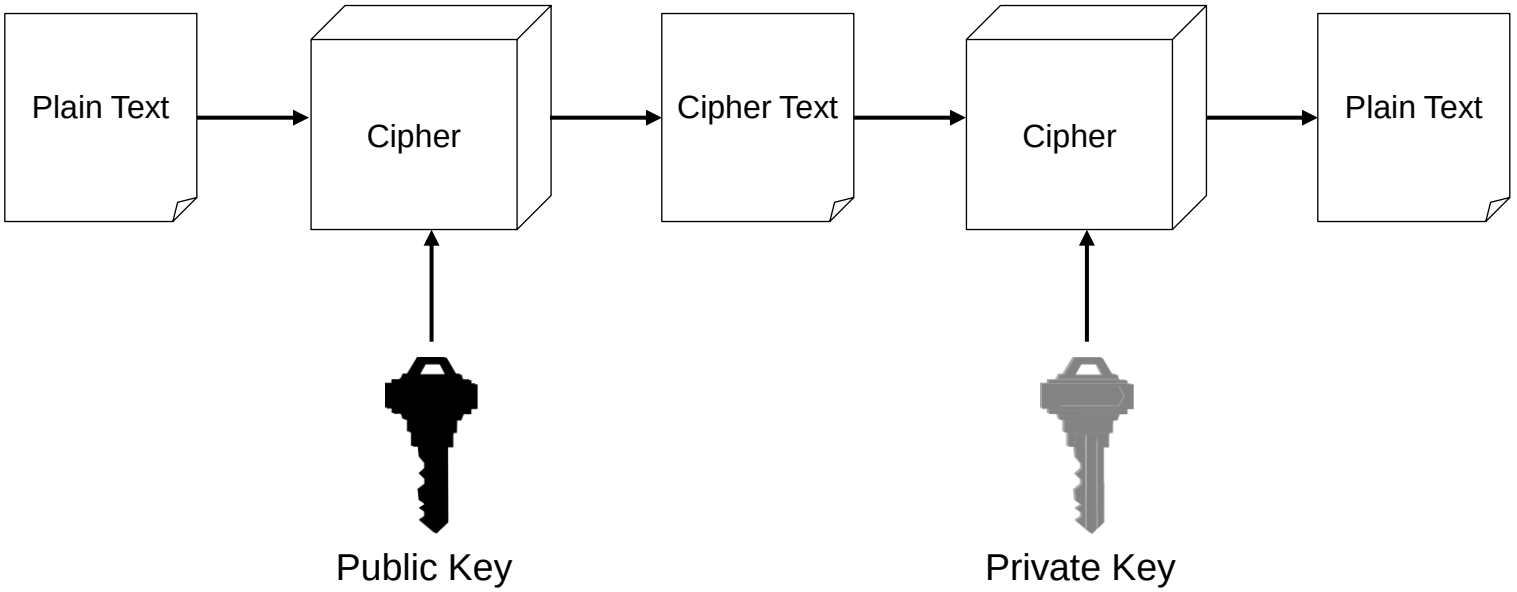
Public Key (Asymmetric) Cryptographic Algorithm
A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Public Key Infrastructure (PKI)
A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

Source: *Glossary of Key Information Security Terms*, NISTIR 7298.

Asymmetric Encryption

- Uses a pair of keys for encryption
  - Public key for encryption
  - Private key for decryption
- Messages encoded using public key can only be decoded by the private key
  - Secret transmission of key for decryption is not required
  - Every entity can generate a key pair and release its public key



- **Symmetric vs Asymmetric encryption**
- Symmetric type of encryption is typically used to secure data that is being transmitted over a network or stored on a device. Symmetric encryption is generally faster and more efficient than asymmetric encryption and is often used for large amounts of data.
- Asymmetric type of encryption is typically used for secure communication between two parties.
- Application of symmetric encryption include:

Encrypting files and folders on a computer

Protecting data stored in a database

Securing network traffic, such as email or web traffic

Securing mobile devices, such as smartphones and tablets

- Application of asymmetric encryption include:

Securing online transactions, such as online banking and shopping

Protecting sensitive information, such as passwords and credit card numbers

Providing secure communication between users on a network

Authenticating users in a secure manner, such as with digital signatures or certificates.

Public-Key Cryptosystems

- A public-key encryption scheme has six ingredients:
- Plaintext
  - **The readable message or data that is fed into the algorithm as input**
- Encryption algorithm
  - **Performs various transforma-tions on the plaintext**
- Public key
  - **Used for encryption or decryption**
- Private key
  - **Used for encryption or decryption**
- Ciphertext
  - **The scrambled message produced as output**
- Decryption algorithm
  - **Accepts the ciphertext and the matching key and produces the original plaintext**

Figure 9.1 Public-Key Cryptography
(1 of 2)

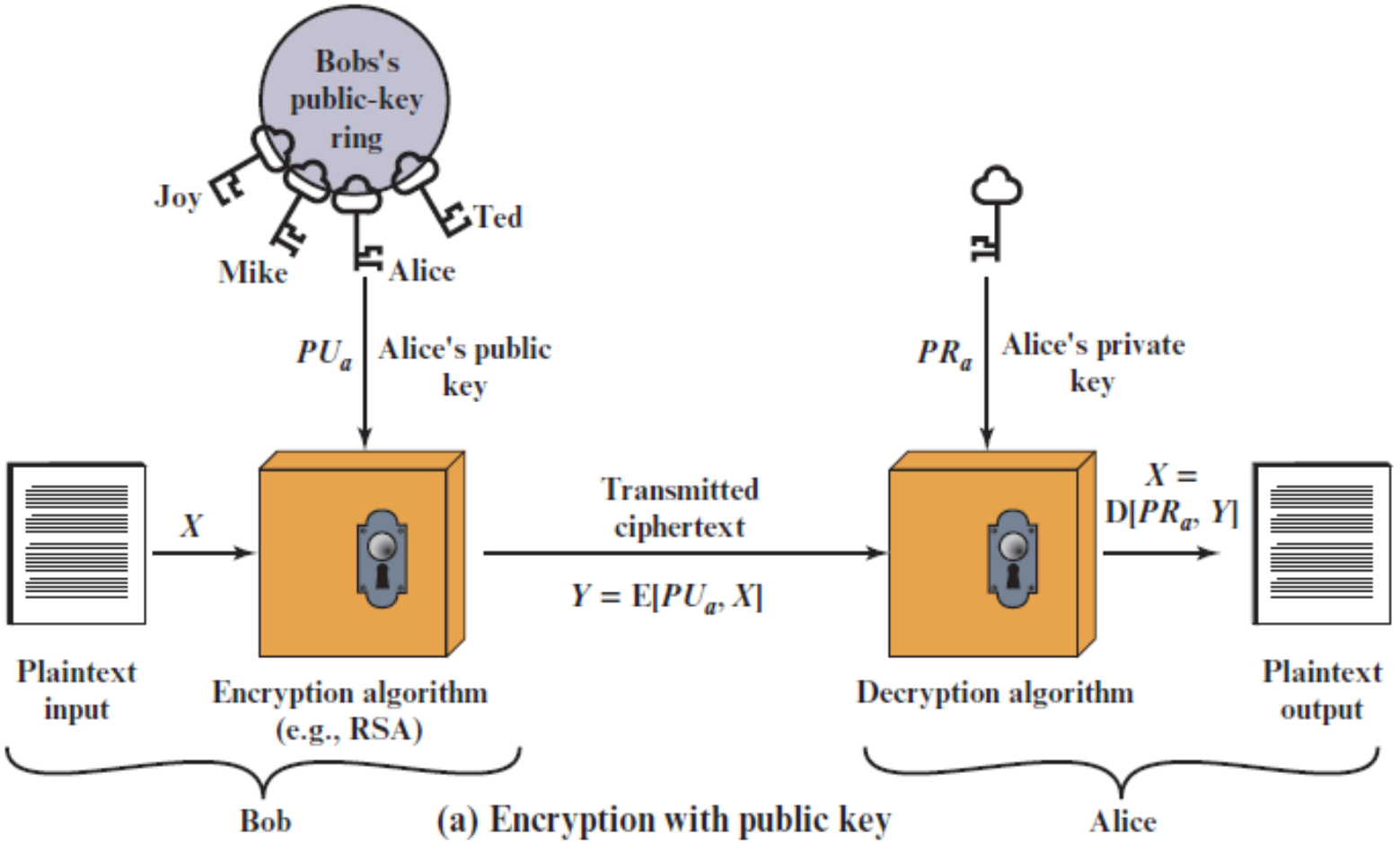


Figure 9.1 Public-Key Cryptography (2 of 2)

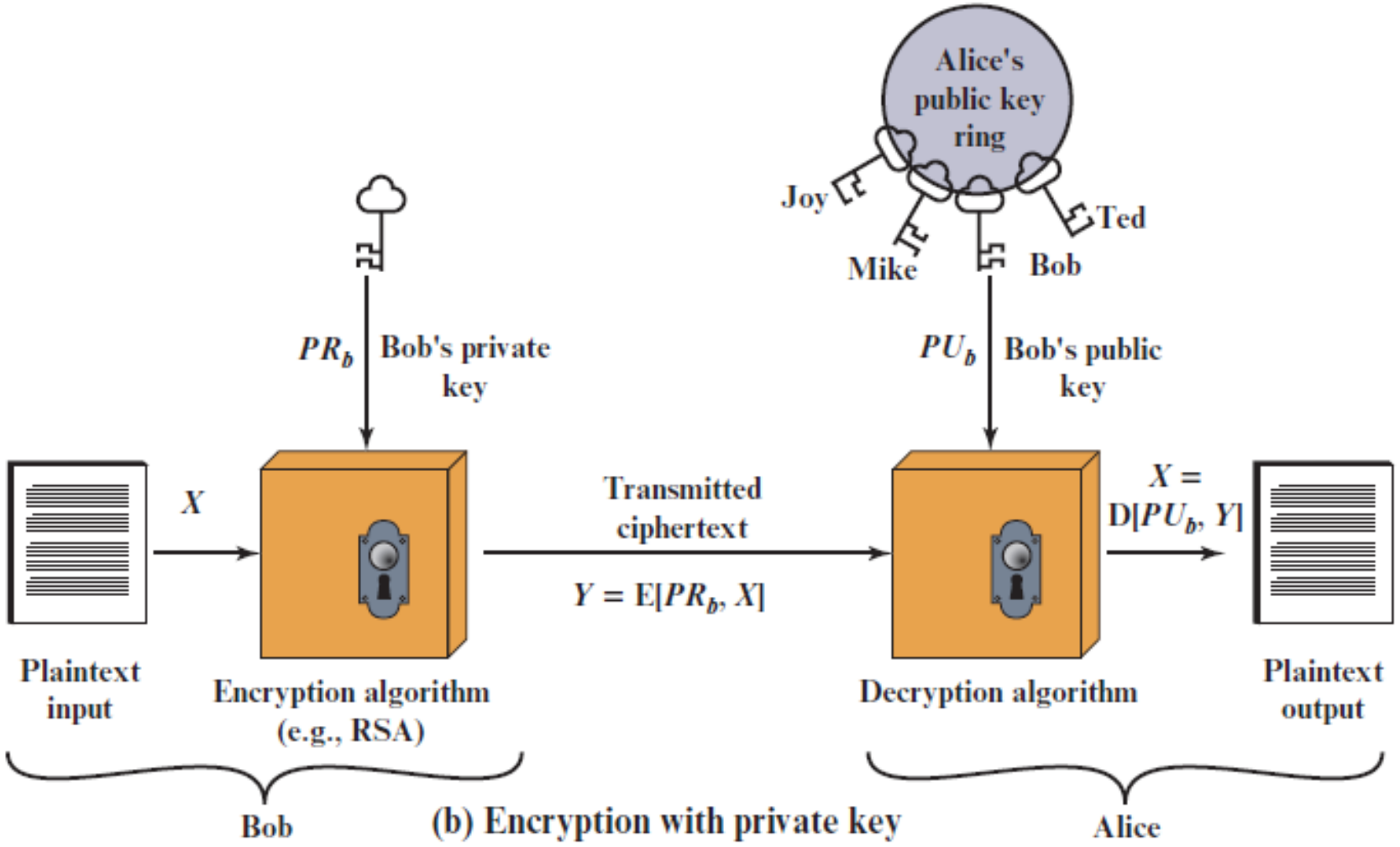
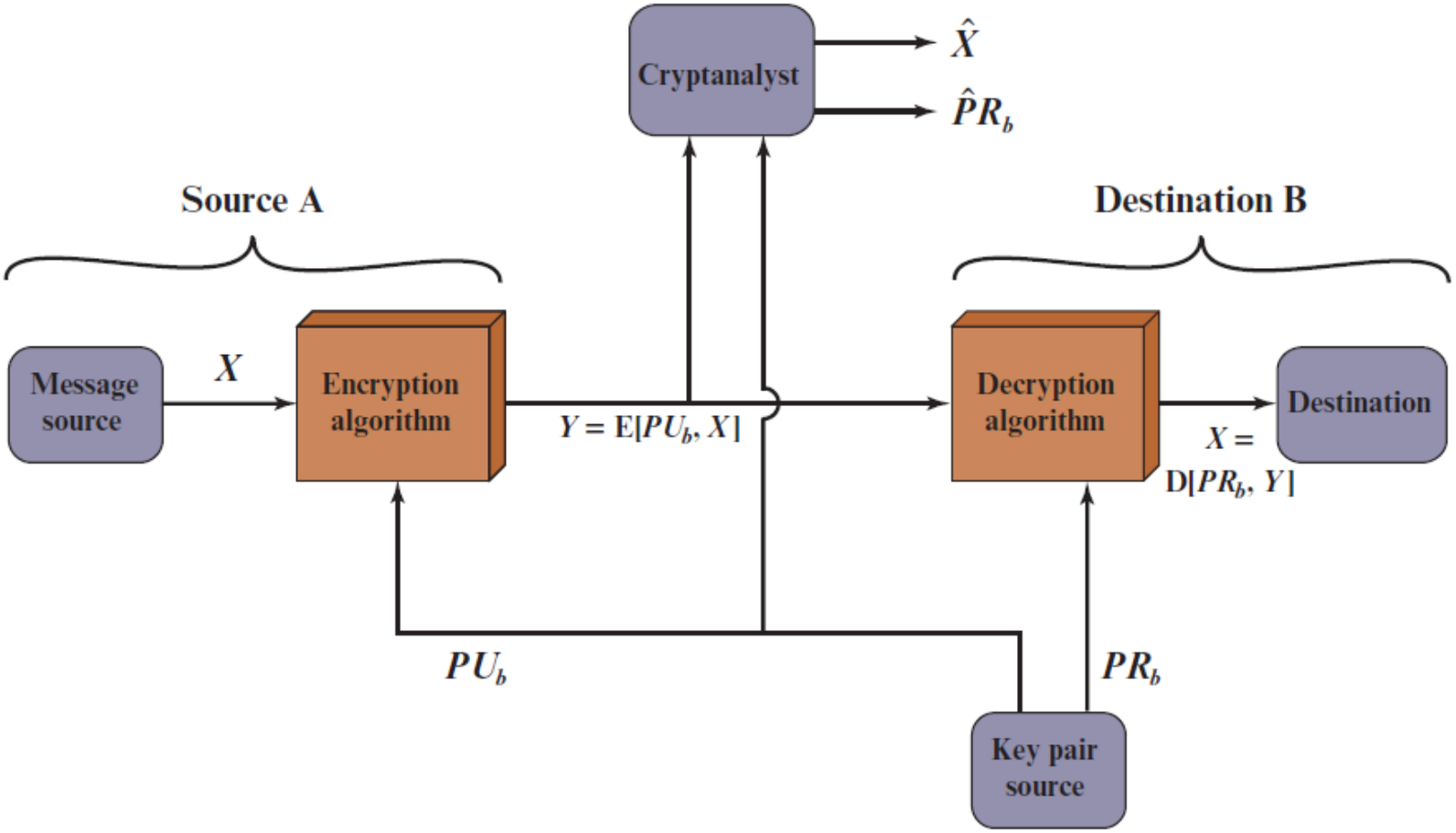


