INF3510 Information Security
University of Oslo
Spring 2010

Lecture 2

Cryptography

Audun Jøsang

Outline

- What is cryptography?
- When is cryptography used?
- Is there a ‘perfect’ cipher?
- Symmetric ciphers
  - Stream ciphers
  - Block ciphers
- Hash functions
- Asymmetric ciphers
- Ciphers and security

What is cryptography?

- Cryptography is part of the field of study known as **cryptology**.
- Cryptology includes
  - cryptography:
    - derived from the Greek, means ‘hidden writing’.
    - the study of methods for secret writing: for transforming messages into an unintelligible form, and for recovering them, using some secret knowledge.
  - cryptanalysis:
    - analysis of cryptographic systems, inputs and outputs to derive confidential information, usually without using the secret knowledge.

What is cryptography?

- Cryptographic terminology and notation:
  - Plaintext message (M): the original message or data
  - Encryption (E): transformation of the plaintext into another form so that the meaning is not obvious, using an algorithm and some secret knowledge
  - Cryptographic Key (K): secret knowledge
  - Ciphertext (C): encrypted plaintext, so the message is now ‘hidden’
  - Decryption (D): transformation of the ciphertext back to the original plaintext, using an algorithm and the key

What is cryptography?

Example from Classical Cryptography: *Caesar cipher*

Plaintext: **how are you today**

encrypt by shifting characters
k places forward in alphabet

Ciphertext: **krz duh brx wrgdb**

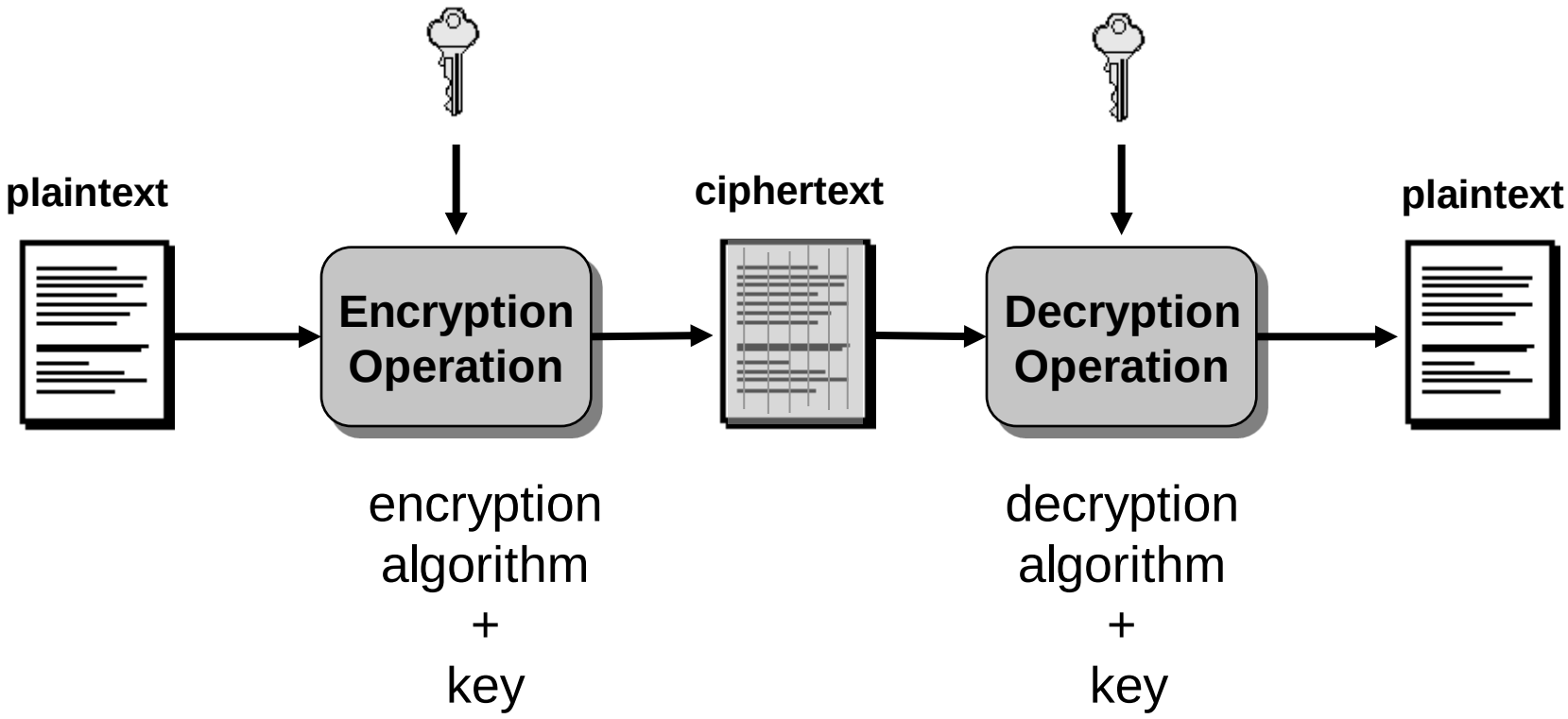
decrypt by shifting characters
k places back again

Recovered Plaintext: **how are you today**

What is cryptography?

- For the Caesar Cipher example:
  - The encryption algorithm is:
    - step forward a number of places, *k*, in the alphabet.
  - The decryption function is:
    - step back *k* places in the alphabet
  - The key is the value of *k*.
  - Caesar used this cipher with *k* = 3, therefore the name.
  - This is an insecure cipher. Modern ciphers are much more complex than this!

What is cryptography?
Basic cryptographic system:



What is cryptography?

- Encoding and Encryption:
  - Many authors incorrectly use the terms encoding and encryption interchangeably. To be strictly correct,
  - Encoding is the transformation of data from one form to another using *an encoding algorithm*.
    - Anyone who knows the corresponding decoding algorithm can decode the data.
    - For example: binary, hex, ZIP, ASCII
  - Encryption is the transformation of data from one form to another using *an encryption algorithm and a secret key*.
    - You need to know the corresponding decryption algorithm and secret key to recover the data.

What is NOT cryptography?

- **Steganography:** – used to hide the existence of a message
  - Hide the information within a document or image, so that the presence of the message is not detected
- **Steganographic techniques include**
  - Using invisible ink (try writing in lemon juice)
  - Microdots
  - Character arrangement and selection
  - Hiding information, e.g. in graphics and sound files
- **Steganographic techniques do not use a secret key**

When is cryptography used?

- If you require
  - **Confidentiality:**
    - so that your data is not made available to anyone who shouldn't have access.
    - That is, protection against snoops or eavesdroppers
  - **Integrity:**
    - So you know that the message content is correct, and has not been altered, either deliberately or accidentally
  - **Authentication:**
    - So you can be sure that the message is from the place or sender it claims to be from
- Cryptography can provide these security services.

When is cryptography used?

- Some example situations:
  - **Historically**, the military and spy agencies were the main users of cryptology
    - Situation: transmitting messages over insecure channels
  - **Now**, it is used in many other areas, especially in electronic information processing and communications technologies:
    - Banking: your financial transactions, such as EFTPOS
    - Communications: your mobile phone conversations
    - Info stored in databases: hospitals, universities, etc.
- Cryptography can be used to protect information in storage or during transmission

When is cryptography used?

- Cryptographic mechanisms such as ciphers and hash functions can provide data integrity services.
- If a message is altered, the changes to a message or data file can be detected using:
  - manipulation detection codes (MDC)
    - based on (unkeyed) hash functions
  - message authentication codes (MAC)
    - based on keyed hash functions (such as HMac), or
    - Block ciphers used in suitable modes.

Is there a ‘perfect’ cipher?