Figure 9.1 Public-Key Cryptography

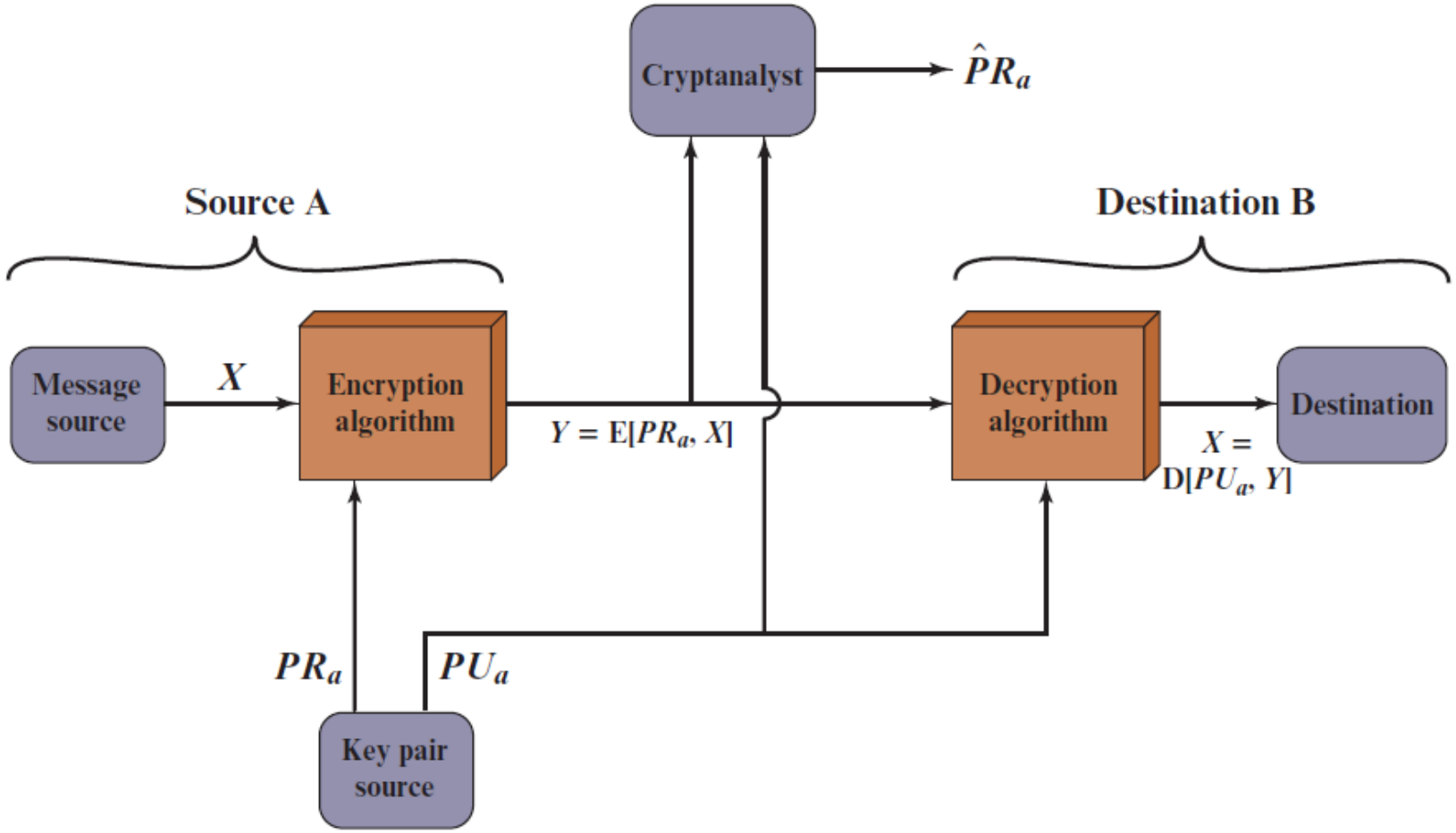
Public-Key Cryptosystem: Confidentiality

Figure 9.2 Public-Key Cryptosystem: Confidentiality



Public-Key Cryptosystem: Authentication

Figure 9.3 Public-Key Cryptosystem: Authentication



Public-Key Cryptosystem: Authentication and Secrecy

Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy

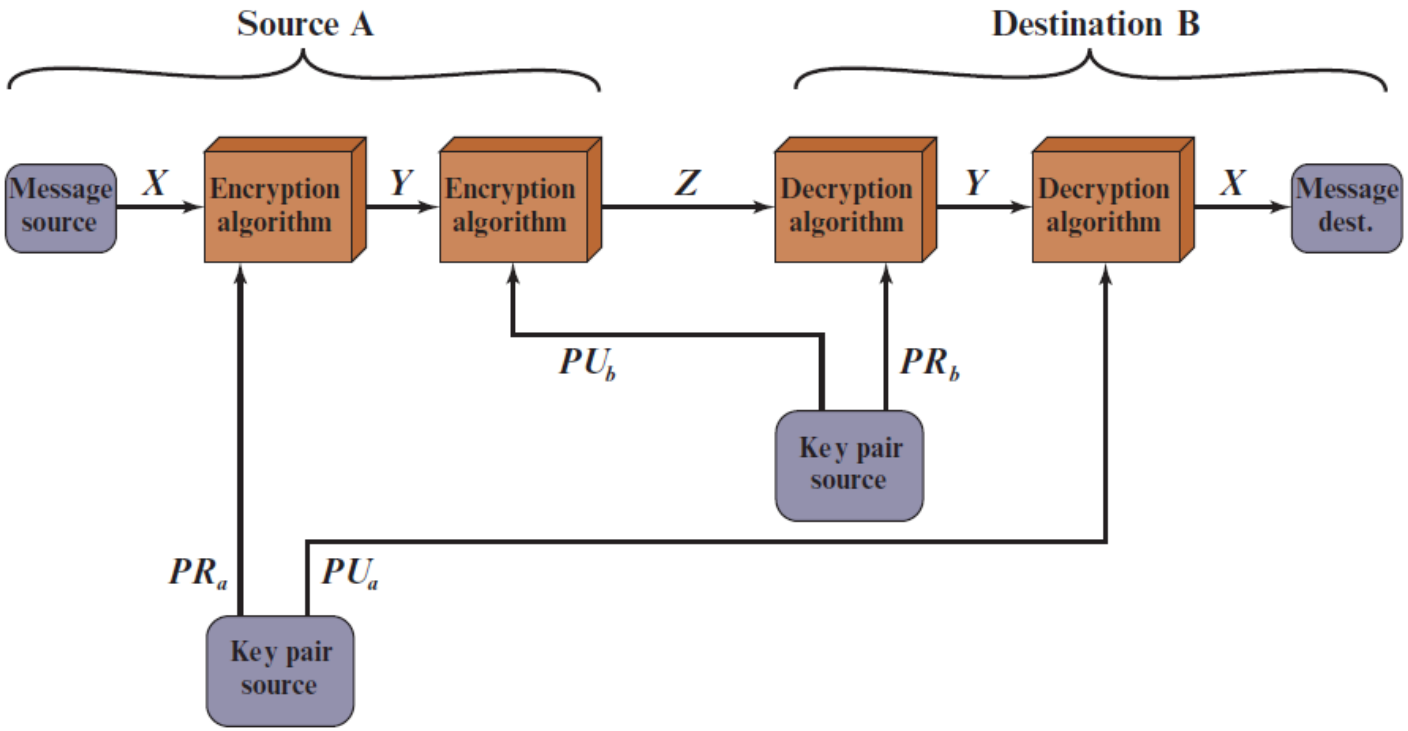


Table 9.3 Applications for Public-Key Cryptosystems

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie–Hellman	No	No	Yes
DSS	No	Yes	No

RSA Key Generation

- Two large distinct prime numbers ***p* and *q*, chosen at random**

- Block *B* of text has been encoded by some function into an integer *T* such that *T* is an integer and $0 < T < n$

- **Calculate $n=pq$**

- **Compute** the Euler phi function. $\varphi(n) = (p-1)(q-1)$ because *n* is the product of 2 primes.

- **Choose an integer *e*** such that $1 < e < \varphi(n)$ and **$\gcd(e,\varphi(n)) = 1$** (*e* and $\varphi(n)$ are co-prime)

- $\gcd(e,\varphi(n)) = 1$ means that 1 is a linear combination of *e* and $\varphi(n)$:

Use Euclidean algorithm to find unique value for d and such that $1 < d < \varphi(n)$

$$d * e \mod \varphi(n) = 1$$

Or , $\mathbf{d = e^{-1} \pmod{\varphi(n)}}$

- Public key is pair of values (*e*,*n*).

- Private key is pair values (*d*,*n*)

Figure 9.5 The RSA Algorithm

Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n)=(p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Example 1

- **Problem:**
- prime numbers p=3, q=11, e =7, plaintext T= 31

Find the key pair?

- **Calculate**
- $n = pq$ = 3x11 = 33
- $\varphi(n) = (p-1)(q-1)$ = 2x10 = 20
-

Example 1 cont'd

- So $n = 33$, $\varphi(n) = 20$, $e = 7$

- and d<20
- with 7 and 20 the (gcd(20, 7)).

- $d * e \mod \varphi(n) = 1$

  7*d mod 20 = 1

  so d = 3

Encryption and Decryption

Plaintext : $T < n$

Cipher Text: $C = T^{e} \pmod{n}$

Cipher Text : C

Plaintext : $T = C^{d} \pmod{n}$

Example 1 Cont'd

Encrypt:

Plaintext : Let $T = 31 < n$

Cipher Text: $C = T^{e} \pmod{n} = 31^{7} \pmod{33}$

$= 27512614111 \bmod 33$

$= 4$

Example 1 Cont'd

Decrypt:

Cipher Text : $C = 4$

Plaintext *:* $T = C^{d} \pmod{n} = 4^{3} \pmod{33}$

$=64 \bmod 33$

$=31$

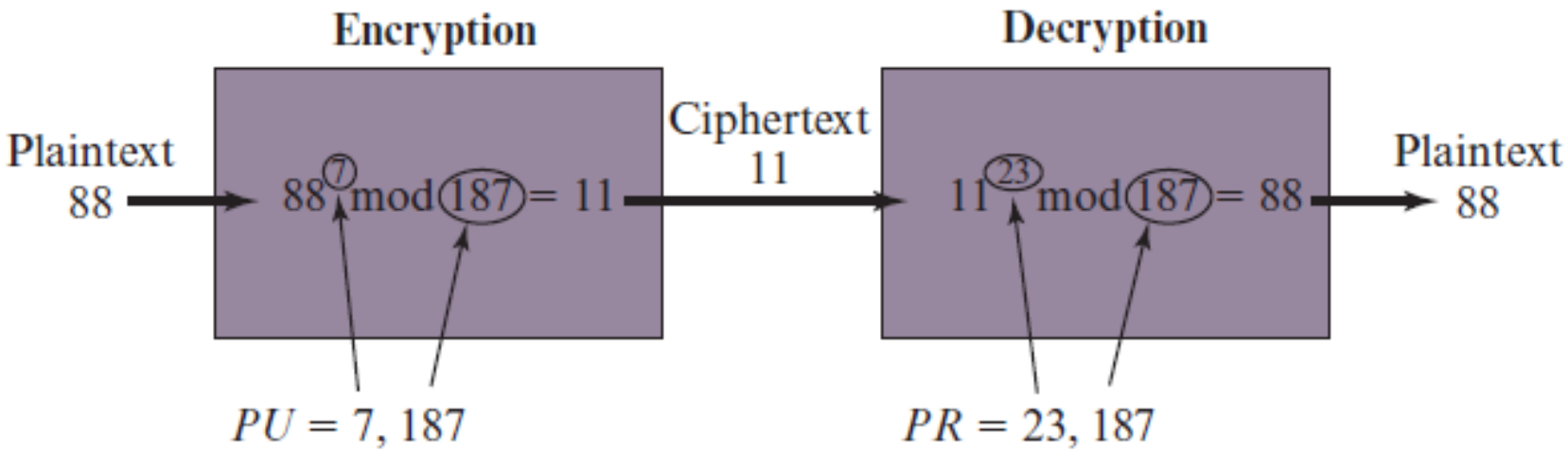
So the

Public key (encryption key)= (7, 33)

Private key (decryption key) = (3, 33)

Example of RSA Algorithm

Figure 9.6 Example of RSA Algorithm



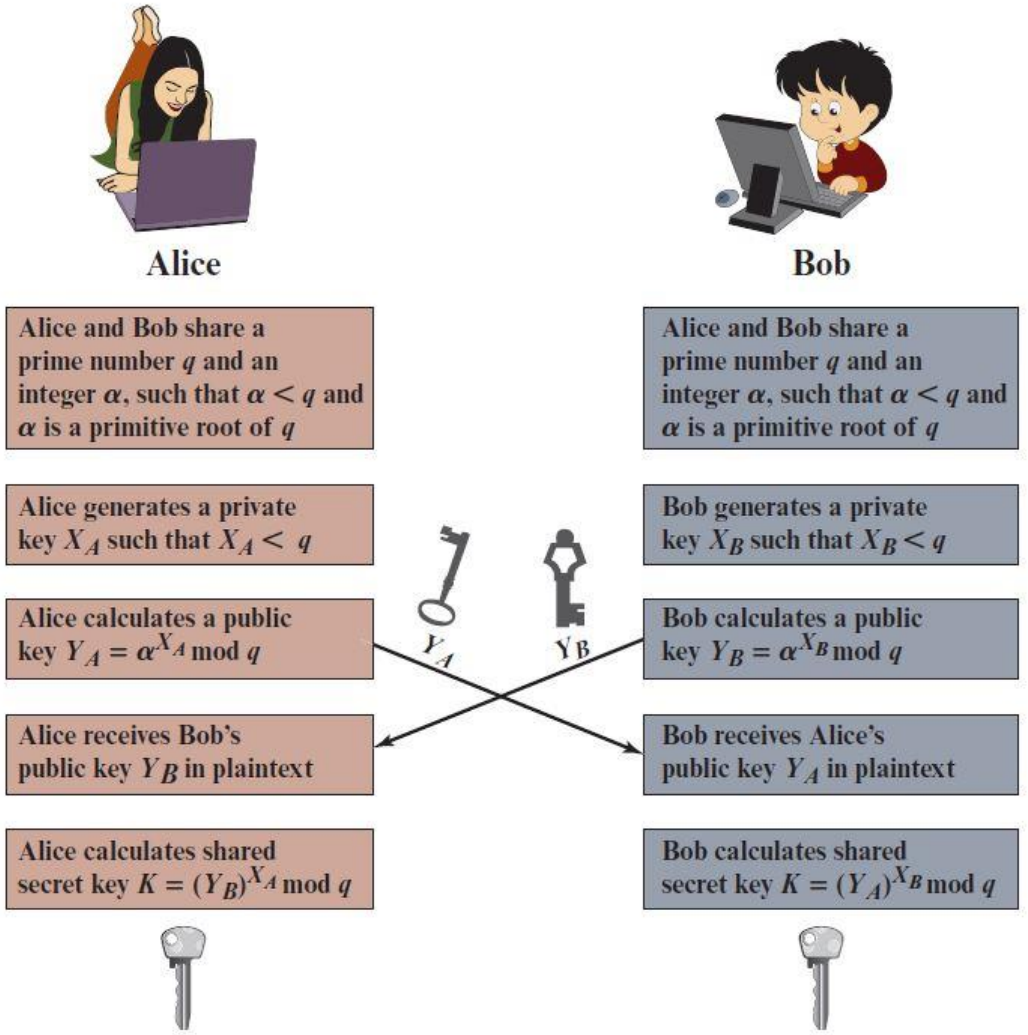
Chapter 10

Other Public-Key Cryptosystems

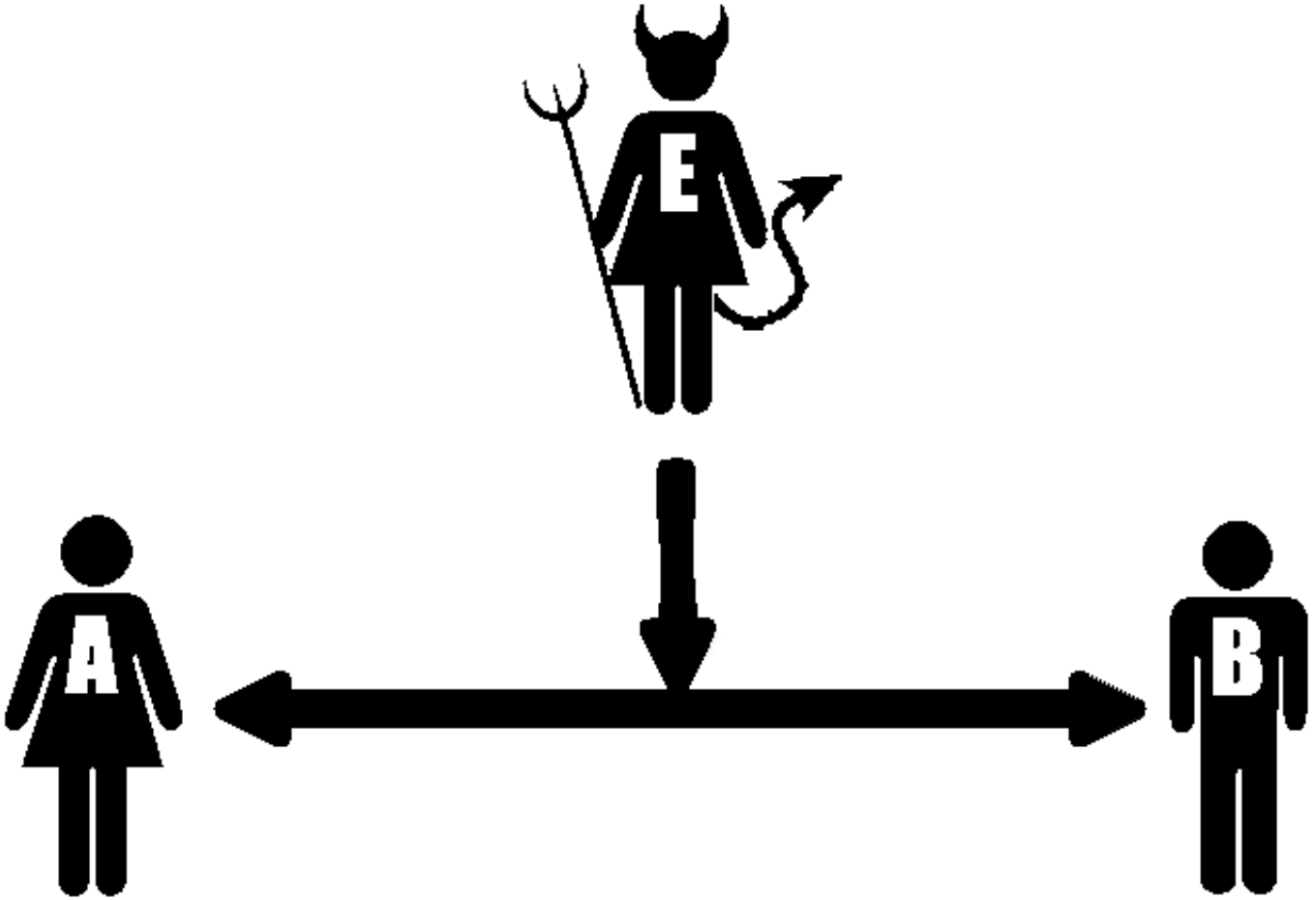
Diffie-Hellman Key Exchange

- First published public-key algorithm
- A number of commercial products employ this key exchange technique
- Purpose is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages
- The algorithm itself is limited to the exchange of secret values
- Its effectiveness depends on the difficulty of computing discrete logarithms

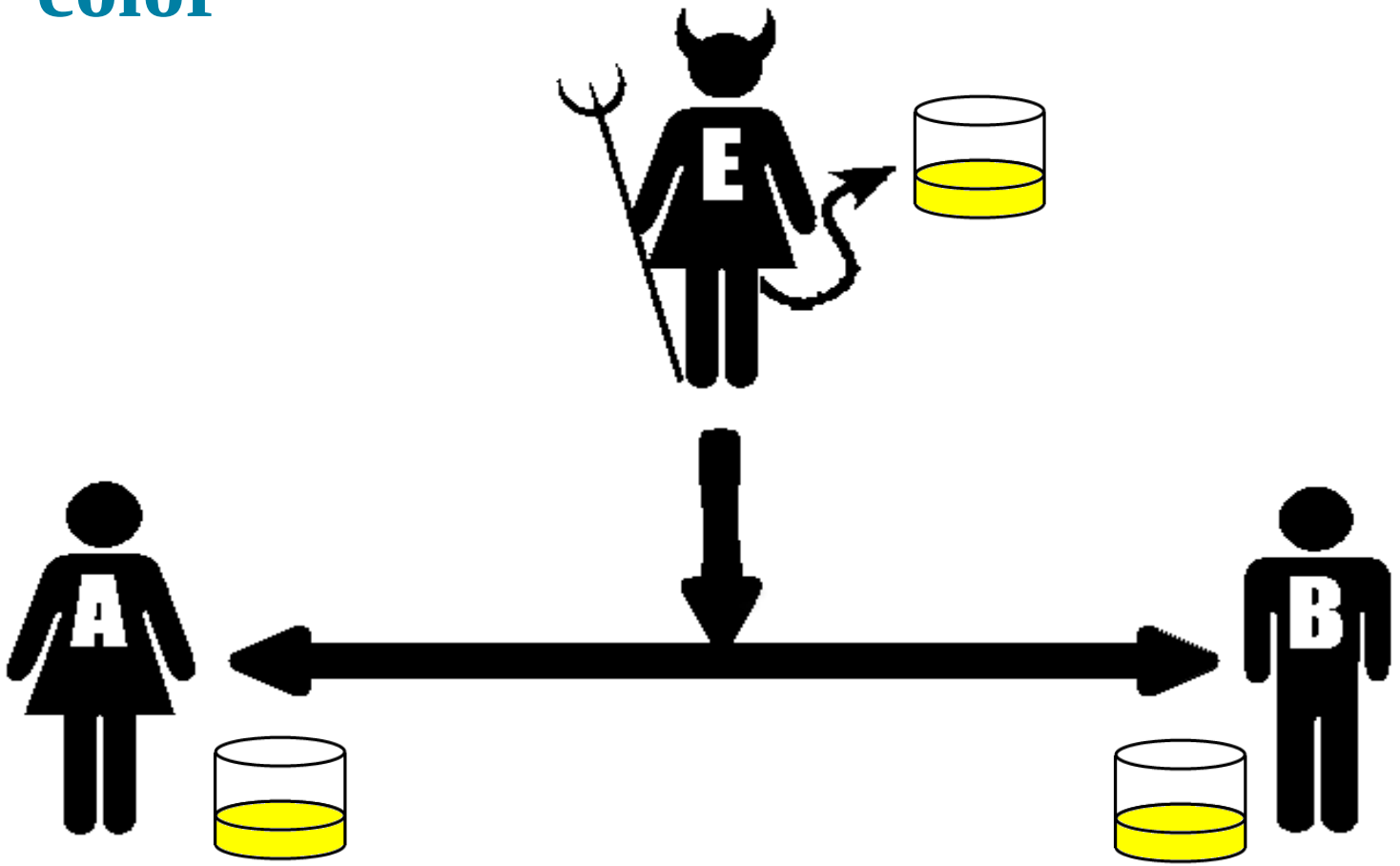
Figure 10.1 The Diffie–Hellman Key Exchange