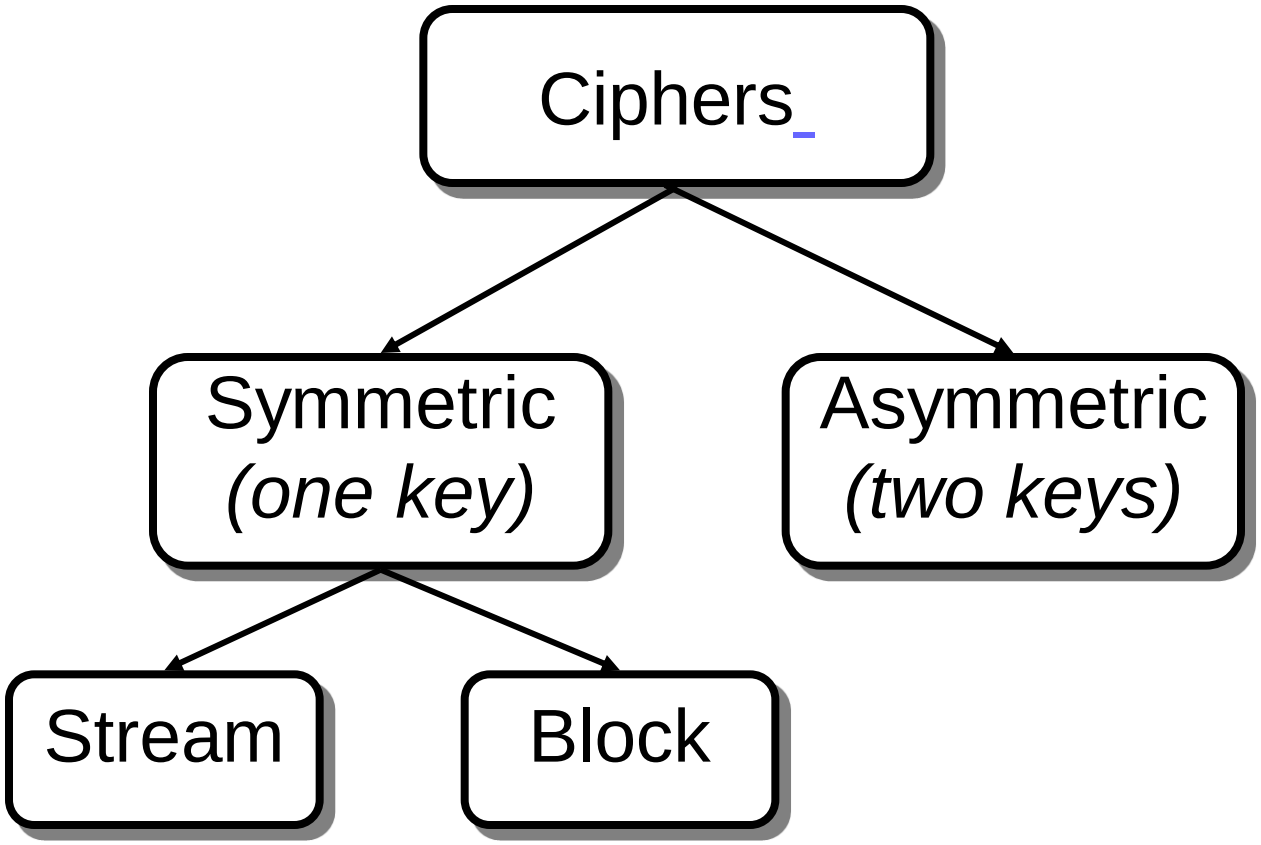
- Yes - if you require confidentiality, the One Time Pad is provably secure.
- BUT we don’t use it much.

- To understand
  - why the OTP is not widely used, and
  - how to provide other security services like integrity or authentication

  you need to know a bit more about ciphers.

- Basically, there are two types: symmetric and asymmetric.

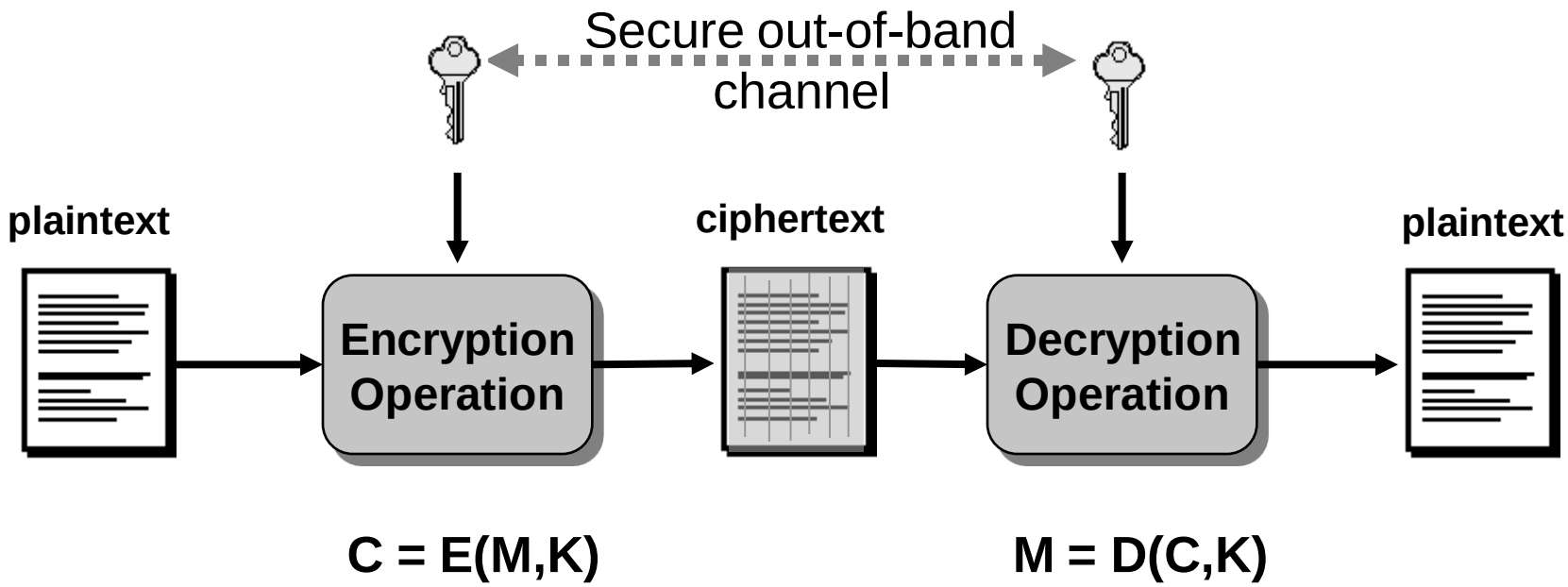
Taxonomy of modern ciphers



Symmetric ciphers

- Encryption and decryption keys are the same (or one can easily be deduced from the other)
- The encryption and decryption algorithms are usually made public.
- Notation:
  - Encryption: C = E(M,K)
  - Decryption: M = D(C,K)
- The cryptographic key K
  - must be kept secret
  - is used for both encryption and decryption, so has to be distributed or stored securely
- Two types of symmetric ciphers:
  1. Stream ciphers
  2. Block ciphers

Symmetric ciphers: Operation



Stream ciphers

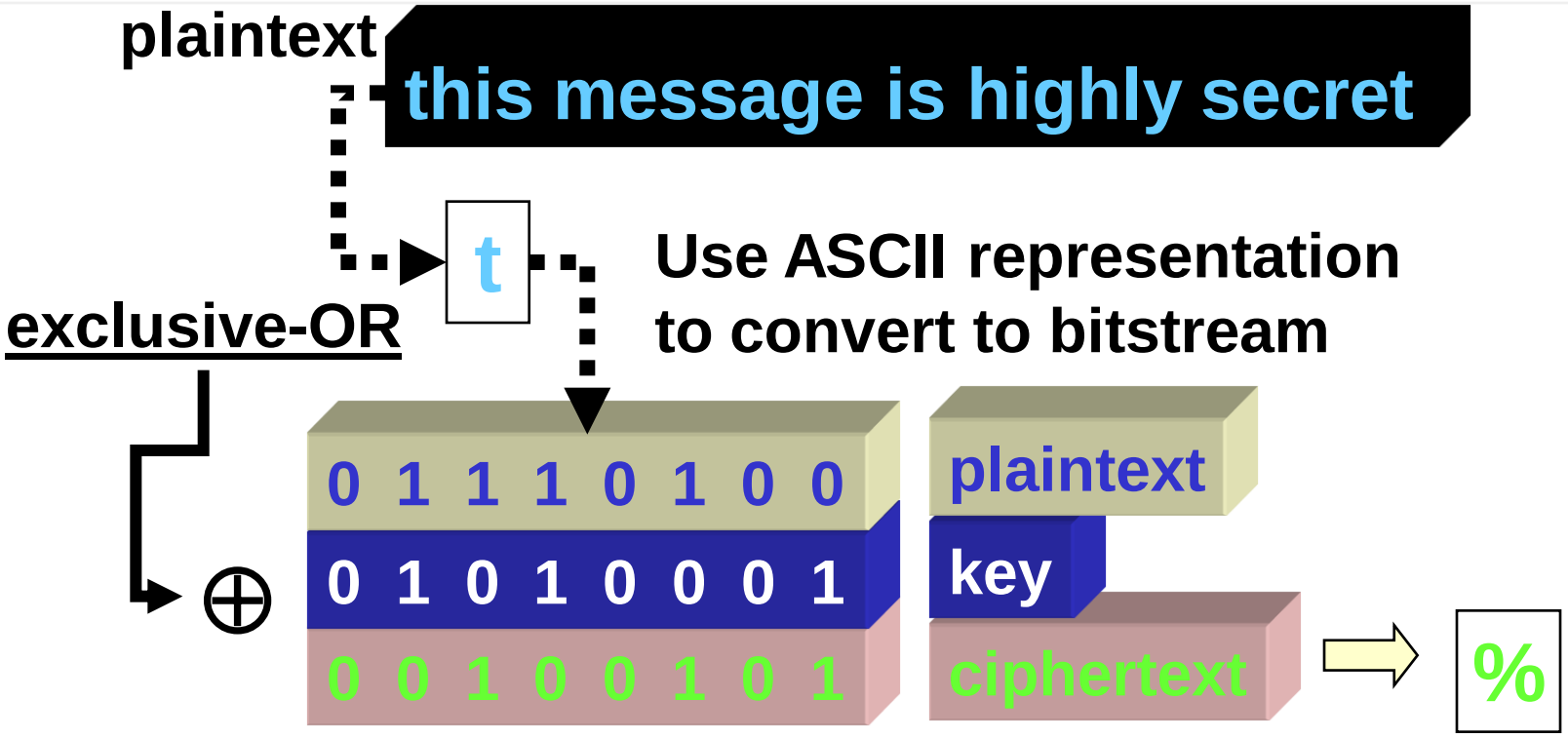
- These are symmetric ciphers
- The plaintext and ciphertext are viewed as streams of characters
  - Character size could be one bit, or an *n*-bit word e.g. 8 bit byte
- Plaintext is encrypted one character at a time, under a time-varying function of the key
- Ciphertext is decrypted one character at a time, under a time-varying function of the key
- Most common type is binary additive stream cipher