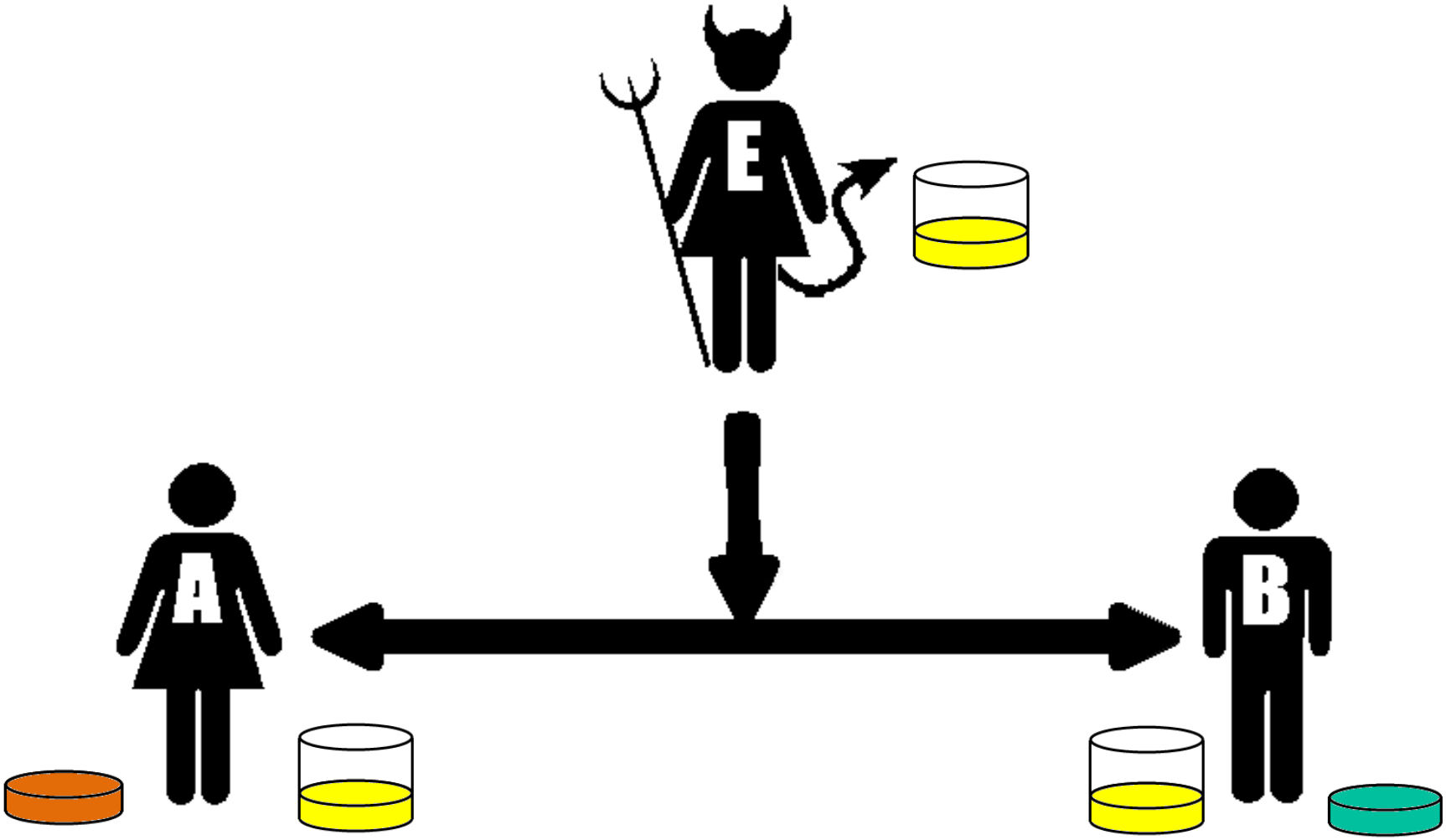
Alice & Bob with Eve listening wish to make a secret shared color



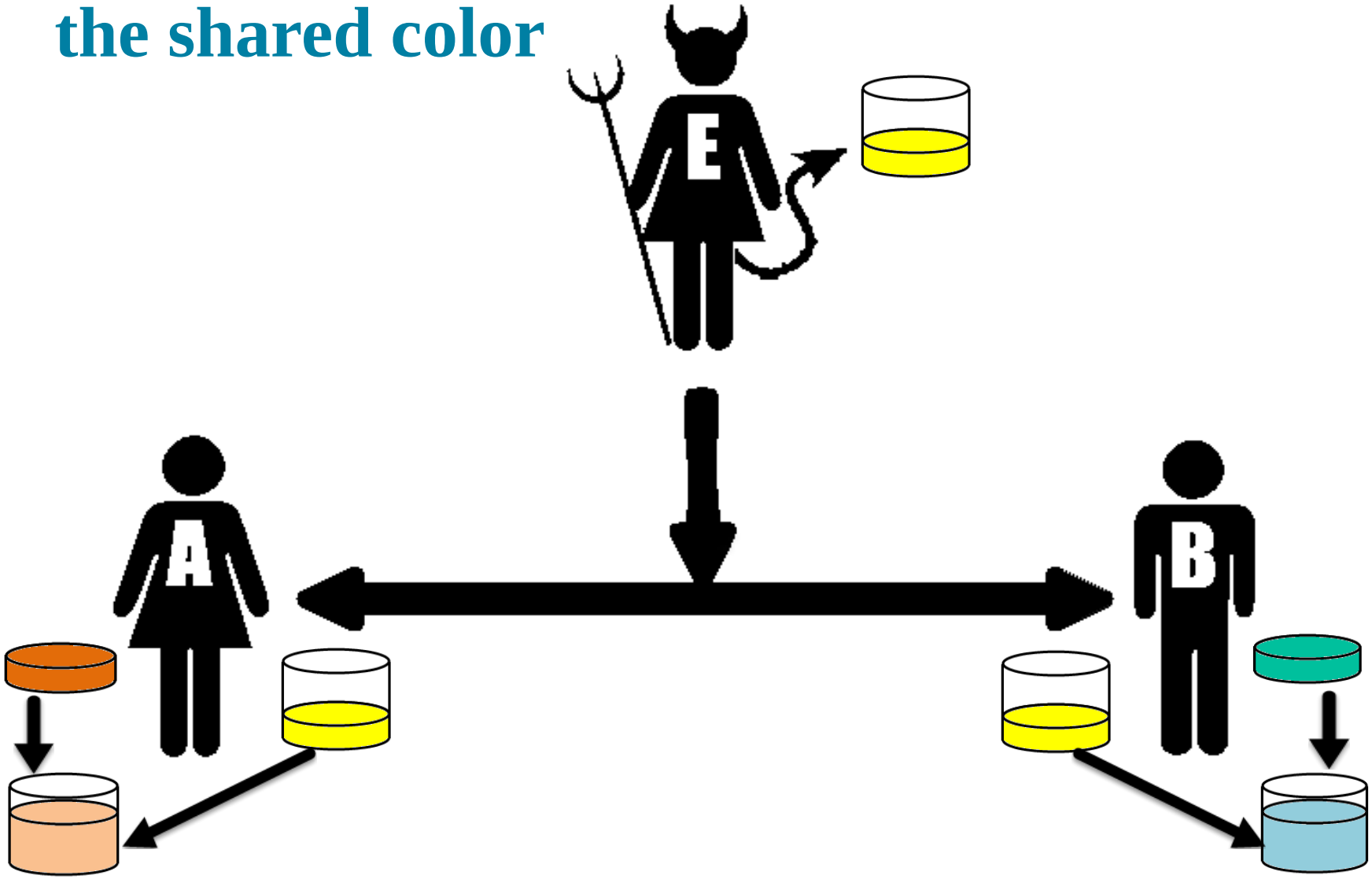
Step 1 - Both publicly agree to a shared color



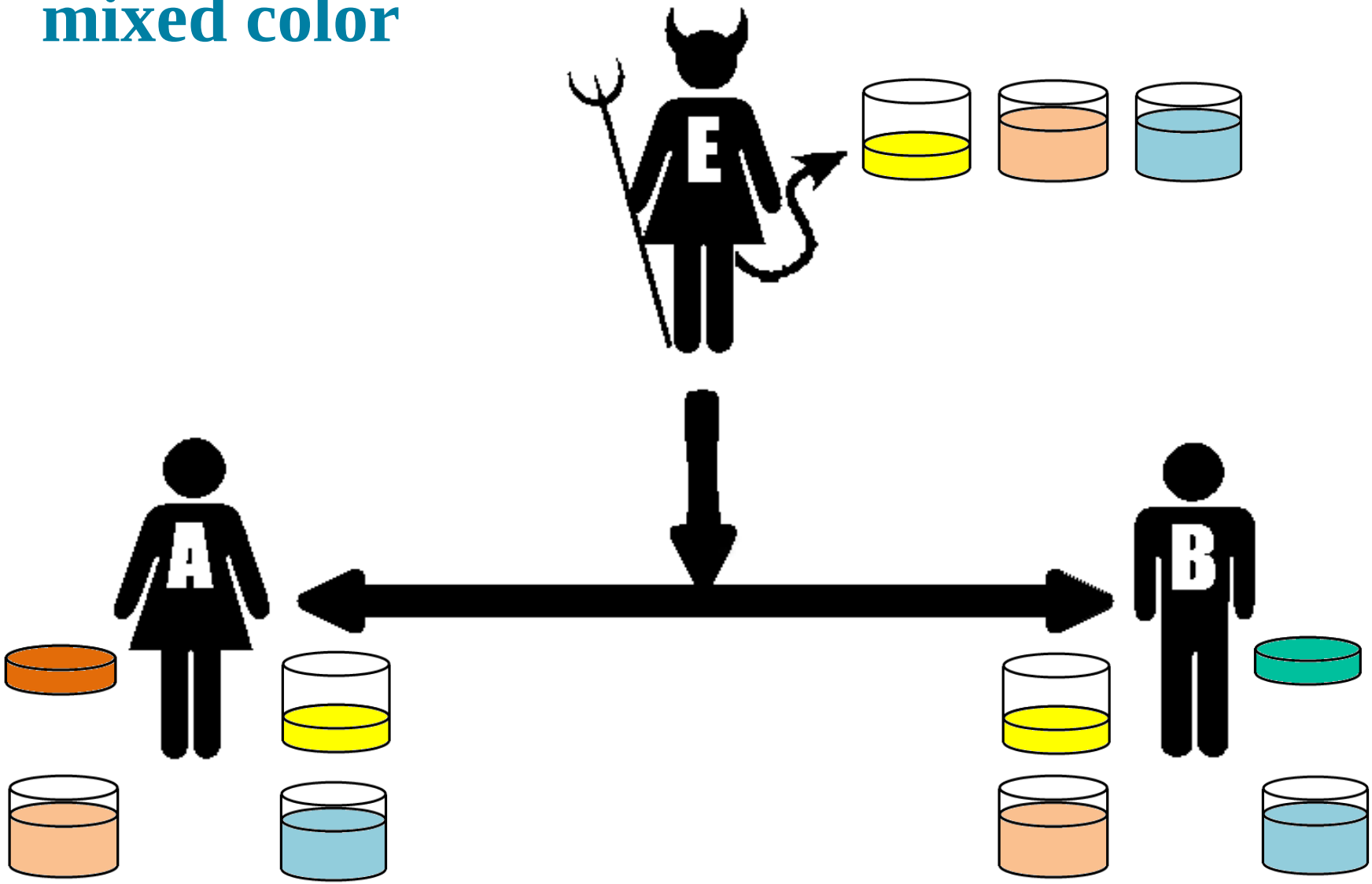
Step 2 - Each picks a secret color



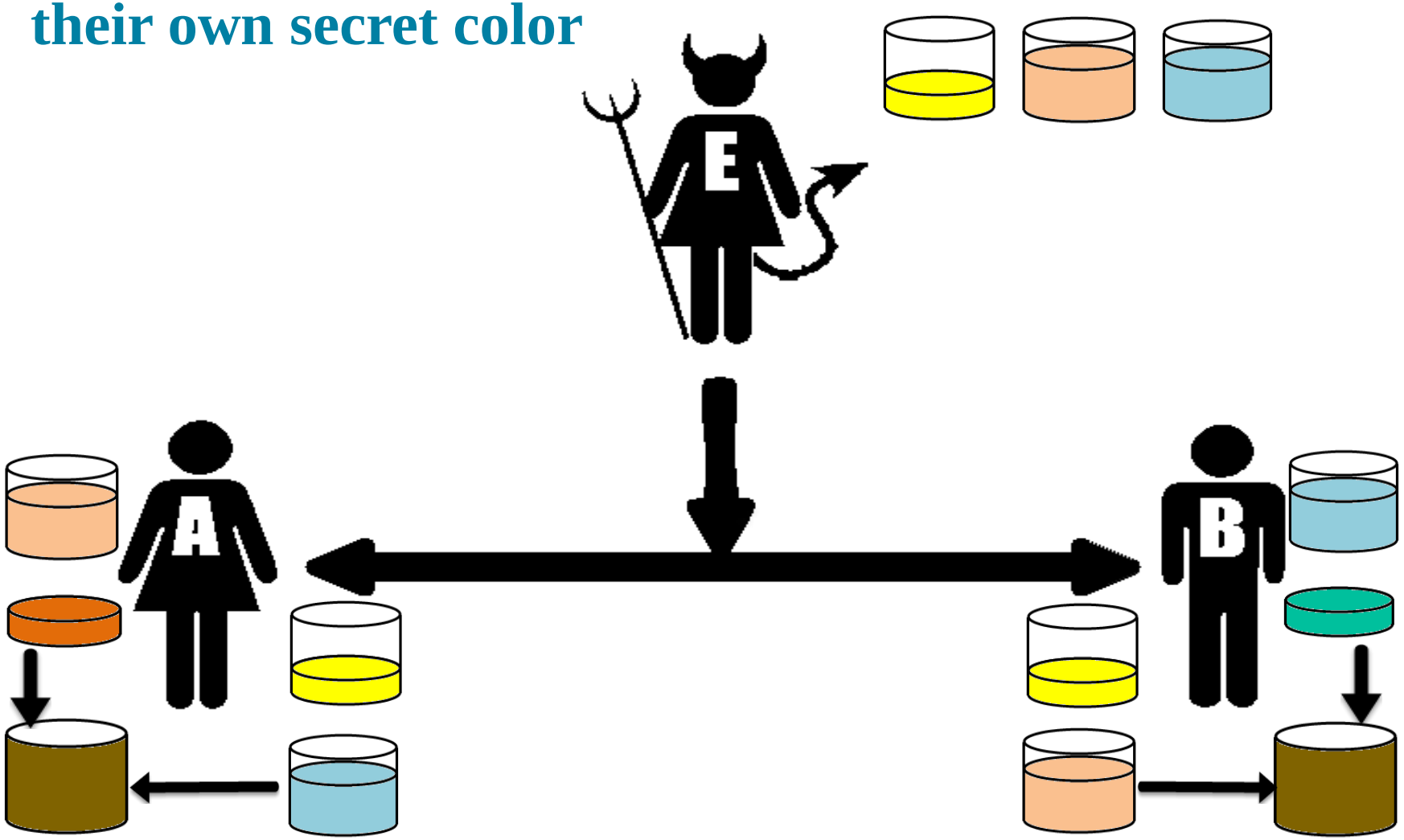
Step 3 - Each adds their secret color to the shared color



Step 4 - Each sends the other their new mixed color



Each combines the shared color from the other with their own secret color



⇒ Consider a prime number 'q'

⇒ Set α such that it must be the primitive root of q and $\alpha < q$

⇒ α is primitive root of q means:

$\alpha^1 \bmod q$

$\alpha^2 \bmod q$

$\alpha^3 \bmod q$

$\vdots$

$\alpha^{q-1} \bmod q$

→ gives result $\{1, 2, 3, \ldots q-1\}$

such as $3^1 \bmod 7 = 3$

$3^2 \bmod 7 = 2$

$3^3 \bmod 7 = 6$

$3^4 \bmod 7 = 4$

$3^5 \bmod 7 = 5$

$3^6 \bmod 7 = 1$

⇒ Values should not be repeated & we should have all the values in O/p set from 1 to 9-1

such as

$2^1 \bmod 7 = 2$

$2^2 \bmod 7 = 4$

$2^3 \bmod 7 = 1$

$2^4 \bmod 7 = 2$

$2^5 \bmod 7 =$

$2^6 \bmod 7 =$

So 2 can't be primitive root of 7

(3)

α and q $\Rightarrow$ these are global (public) elements (known to everyone in the network)

X $\Rightarrow$ Private key of the user (assume)

Y $\Rightarrow$ Public key of the user (assume)

(1) Now Assume X_A (Private key of A user) and $X_A < q$

So find Y_A (Public key of A) $= \alpha^{X_A} \bmod q$

(2) Similarly X_B (Private key of B user) and $X_B < q$

So find Y_B (Public key of B user) $= \alpha^{X_B} \bmod q$

(3) For secure exchang $K_A = K_B$

$K_A = (Y_B)^{X_A} \bmod q$, $K_B = (Y_A)^{X_B} \bmod q$

Question:- Diffi Hellman key exchange for (4)
the prime number 7 and primitive root 5
Private key for user 'A' is 3 and 'B' is 4.
Calculate their respective public keys and
check 'did they securely send the secret key'?

Ans:- $q=7$; $\alpha=5$; $X_A=3$; $X_B=4$; $Y_A=?$, $Y_B=?$

K_A equal to K_B ?

$Y_A \Rightarrow 5^3 \bmod 7 \Rightarrow 125 \bmod 7 \Rightarrow 6$, $Y_A = 6$

$X_B = 5^4 \bmod 7 \Rightarrow 2$ $Y_B = 2$

$K_A = (2)^3 \bmod 7 \Rightarrow 1$

$K_B = (6)^4 \bmod 7 \Rightarrow 1$

Man-in-the-Middle (MITM) Attack Concept

E{a,b,c} = Alice's, Bob's, and Charlie's public keys, respectively

Alice wants to send secure messages to Bob.

Charlie intercepts Alice's messages.

Charlie talks to Alice and pretends to be Bob.

Charlie talks to Bob and pretends to be Alice.

MITM Attack Concept

- Alice uses the *public key* she thinks she received from Bob (Charlie's)
- Bob uses the key he thinks is Alice's (also Charlie's)
- As a result, Charlie not only gains *access* to secure information but also can *modify* it (e.g. *transfer money to a different account* etc.)

Elgamal Cryptography

- Announced in 1984 by T. Elgamal
- Public-key scheme based on discrete logarithms closely related to the Diffie-Hellman technique
- Used in the digital signature standard (DSS) and the S/MIME e-mail standard
- Global elements are a prime number q and a which is a primitive root of q
- Security is based on the difficulty of computing discrete logarithms

Figure 10.3 The ElGamal Cryptosystem

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice	
Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

What's wrong with RSA?

- RSA is based upon the 'belief' that factoring is 'difficult' – never been proven
- Prime numbers are getting too large
- Amount of research currently devoted to factoring algorithms
- Quantum computing will make RSA obsolete overnight

Table 10.3 Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

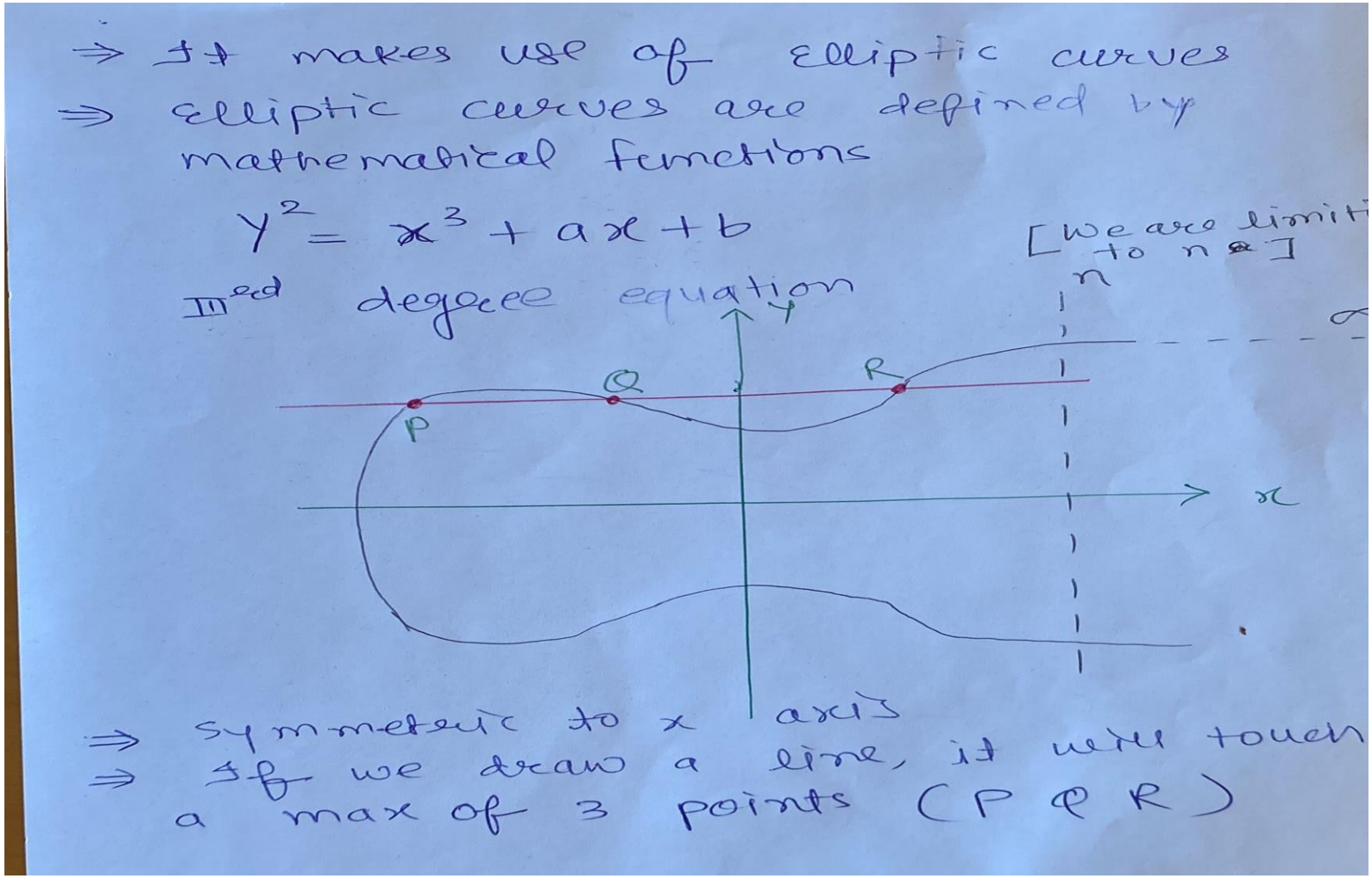
Symmetric Key Algorithms	Diffie–Hellman, Digital Signature Algorithm	RSA (size of n in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512 +

Note: L = size of public key, N = size of private key.

Elliptic Curve Arithmetic

- Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA
  - The key length for secure RSA use has increased over recent years and this has put a heavier processing load on applications using RSA
- Elliptic curve cryptography (ECC) is showing up in standardization efforts including the IEEE P1363 Standard for Public-Key Cryptography
- Principal attraction of ECC is that it appears to offer equal security for a far smaller key size

Elliptic Curve Cryptography (ECC)



Elliptic Curve Cryptography (ECC)

- Addition operation in ECC is the counterpart of modular multiplication in RSA
- Multiple addition is the counterpart of modular exponentiation
- To form a cryptographic system using elliptic curves, we need to find a "hard problem" corresponding to factoring the product of two primes or taking the discrete logarithm
  - $Q=kP$, where Q, P belong to a prime curve
  - Is "easy" to compute Q given k and P
  - But "hard" to find k given Q, and P
  - Known as the elliptic curve logarithm problem

Figure 10.7 ECC Diffie–Hellman Key Exchange

Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

Security of Elliptic Curve Cryptography

- Depends on the difficulty of the elliptic curve logarithm problem
- Fastest known technique is “Pollard rho method”
- Compared to factoring, can use much smaller key sizes than with RSA
- For equivalent key lengths computations are roughly equivalent
- Hence, for similar security ECC offers significant computational advantages

Chapter 11

Cryptographic Hash Functions

Hash Functions

- A hash function H accepts a variable-length block of data *M* as input and produces a fixed-size hash value
  - $h = H(M)$
  - Principal object is data integrity
- Cryptographic hash function
  - An algorithm for which it is computationally infeasible to find either:

    (a) a data object that maps to a pre-specified hash result (the one-way property)

    (b) two data objects that map to the same hash result (the collision-free property)