Stream ciphers

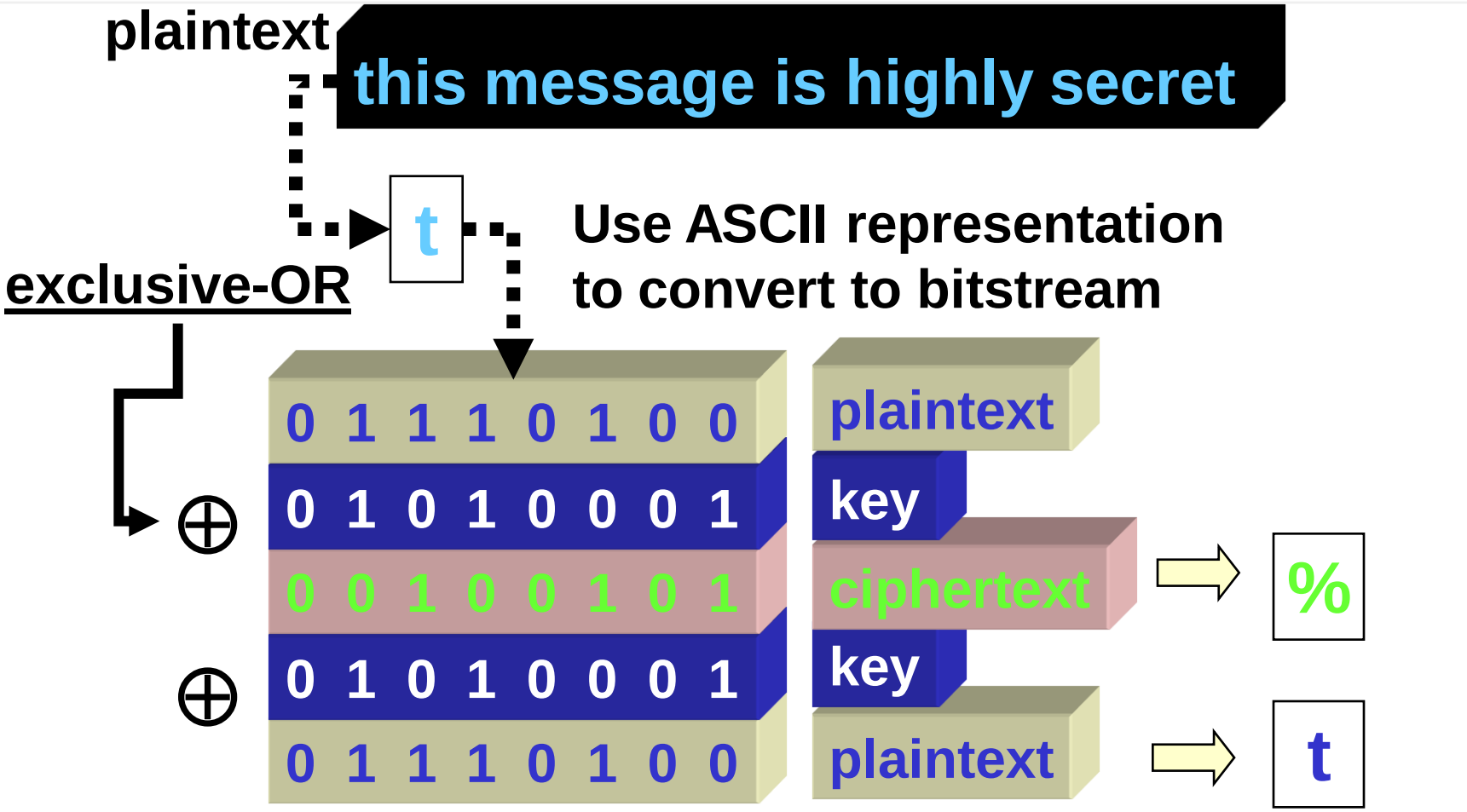
- Famous example: *Vernam one-time pad*
  - One-time pad is the only provably secure cipher
  - Vernam OTP:
    - Plaintext is a stream of bits
    - Key is a truly random binary sequence same length as message
    - The cipher is a binary additive stream cipher, so
      - Ciphertext is obtained by binary addition (XOR) of plaintext and key
      - Plaintext is recovered by binary addition (XOR) of ciphertext and key
  - NOTE: Key can be used once only (hence the name), so each message requires a new, truly random key

Stream ciphers

Binary additive stream cipher example: encryption



Stream ciphers
Binary additive stream cipher example: decryption



Stream ciphers

- Example: *Vernam one-time pad continued*
  - This is provably secure:
    - if an attacker intercepts the ciphertext, they can try all possible keystreams, and recover all possible plaintexts, but they will have no way of determining which of these was correct
  - In our example:
    - the keystream required is a truly random binary sequence 29x8=232 bits long
    - There are 2^{232} possible keystreams, and since it is a truly random sequence each sequence is equally likely
    - For a given 232-bit ciphertext, there will be 2^{232} possible plaintexts: that is, every plaintext is possible
  - The attacker gains no information about the plaintext from the intercepted ciphertext

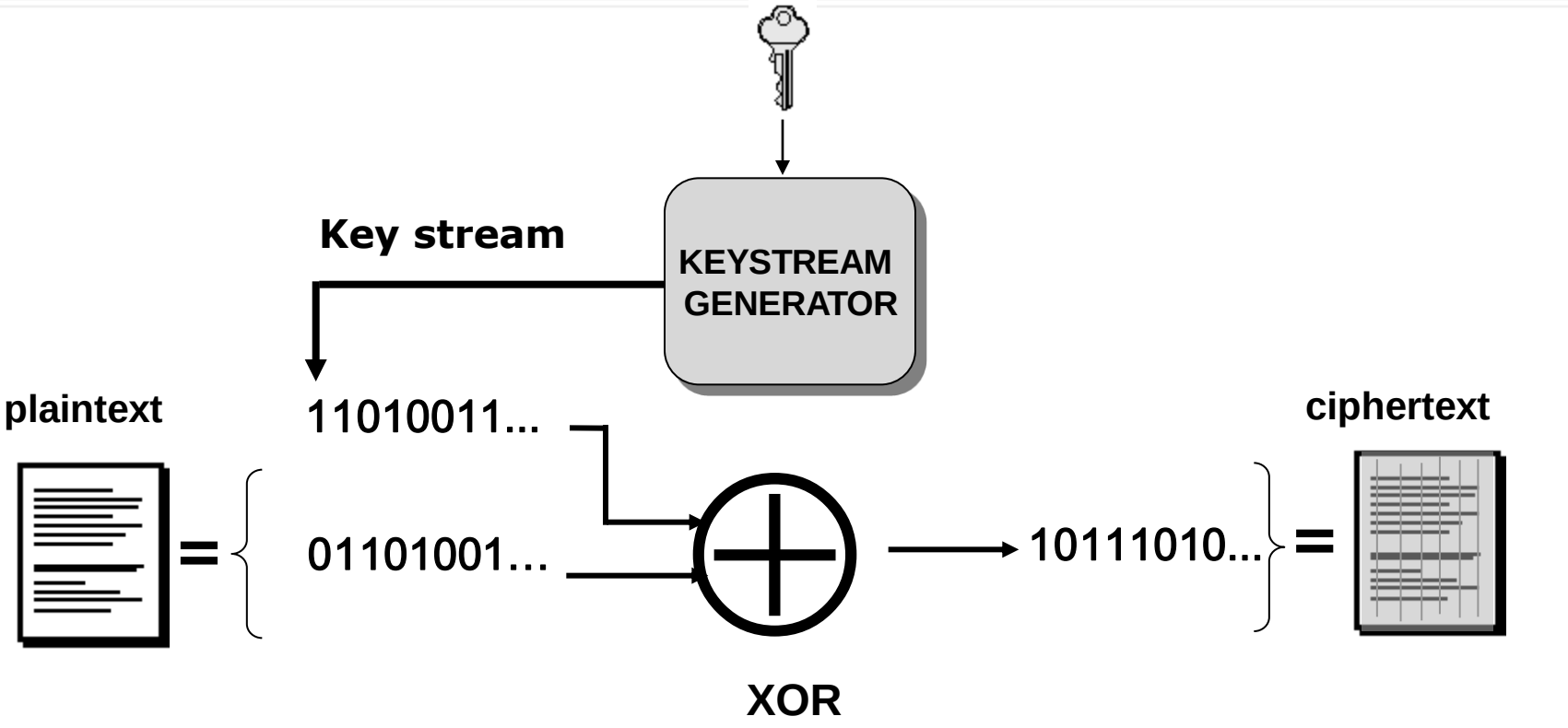
Stream ciphers

- Example: *Vernam one-time pad continued*
  - Q: That's fantastic – why don't we just use OTP?
  - A: There's a practical problem: the same keystream is required for encryption and decryption of a message, and we can't reuse keys, so
    - If you are transmitting info, can you distribute the key to the receiver securely?
    - If you are storing info, can you store this key securely?
    - If you can send/store the key securely, maybe you could just send/store the message securely?
  - Major problem is key management