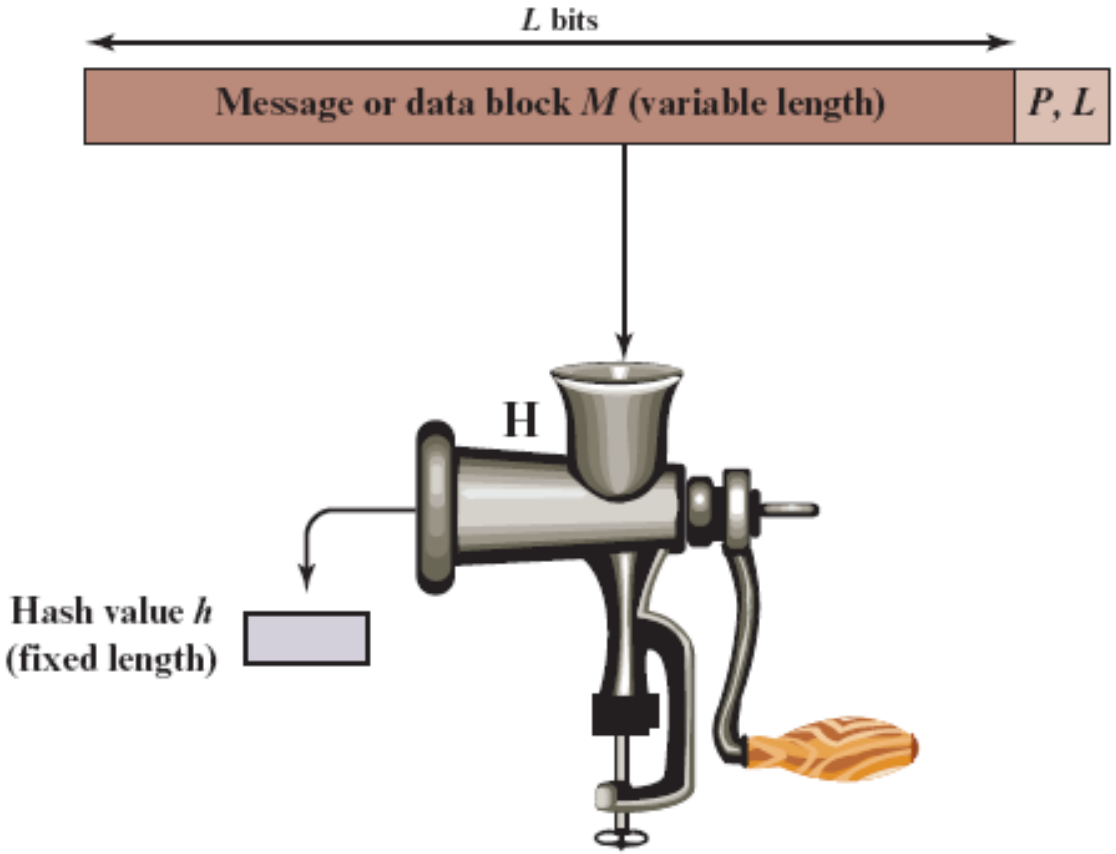
Application of Hash Function in daily life

1. Password storage: Many websites and applications use hash functions to store passwords securely. Instead of storing the actual passwords, the websites and applications store the hash values of the passwords. When a user logs in, the system hashes the user's input and compares it to the stored hash value. If the values match, the user is granted access.

2. Data integrity verification: Hash functions are used to ensure the integrity of data during transmission or storage. For example, when you download a file from the internet, the website may provide the hash value of the file. After you download the file, you can use a hash function to generate a hash value for the file and compare it to the hash value provided by the website. If the values match, you can be sure that the file has not been tampered with during transmission.

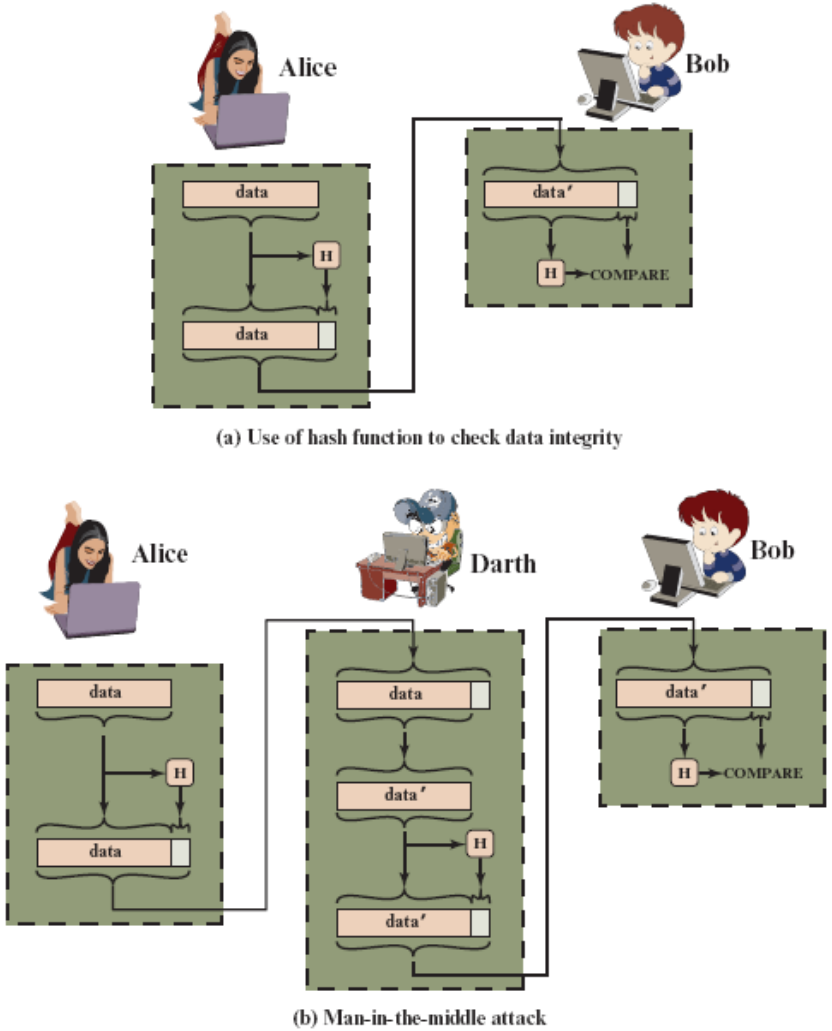
3. Digital signatures:

Figure 11.1 Cryptographic Hash Function; $h = H(M)$



P, L = padding plus length field

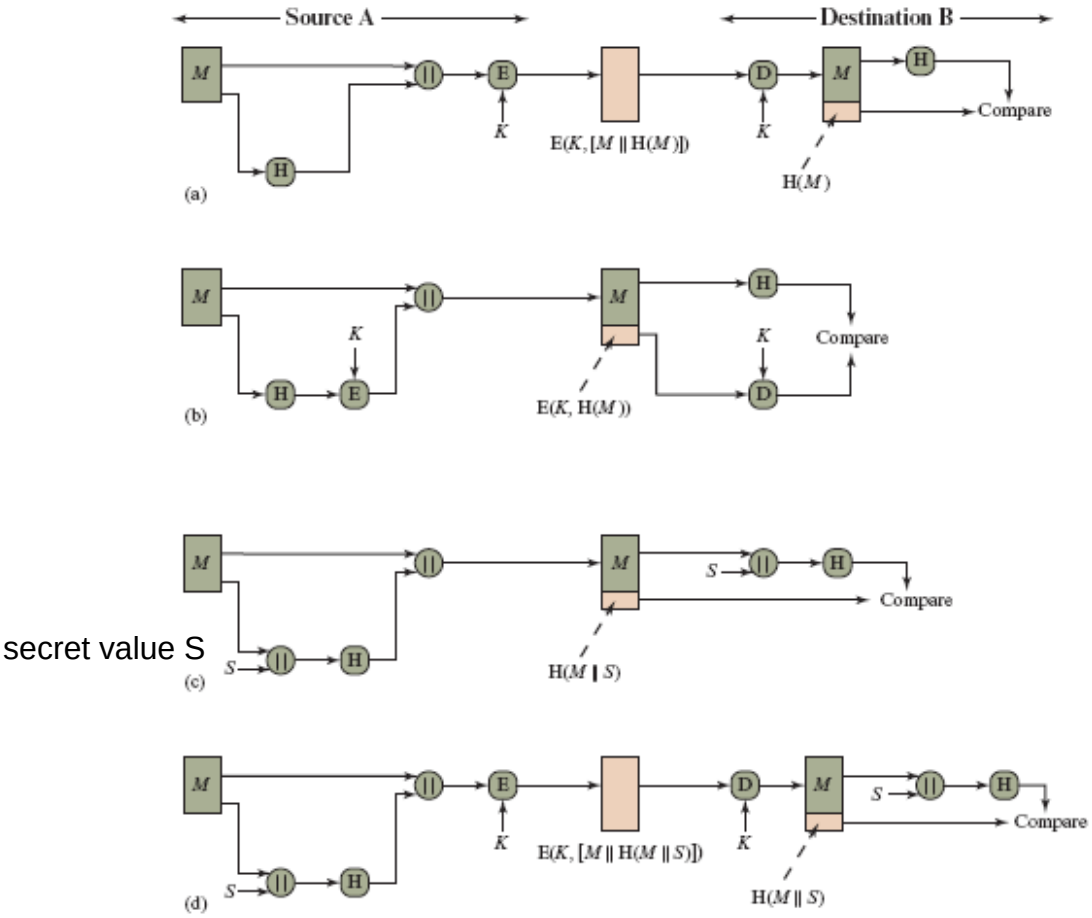
Figure 11.2 Attack Against Hash Function



(a) Use of hash function to check data integrity

(b) Man-in-the-middle attack

Figure 11.3 Simplified Examples of the Use of a Hash Function for Message Authentication



secret value S

Message Authentication Code (MAC)

- Also known as a *keyed hash function*
- Typically used between two parties that share a secret key to authenticate information exchanged between those parties
- Takes as input a secret key and a data block and produces a hash value (MAC) which is associated with the protected message
  - If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value
  - An attacker who alters the message will be unable to alter the associated MAC value without knowledge of the secret key

Digital Signature

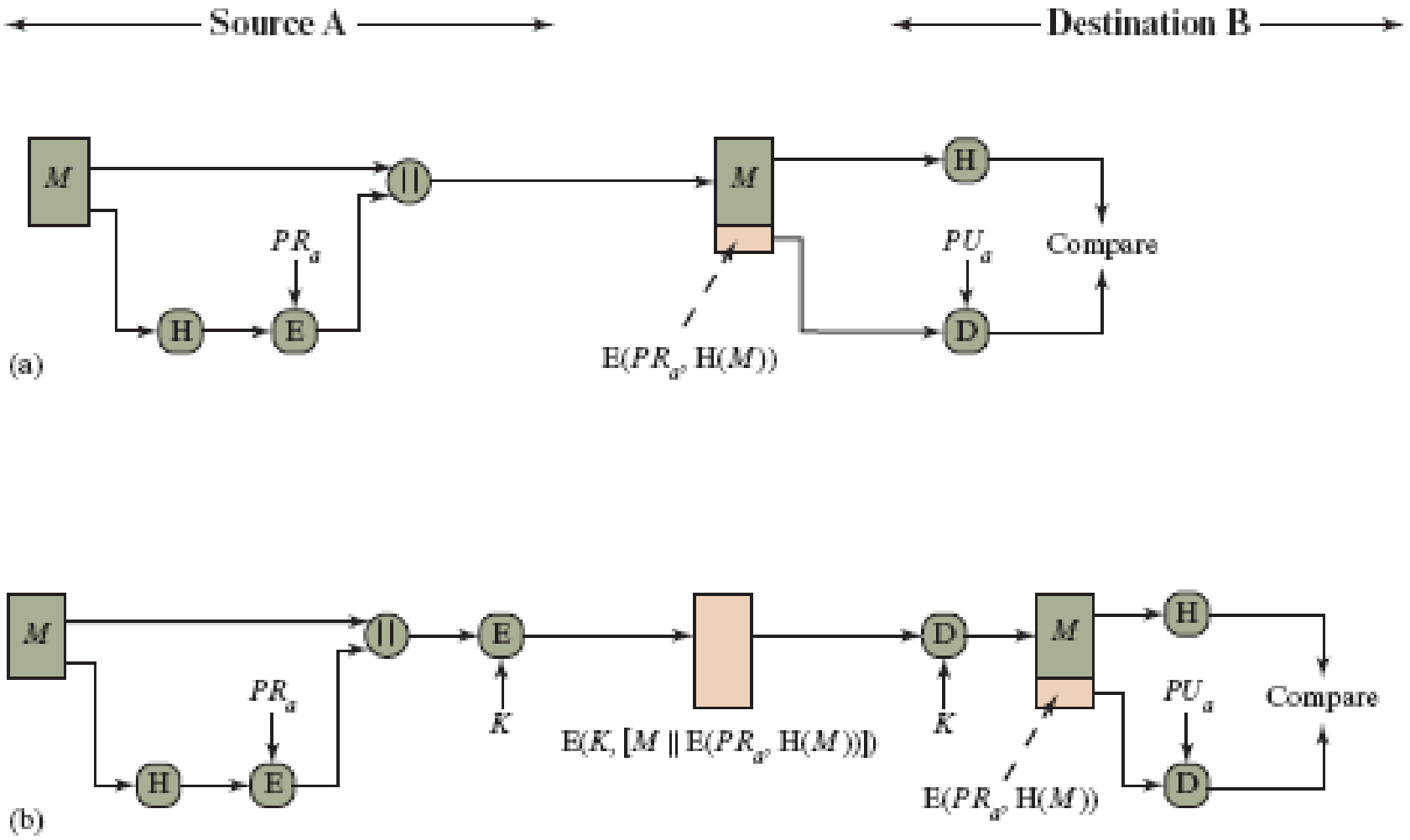
- Operation is similar to that of the MAC
- The **hash value** of a message is **encrypted** with a user's **private key**
- Anyone who knows the user's public key can verify the integrity of the message
- Implications of digital signatures go beyond just message authentication

Digital Signature applications

- DSS has several applications in daily life, some of which are:

1. **Electronic contracts**: DSS is commonly used in electronic contracts to provide legal authenticity and enforceability. By using digital signatures, parties can sign contracts remotely without the need for physical signatures, and the signatures are legally binding.
2. **Online banking:** DSS is widely used in online banking to secure transactions and prevent fraud. Digital signatures are used to authenticate transactions and ensure that they cannot be altered after signing.
3. **Email communication:** DSS can be used to secure email communication by providing authenticity and integrity to the email. By using digital signatures, the recipient can verify the identity of the sender and ensure that the message has not been tampered with.
4. **Software distribution:** DSS can be used to ensure that software packages are authentic and have not been tampered with. By using digital signatures, software developers can sign their packages, and users can verify the signatures before installing the software.

Figure 11.4 Simplified Examples of Digital Signatures



Secure Hash Algorithm (SHA)

- SHA was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993
- Was revised in 1995 as SHA-1
- Based on the hash function MD4 and its design closely models MD_4
- Produces 160-bit hash values
- In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with hash value lengths of 256, 384, and 512
  - Collectively known as SHA-2

Table 11.3 Comparison of SHA Parameters

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Note: All sizes are measured in bits.

SHA-1

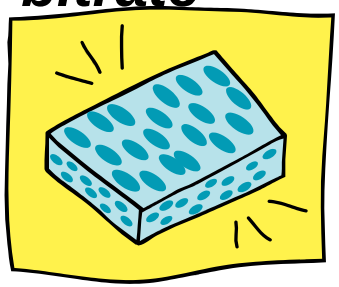
- Google publicly broke one of the major algorithms in web encryption, called SHA-1. The company's researchers showed that with enough computing power — **roughly 110 years** of computing from a single GPU for just one of the phases — you can produce a collision, effectively breaking the algorithm.

SHA-3

- SHA-1 has not yet been "broken”
  - No one has demonstrated a technique for producing collisions in a practical amount of time
  - Considered to be insecure and has been phased out for SHA-2
- SHA-2 shares the same structure and mathematical operations as its predecessors so this is a cause for concern
  - Because it will take years to find a suitable replacement for SHA-2 should it become vulnerable, NIST decided to begin the process of developing a new hash standard
- NIST announced in 2007 a competition for the SHA-3 next generation NIST hash function
  - Winning design was announced by NIST in October 2012
  - SHA-3 is a cryptographic hash function that is intended to complement SHA-2 as the approved standard for a wide range of applications

The Sponge Construction

- Underlying structure of SHA-3 is a scheme referred to by its designers as a *sponge construction*
- Takes an **input message and partitions** it into **fixed-size blocks**
- Each block is **processed** in turn with the **output** of each iteration fed into the **next iteration**, finally producing an output block
- The sponge function is defined by three parameters:
  - f = the **internal function** used to process each input block
  - r = the size in bits of the input blocks, called the ***bitrate***
  - pad = the padding algorithm



Chapter 12

Message Authentication Codes

MAC applications in daily life:

1. Online transactions: MAC is widely used in online transactions, such as online banking, e-commerce, and payment gateways. MAC provides data integrity and authenticity, ensuring that the transactions are secure and cannot be tampered with.

2. Email communication: MAC can be used to secure email communication by providing message integrity and authenticity. By appending a MAC tag to the email, the recipient can verify that the email was sent by the claimed sender and that the message has not been altered during transmission.

3. Secure messaging apps: Many messaging apps, such as WhatsApp, Signal, and Telegram, use MAC to secure their messages. By using MAC, these apps can ensure that the messages are authentic, and any attempts to modify the messages will be detected.

4. Access control: MAC can be used to provide access control in computer systems. By generating a MAC tag for each user, the system can authenticate the user's identity and grant access only to authorized users.

- Overall, MAC is a valuable tool for ensuring data integrity and authenticity in various applications.

Message Authentication Requirements (1 of 2)

- Disclosure
  - Release of message contents to any person or process not possessing the appropriate cryptographic key
- Traffic analysis
  - Discovery of the pattern of traffic between parties
- Masquerade
  - Insertion of messages into the network from a fraudulent source
- Content modification
  - Changes to the contents of a message, including insertion, deletion, transposition, and modification

Message Authentication Requirements (2 of 2)

- Sequence modification
  - Any modification to a sequence of messages between parties, including insertion, deletion, and reordering
- Timing modification
  - Delay or replay of messages
- Source repudiation
  - Denial of transmission of message by source
- Destination repudiation
  - Denial of receipt of message by destination

Message Authentication Functions

- Two levels of functionality:
- Lower level
  - There must be some sort of function that produces an authenticator
- Higher-level
  - Uses the lower-level function as a primitive in an authentication protocol that enables a receiver to verify the authenticity of a message
- Hash function
  - A function that maps a message of any length into a fixed-length hash value which serves as the authenticator
- Message encryption
  - The ciphertext of the entire message serves as its authenticator
- Message authentication code (MAC)
  - A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

Figure 12.1 Basic Uses of Message Encryption

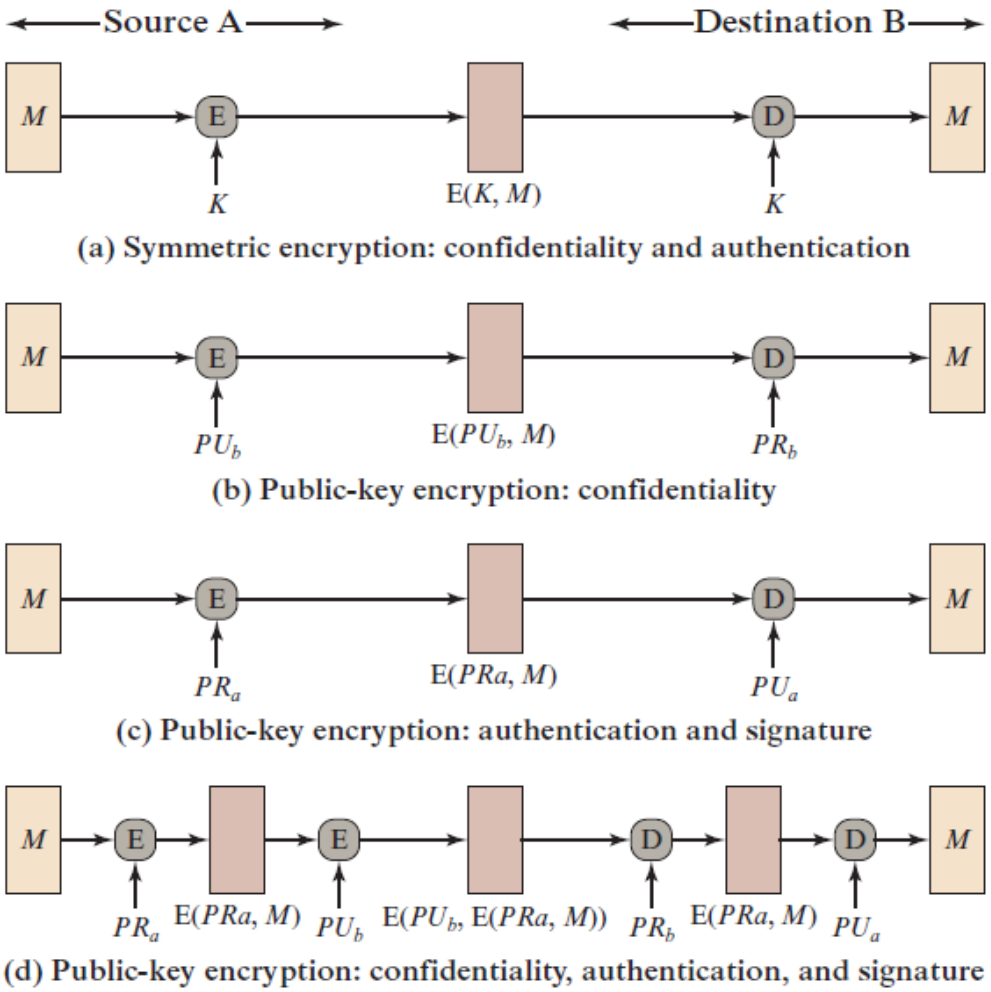


Figure 12.2 Internal and External Error Control

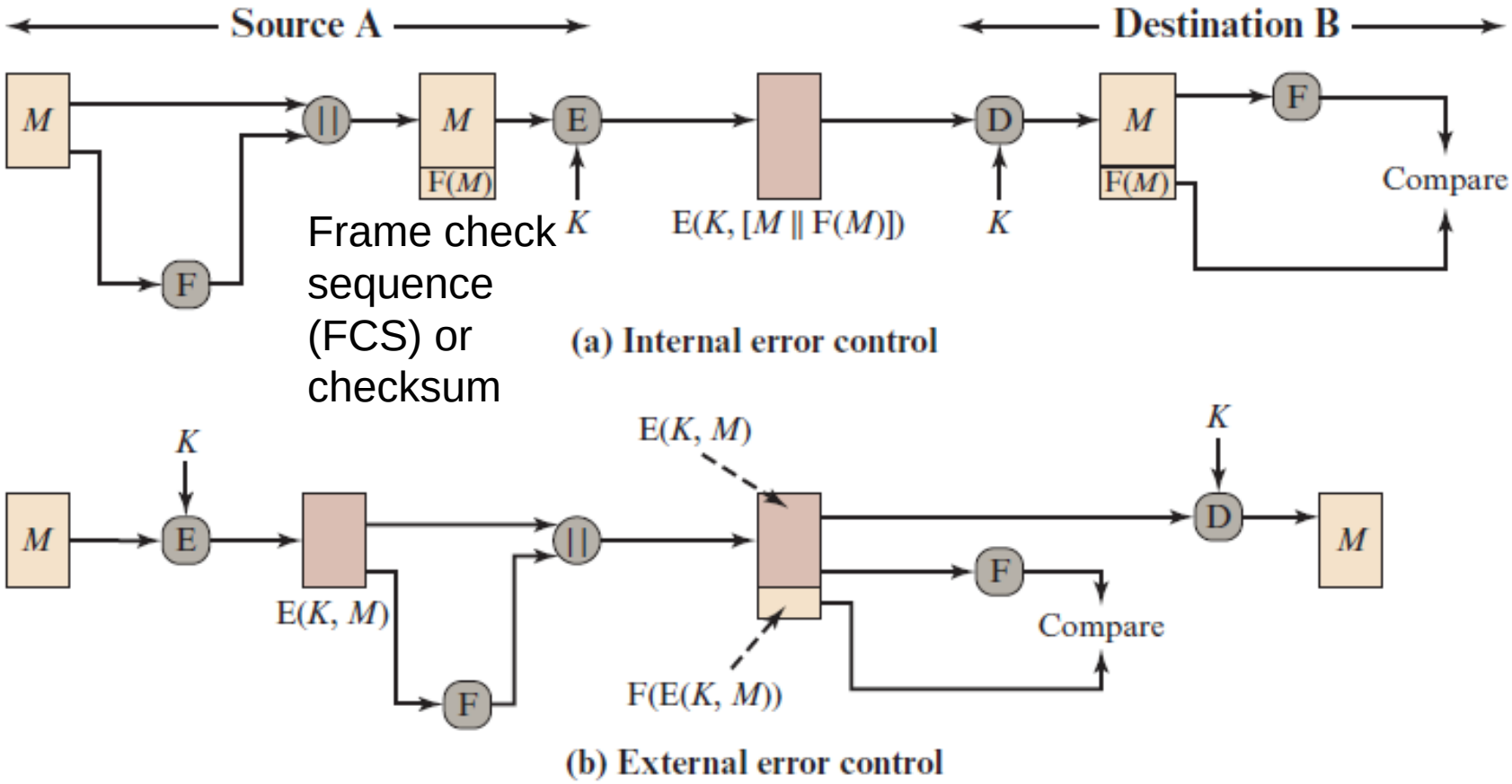
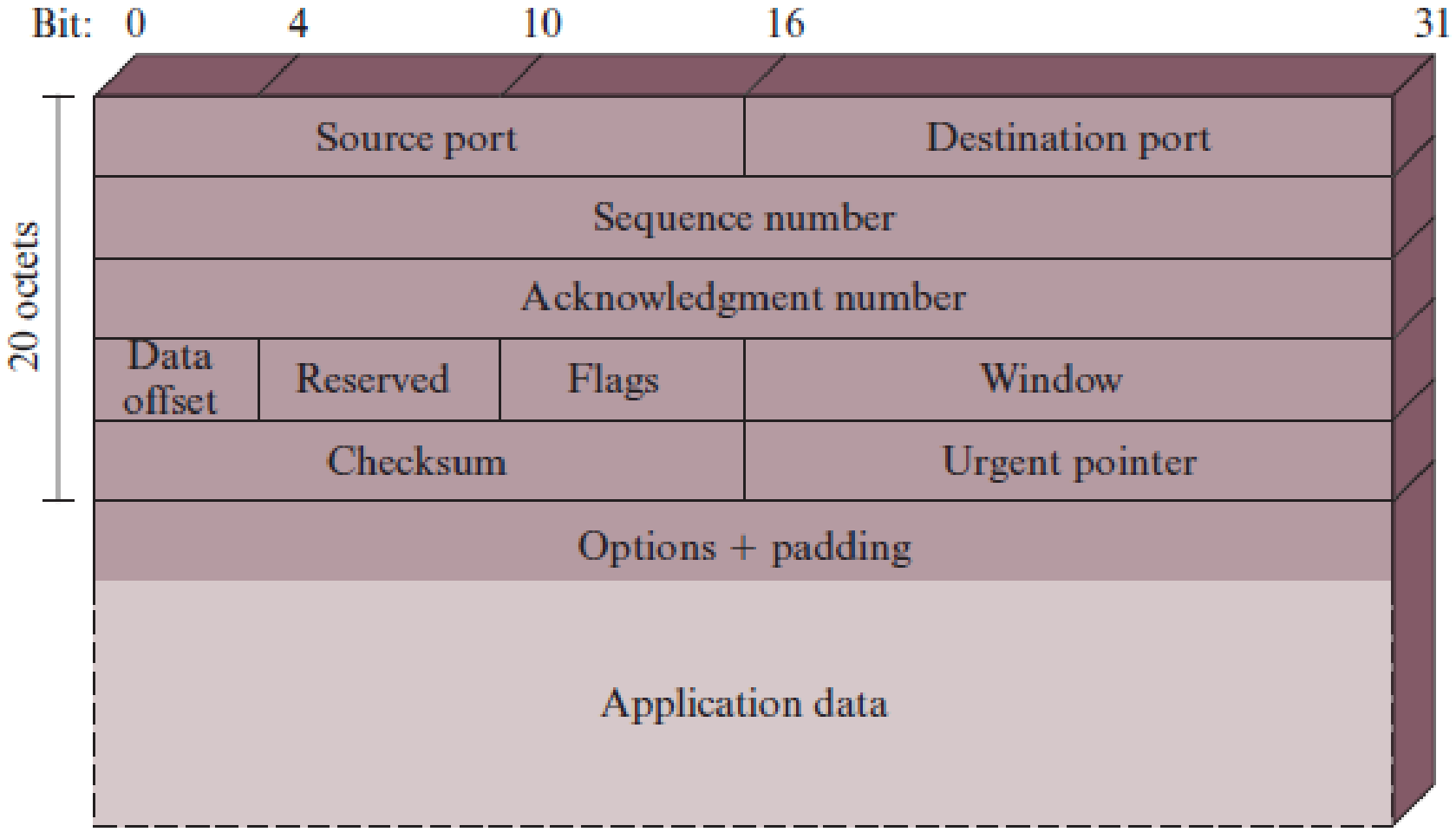