Stream ciphers

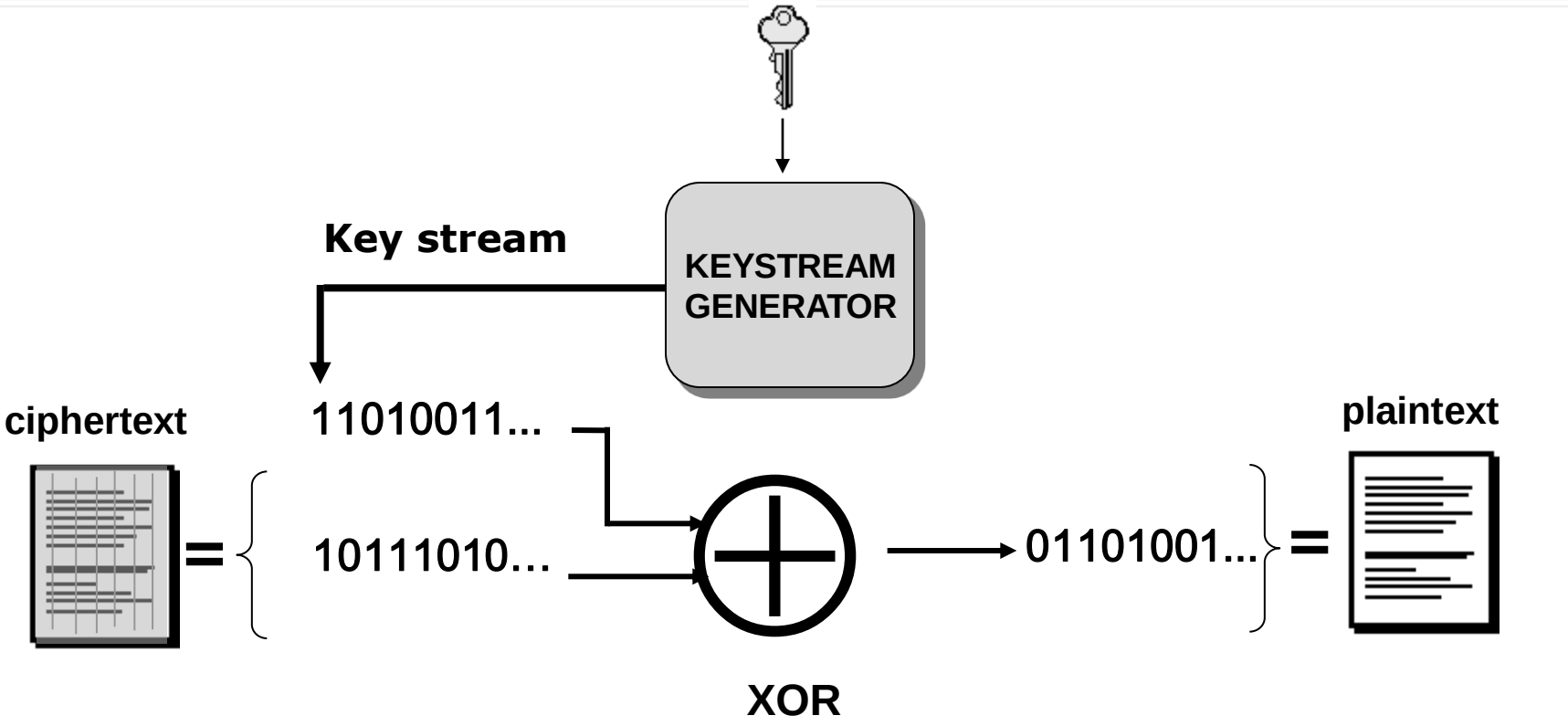
- Most stream ciphers are binary additive stream ciphers
- They try to mimic the one-time pad, BUT
  - instead of using a truly random binary sequence same length as message
  - they use a keystream generator
    - The generator input is a short binary key
    - The generator output is a longer pseudorandom binary sequence, called the keystream
    - NOTE: this is not random, it is deterministic
  - a good keystream generator should produce binary sequences that look random

Binary additive stream cipher:
Encryption operation



The key is input to the keystream generator, and the pseudorandom keystream is produced. Each plaintext bit is XOR'd with the corresponding keystream bit to produce the ciphertext bits.

Binary additive stream cipher:
Decryption operation



To recover the plaintext, use the **same key** with the **same keystream generator algorithm**, to produce the **same keystream**. Then XOR each ciphertext bit with the corresponding keystream bit to produce the plaintext bit.

Binary additive stream cipher

- Major issue: requires synchronisation
  - The keystream used to decrypt **must** be synchronised with the keystream used to encrypt: same keystream in same position
- What happens when there is an error?
  - If there is a bitflip error (0 to 1 or vice versa) only that bit will be decrypted incorrectly
  - If a ciphertext bit is inserted or deleted this causes a loss of synchronisation
    - the message can't be recovered from that point on
    - an attacker inserting or deleting data will be detected

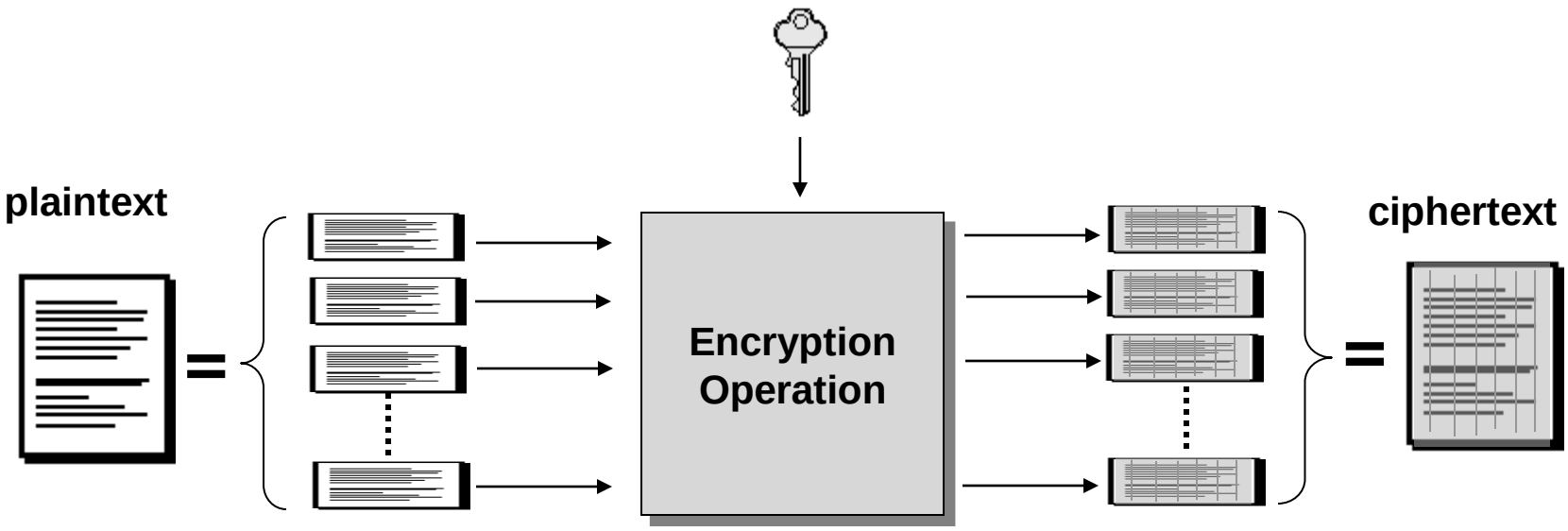
Stream cipher: Applications

- Stream ciphers are fast,
  - so they are used for real-time applications where time delays may be unacceptable.
- Application examples:
  - Communications, including internet traffic
  - Mobile telephony
  - Video (pay TV)
  - Voice scrambling, etc.

Block ciphers

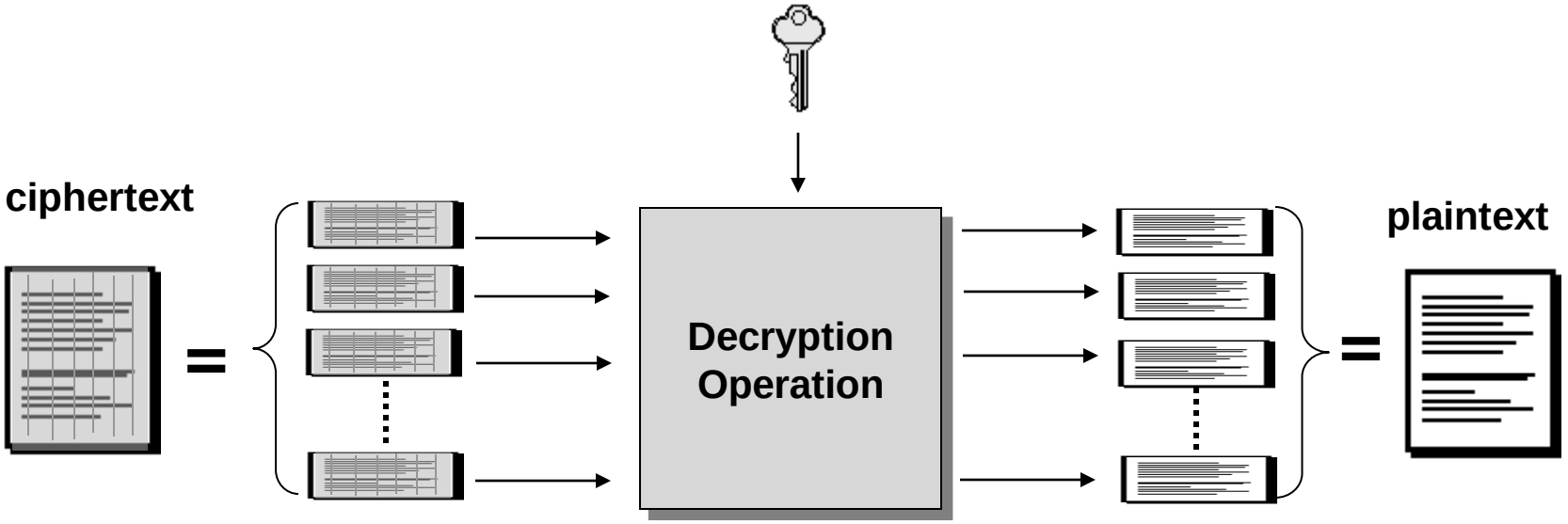
- These are symmetric ciphers
- Plaintext is encrypted one block at a time
  - Block size is usually larger than a single character, common block sizes are 64-bit or 128-bit blocks
  - In the basic block cipher mode of operation,
    - Both the plaintext block and the key are input to the encryption algorithm
    - The output of the encryption algorithm is one block of ciphertext
    - NOTE: for a fixed key, same plaintext input gives same ciphertext output
- Ciphertext is decrypted one block at a time
  - produces a block of plaintext each time

Block Ciphers:
Basic encryption operation (**E**lectronic **C**ode **B**ook mode)



The plaintext is divided into blocks $M_1M_2M_3$ .... Each block of plaintext is enciphered (using the key **K**) to produce the corresponding ciphertext block. The ciphertext consists of $C_1C_2C_3$….

Block Ciphers:
Basic decryption operation (**E**lectronic **C**ode **B**ook mode)



The ciphertext is divided into blocks $C_1C_2C_3$ .... Each block of ciphertext is deciphered (using the key **K**) to produce the corresponding plaintext block. Then the plaintext is reassembled.

Block ciphers: Example

- DES algorithm
  - **D**ata **E**ncryption **S**tandard (actually Data Encryption Algorithm – DEA)
  - Published by US National Bureau of Standards as FIPS publication 46, in 1977
    - i.e. this is a US standard, but a defacto global standard
  - Plaintext and ciphertext are 64 bit blocks
  - Key size: 56 bits
  - DES is considered insecure but is still used widely.
  - triple-DES (3DES) provides a way to extend the useful life of DES, approved by the FIPS
    - For this ANSI X9.52-1998, *Triple Data Encryption Algorithm Modes of Operation*, must be used in conjunction with FIPS 46-3.

Block ciphers: Example 2

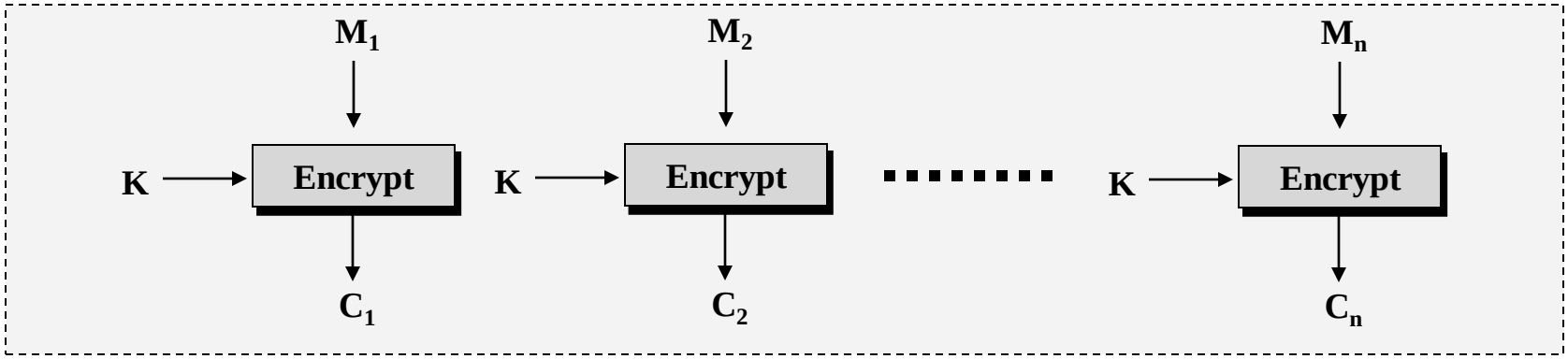
- Advanced Encryption Standard (AES)
  - AES made a call for submissions to establish a standard block cipher more secure than DES
    - The chosen cipher announced at the end of 2001
  - "Rijndael“ developed by Joan Daemen and Vincent Rijmen (Belgian cryptographers) was selected
    - Published by US National Bureau of Standards as FIPS publication 197, in 2002
  - Plaintext and Ciphertext are 128-bit blocks
  - Key size is variable: 128, 192, or 256-bit keys are allowed

Block Ciphers: Modes of Operation

- Block ciphers can be used in different modes in order to provide different security services.
- Common modes include:
  - **E**lectronic **C**ode **B**ook (ECB)
  - **C**ipher **B**lock **C**haining (CBC)
  - **O**utput **F**eed**b**ack (OFB)
  - **C**ipher **F**eed**b**ack (CFB)
  - Counter Mode

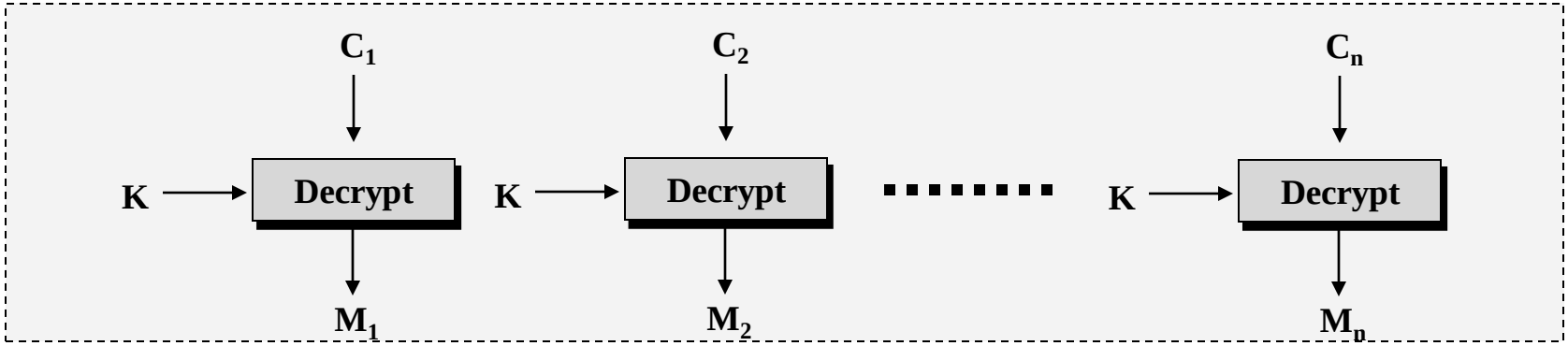
Block Ciphers

- ECB Mode encryption
  - Simplest mode of operation
  - Plaintext data is divided into blocks $M_1, M_2, \ldots, M_n$ (last block is padded as required)
  - Each block is then processed separately
    - Plaintext block and key used as inputs to the encryption algorithm
    - Called Code Book style



Block Ciphers

- ECB mode decryption
  - Ciphertext data is divided into blocks $C_1, C_2, \ldots, C_n$
  - Each block is then processed separately
    - Ciphertext block and key used as inputs to the decryption algorithm
    - Need to use same key as used for encryption (symmetric cipher)
  - Plaintext is assembled by concatenating blocks, removing padding if required



Block Ciphers

- ECB Mode Issues
  - Problem: For a given key, the same plaintext block always encrypts to the same ciphertext block.
    - This may allow an attacker to construct a code book of known plaintext/ciphertext blocks.
    - The attacker could use this codebook to insert, delete, reorder or replay data blocks within the data stream without detection
  - Other modes of operation can prevent this, by not encrypting blocks independently
    - For example, using the output of one block encryption as input to the next (chaining)