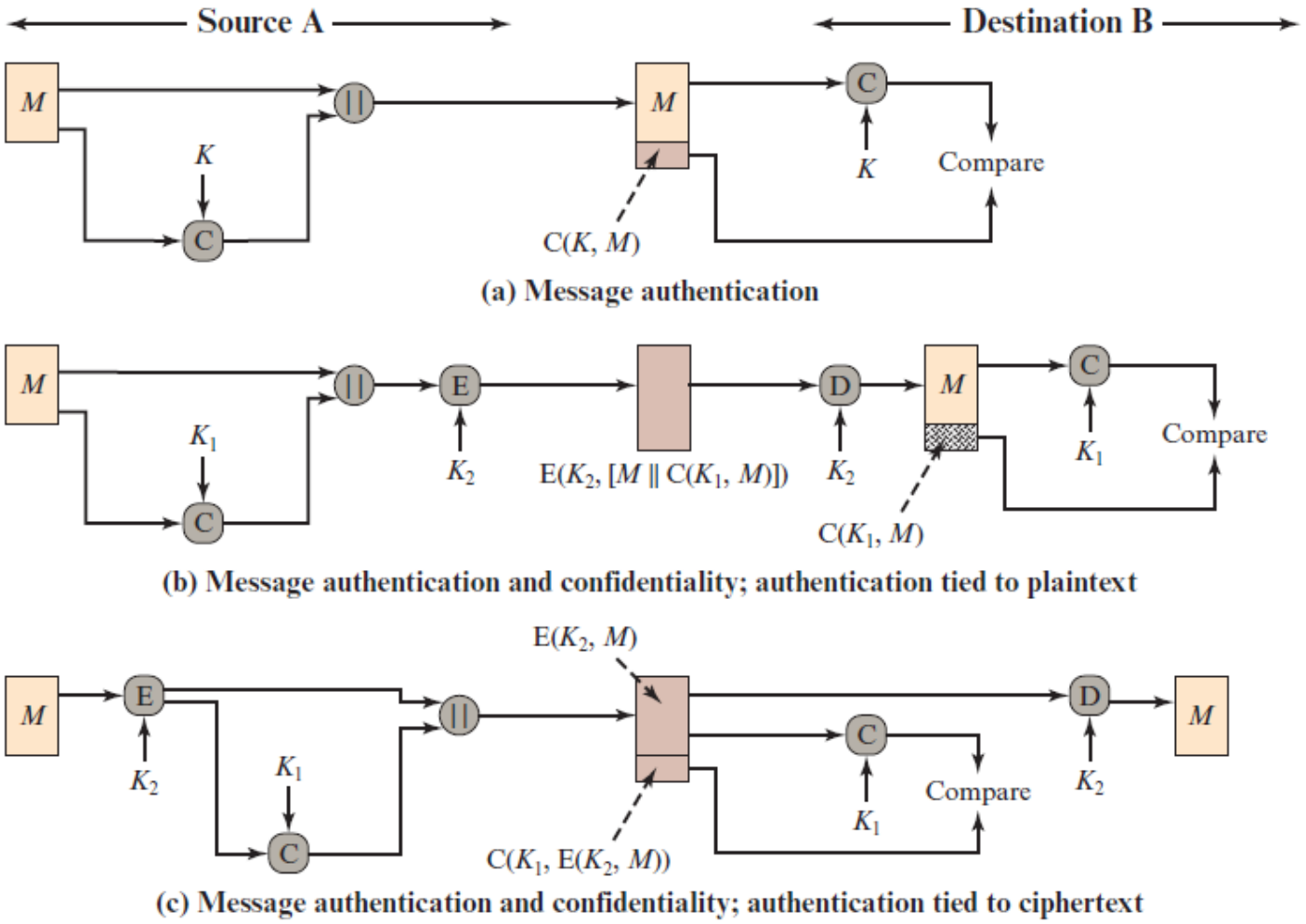
Figure 12.3 TCP Segment



Public-Key Encryption

- The straightforward use of public-key encryption provides **confidentiality but not authentication**
- To provide both confidentiality and authentication, A can encrypt *M* first using its private key which provides the digital signature, and then using B's public key, which provides confidentiality
- Disadvantage is that the public-key algorithm must be exercised four times rather than two in each communication

Figure 12.4 Basic Uses of Message Authentication code (MAC)



MACs Based on Hash Functions: HMAC

- There has been increased interest in developing a MAC derived from a cryptographic hash function
- Motivations:
  - Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES
  - Library code for cryptographic hash functions is widely available
- HMAC has been chosen as the mandatory-to-implement MAC for IP security
- Has also been issued as a NIST standard (FIPS 198)

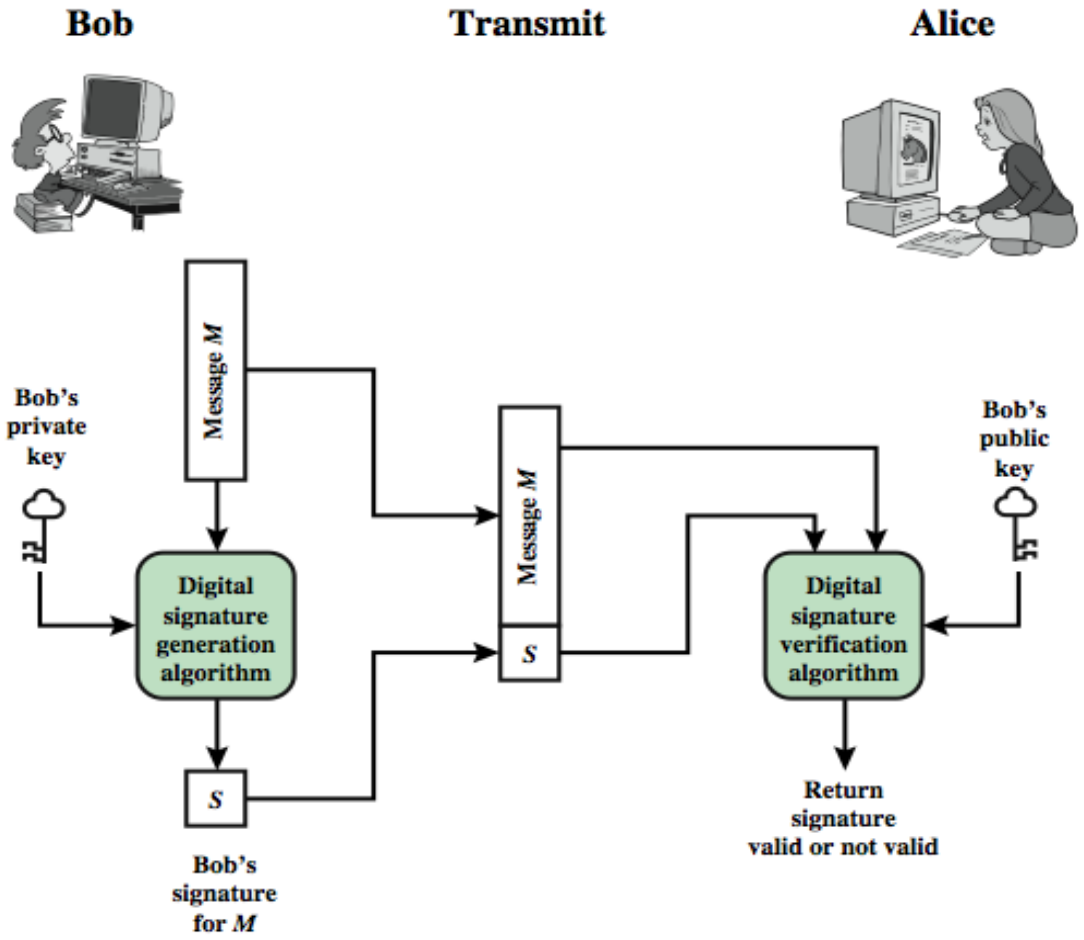
Digital Signatures

- 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 Model



Definition

Birthday attacks are a class of brute-force techniques that target the cryptographic hash functions. The goal is to take a cryptographic hash function and find two different inputs that produce the same output.

Birthday Attacks

The probability that all 23 people have different birthdays is

$$1 \times (1 - \frac{1}{365})(1 - \frac{2}{365}) \ldots (1 - \frac{22}{365}) = 0.493$$

Therefore, the probability of at least two having the same birthday is 1- 0.493=0.507

More generally, suppose we have N objects, where N is large. There are r people, and each chooses an object. Then

$$P(\text{there is a match}) \approx 1 - e^{-r^2/2N}$$

Birthday Attacks

(Example) We have 40 license plates, each ending in a 3-digit number. What is the probability that two of the license plates end in the same 3 digits?

(Solution) N=1000, r=40

1. Approximation: $1-(1-\frac{1}{1000})(1-\frac{2}{1000})...(1-\frac{39}{1000})=0.546$

2. The exact answer:

$$1-e^{-40^2/2\cdot 1000}=0.551$$

Attack Prevention

The important property is the length in bits of the message digest produced by the hash function.

If the number of *m* bit hash , the cardinality *n* of the hash function is

$$n = 2^m$$

The 0.5 probability of collision for m bit hash, expected number of operation *k* before finding a collision is very close to

$$k \approx \sqrt{n} = 2^{m/2}$$

m should be large enough so that it's not feasible to compute hash

References

1. http://www.sfu.ca/~ljilja/ENSC427/News/Kurose_Ross/Chapter_8_V7.0_Accessible.pdf
2. Stallings, W., Cryptography and Network Security: Principles and Practice 0133354695, 9780133354690.
3. **A.K. Dewdney, The New Turning Omnibus, pp. 250-257, Henry Holt and Company, 2001.**