Block Ciphers

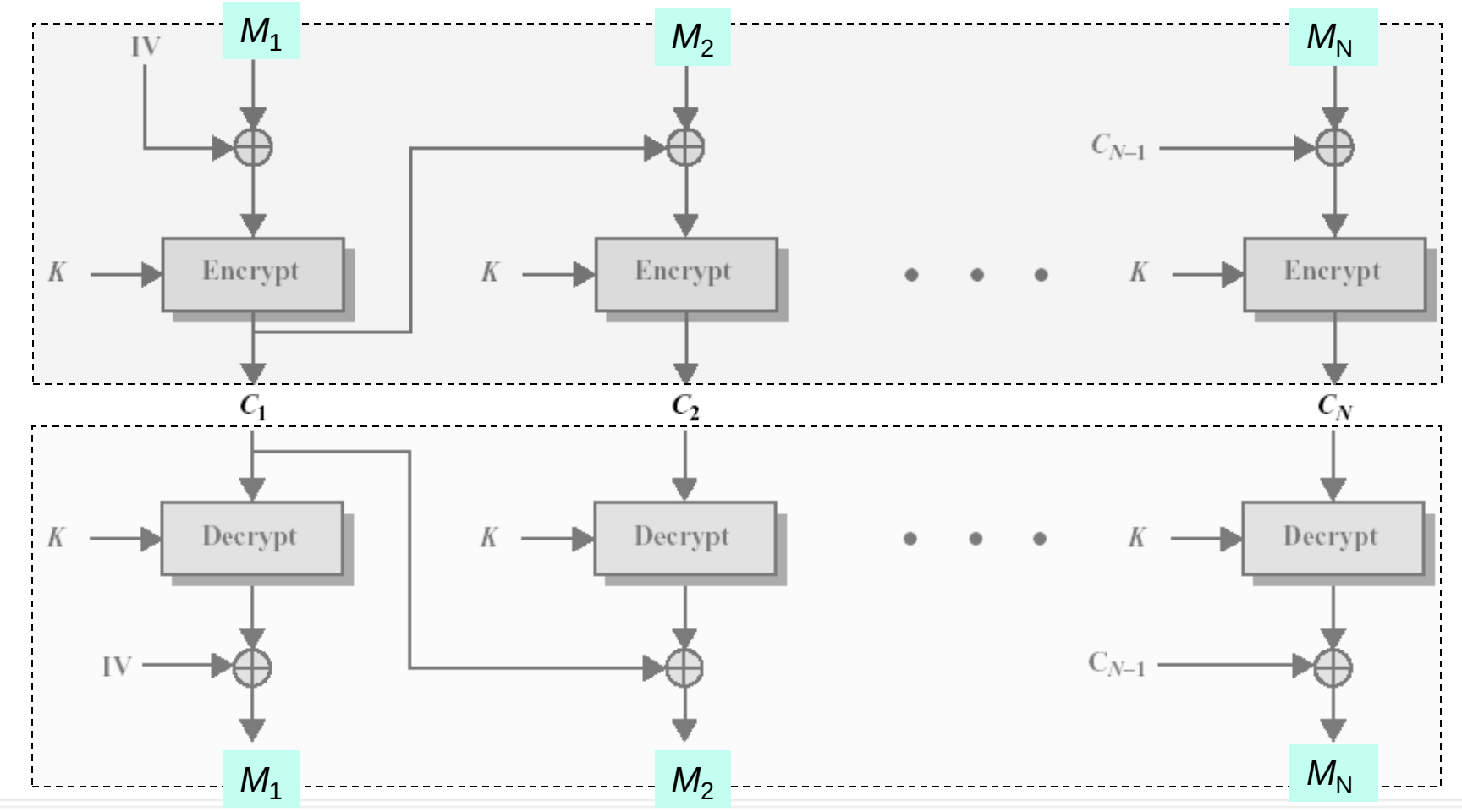
- Cipher Block Chaining (CBC) Mode Encryption
  - Plaintext data is divided into blocks $M_1, M_2, \ldots, M_n$ (last block is padded as necessary)
  - The encryption algorithm now takes **three** inputs:
    - the key, the plaintext, and the previous ciphertext
    - For the first plaintext block, an *initialisation vector (IV)* is used to ensure that two encryptions of the same plaintext will result in different ciphertext.
  - Encryption:

$$C_1 = E(M_1 \oplus IV, K)$$
$$C_i = E(M_i \oplus C_{i-1}, K)$$

Block Ciphers

- CBC Mode Decryption
  - Ciphertext is divided into blocks $C_1, C_2, \ldots, C_n$
  - Decryption:
    $M_1 = \mathrm{D}(C_1, K) \oplus IV$
    $M_i = \mathrm{D}(C_i, K) \oplus C_{i-1}$
  - where
    - K is the same key used during encryption, and
    - IV is the same initialisation vector used during encryption.

Block Ciphers: CBC Mode



Block Ciphers

- CBC Mode Issues
  - Chaining guards against the construction of a code book
    - The same plaintext block encrypts to different ciphertext blocks each time.
  - May assist in detecting integrity breaches
    - Such as the insertion, deletion or reordering of data blocks into the ciphertext.
- What happens when there is an error?
  - If there is a bitflip error (0 to 1 or vice versa) that block and the following block will be decrypted incorrectly
  - If a ciphertext bit, or even a character is inserted or deleted this will be detected because of the incorrect ciphertext length
    - Not multiples of block size
  - Inserting or deleting a block will cause incorrect decryption

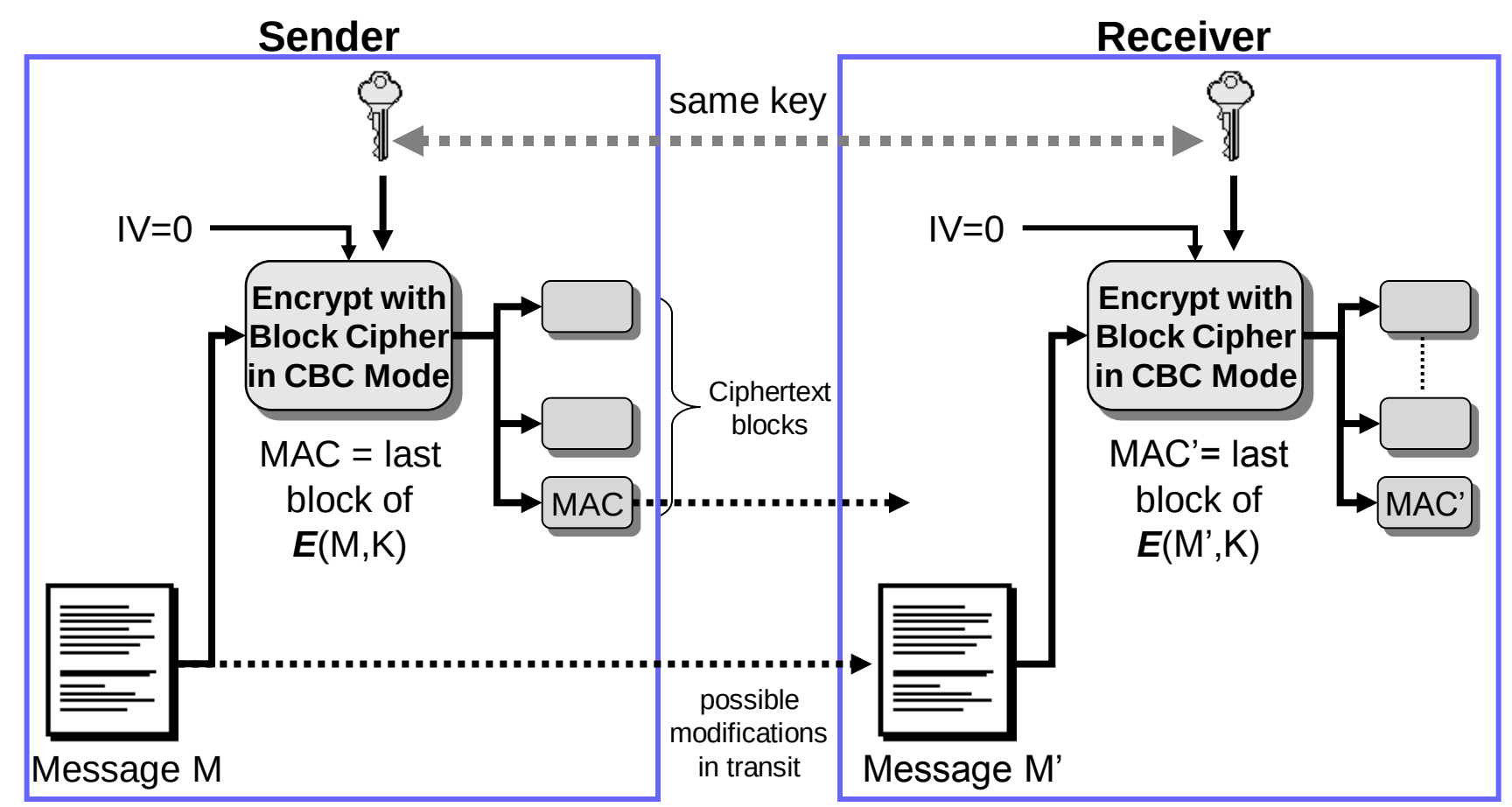
Block cipher: Applications

- Block ciphers are often used for providing confidentiality services
- They are used for applications involving processing large volumes of data, where time delays are not critical.
  - Examples:
    - Computer files
    - Databases
    - Email messages
- Block ciphers can also be used to provide integrity services

Block Ciphers and Message Integrity

- A block cipher used in CBC mode can be used specifically to provide integrity services, rather than confidentiality
  - For a given message or data file,
    - The file is encrypted using the block cipher in CBC mode
    - The last ciphertext block is used as a Message Authentication Code (MAC) value
    - Both the message and the MAC are sent to the receiver
  - Usually referred to as CBC-MAC
  - Described in ISO/IEC 9797-1:1999
  - Considered secure for messages of a pre-selected fixed length

Block Ciphers and Message Integrity



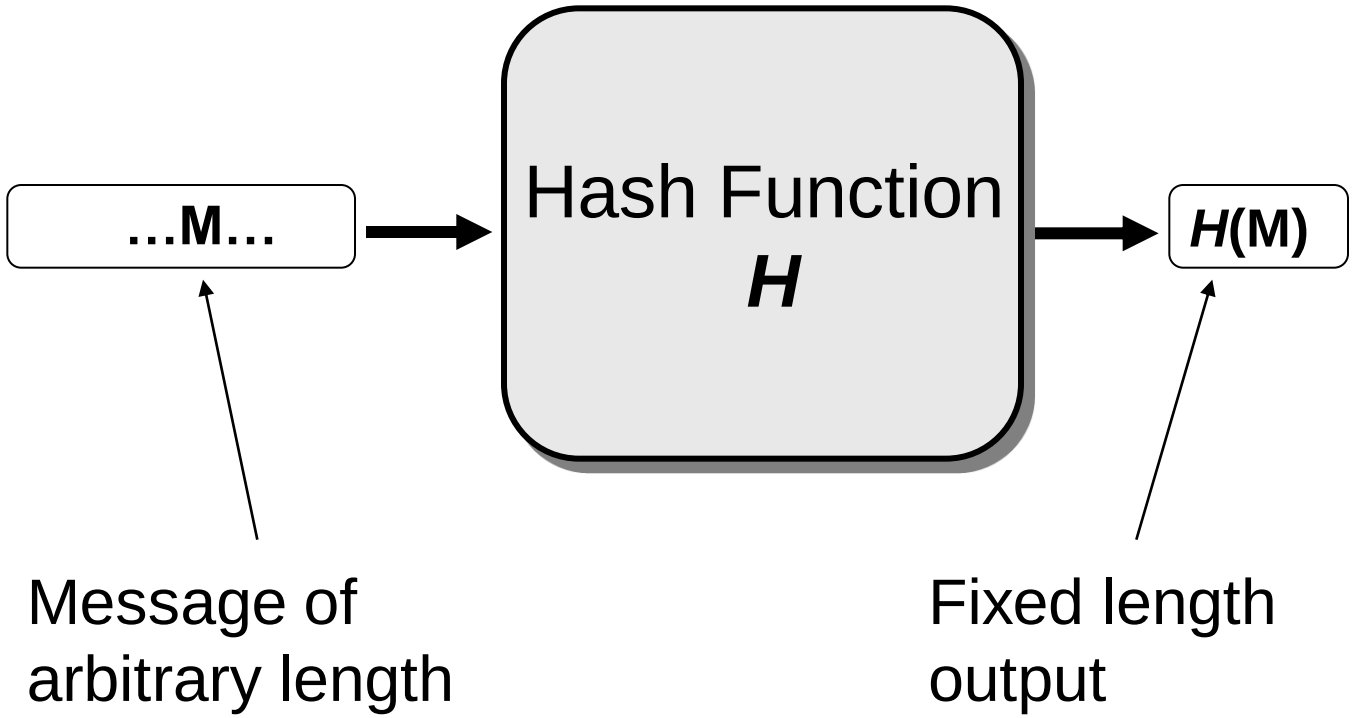
Block Ciphers and Message Integrity

- Using a block cipher in CBC mode to provide integrity services during transmission:
  - **Sender**:
    - generates message M
    - encrypts message M using block cipher in CBC mode, using shared secret key K
    - MAC is last ciphertext block
    - sends M and MAC to Receiver
  - **Receiver:**
    - Receives {M', MAC}
    - generates MAC' from M'
    - compares MAC' and MAC
    - If MAC' = MAC knows that message is unaltered

Hash functions

- A hash function ***H*** is a transformation:
  - which takes as input a message M of arbitrary length
  - and produces as output the fixed-length value ***H***(M)
- ***H***(M) is called the **hash value**
  - (or message digest, or checksum)
- Widely used MD4, MD5 have been broken
  - MD4 and MD5 produce 128-bit hash values
- SHA-1 has vulnerabilities
  - SHA-1 produces a 160-bit hash value
- SHA-224, 256, 384, 512 bit are still OK

Hash function: Operation



Hash functions

- Four basic properties:
  - H1: Fixed length output ***H***(M), for arbitrary length input M
  - H2: ***H***(M) is one-way:
    - given M, it is easy to compute ***H***(M), but
    - given ***H***(M), it is computationally infeasible to compute M
  - H3: ***H***(M) is collision-resistant:
    - Hard to find two messages M and M' so that ***H***(M) = ***H***(M')
  - H4: If you make a small change in M, it produces a major change in ***H***(M)

Hash functions: Application

- Hash functions are often used for providing security services related to integrity
- Examples:
  - computing ‘checksums’ for data files
  - message integrity codes to detect modifications to data
  - as part of the digital signature mechanism
    - More on digital signatures in lecture on authentication

Hash functions: Application

- Using a hash function to check message integrity:
- Alice:
  - generates message M
  - calculates ***H***(M)
  - sends {M, ***H***(M)} to Bob.
- Bob:
  - receives {M', ***H***(M)},
  - calculates ***H***(M'),
  - compares ***H***(M) and ***H***(M')
  - If ***H***(M) = ***H***(M'), then the message he received M' is assumed to be the message M Alice sent: unaltered

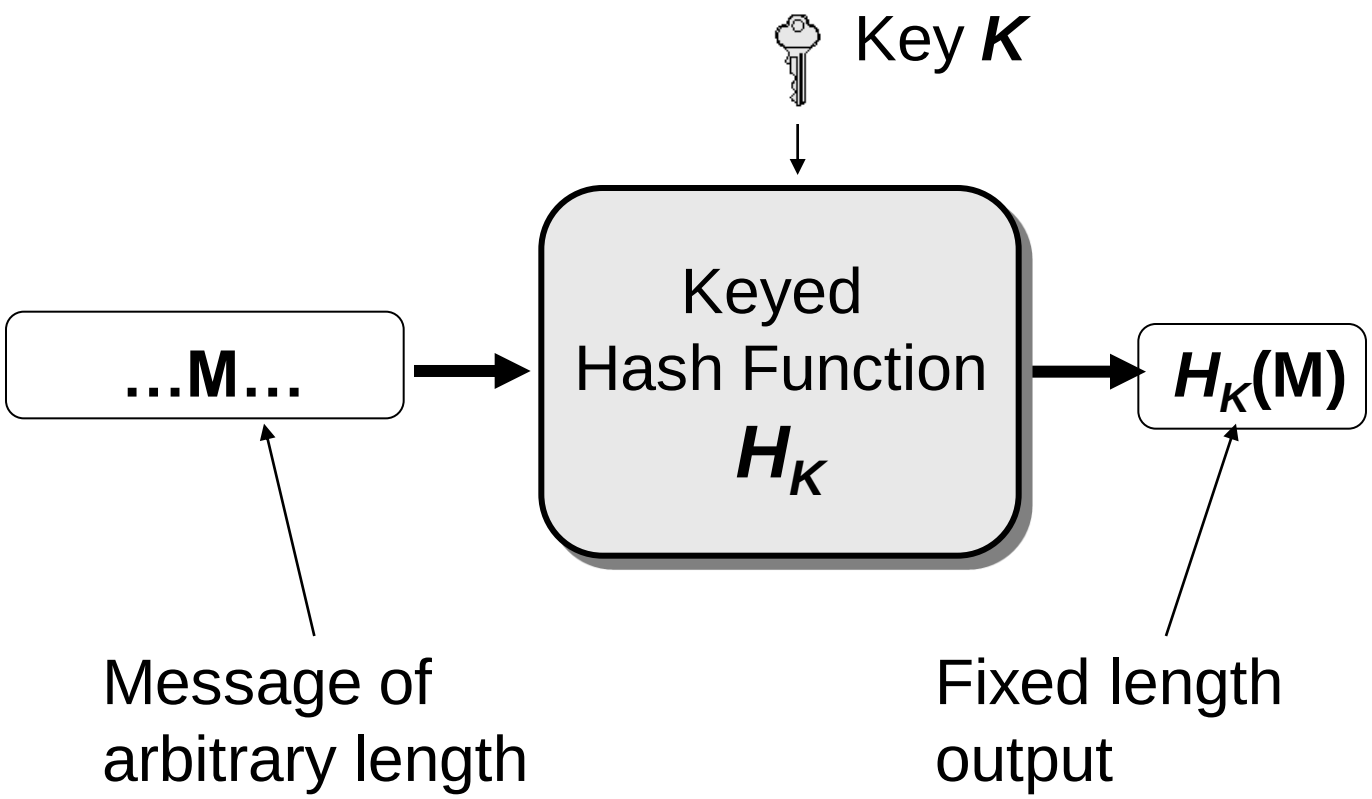
Hash functions:

- Problems for dedicated One-way Hash functions:
  - Useful for detecting accidental modifications
    - For example, transmission errors
  - Not useful for providing integrity services against active attackers.
  - Example: suppose in our transmission from Alice to Bob, Carol
    - intercepts the message {M, ***H***(M)} sent by Alice,
    - changes M to M',
    - recalculates the hash ***H***(M')
    - and sends {M', ***H***(M')} to Bob
  - If Bob relies on the hash function providing integrity protection, he will mistakenly think M' is the unaltered message from Alice!

HMAC: Keyed hash functions

- A solution: Keyed hash functions:
  - $\boldsymbol{H_K}$(M) has two inputs:
    - message M
    - cryptographic key
  - key protects against unauthorized modification of the hash value
    - Calculation of $\boldsymbol{H_K}$(M) value requires knowledge of secret key.
    - Attacker can only successfully modify information and update the $\boldsymbol{H_K}$(M) value appropriately if they know the key.
  - also called cryptographic checksums or message authentication codes (MAC)

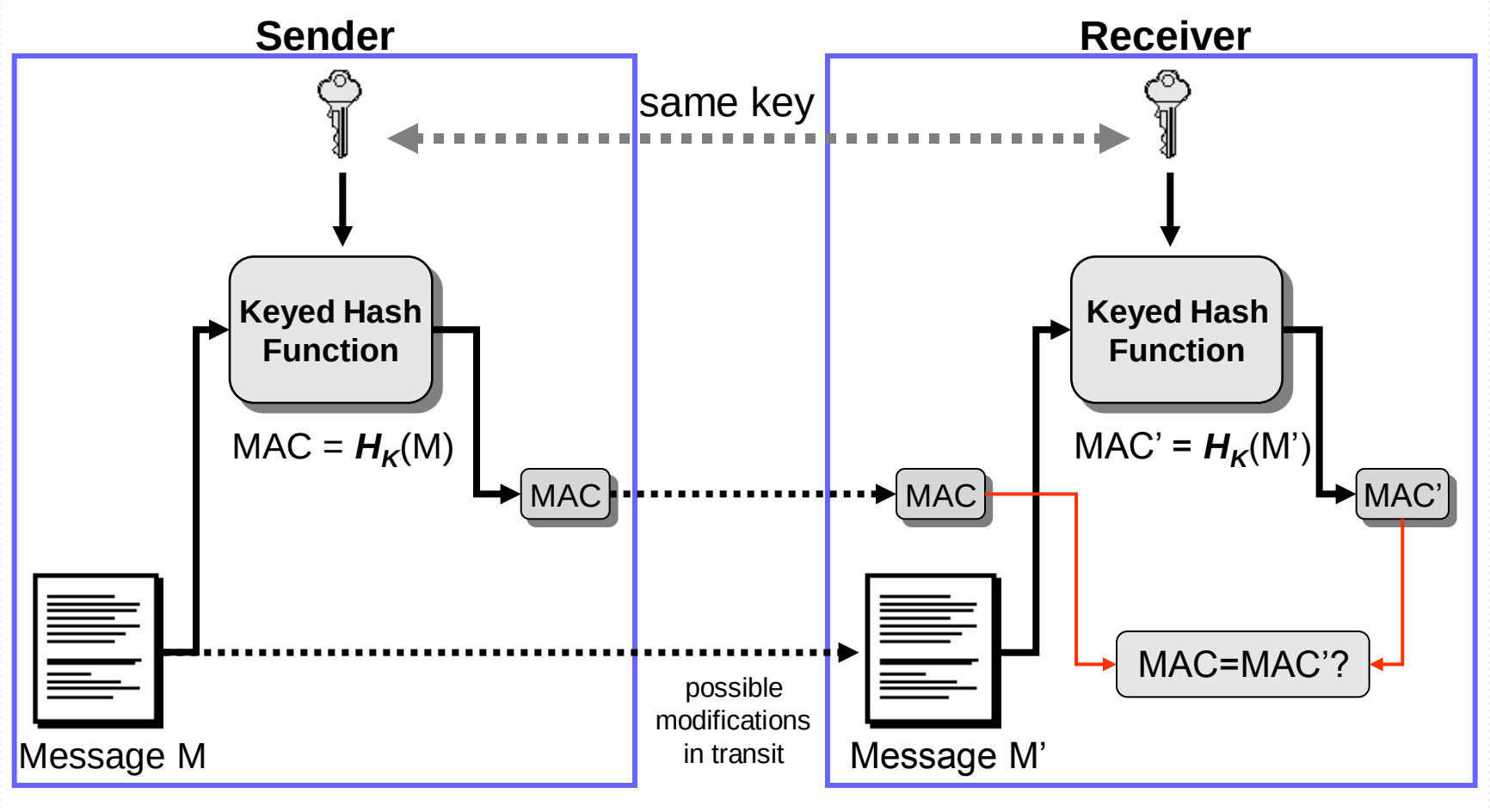
HMAC: Keyed hash functions



HMAC: Keyed hash functions

- Using a keyed hash function to provide integrity services during transmission:
  - **Sender:**
    - generates message M,
    - generates MAC = $\boldsymbol{H_K}$(M) from M, using K
    - sends {M, MAC} to Receiver
  - **Receiver:**
    - receives {M', MAC},
    - generates MAC' from M', using K (shared secret key)
    - compares MAC' and MAC
    - If MAC' = MAC knows that message is unaltered

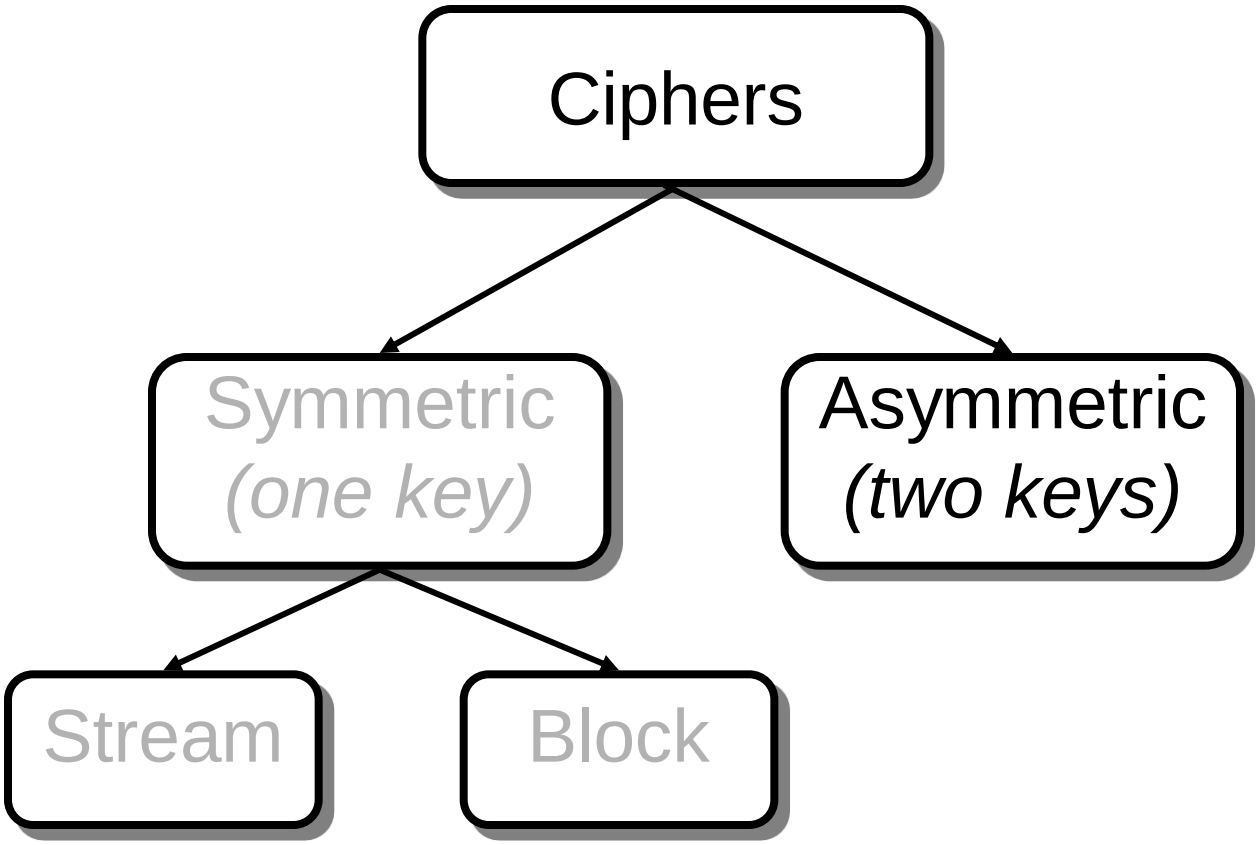
Hash functions: Keyed hash functions



Hash functions and Message Authentication

- A shared secret key is used for both a HMAC and CBC-MAC
- When used during message transmission, this provides an additional security service of Message Authentication:
  - A correct MAC value confirms the sender of the message is in possession of the shared secret key
  - Hence, much like a password, it confirms the authenticity of the message sender to the receiver.
- Indeed, message integrity is meaningless without knowing who sent the message.

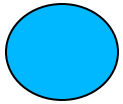
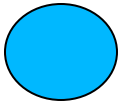
Asymmetric ciphers

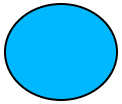
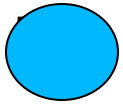


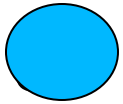
Mathematical Preliminaries: Modular arithmetic

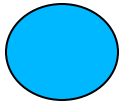
- Most asymmetric cryptosystems makes use of modular arithmetic.
- Modular arithmetic is a form of integer arithmetic.
  - written using the mod operator: a + b mod n, etc …
  - numbers "wrap around" after they reach a certain value called the modulus.
  - often referred to as clock arithmetic.
  - digital computers perform modular arithmetic.
    - In Java, % is the modulo operator. 12 % 5 = 2
    - In MS Excel, there is a mod function.

Modular addition and multiplication

- 5 + 7 = 
- (5 + 7) mod 10 = 

- 5 * 7 = 
- (5 * 7) mod 10 = 

- (12 + 13) mod 11 = 

- (12 * 13) mod 11 = 

Diffie-Hellman key agreement

- Diffie-Hellman key agreement algorithm achieves key agreement; it is not encryption algorithm.
- It allows two hosts to create a shared secret.
  - Diffie and Hellman published the protocol in 1976.
  - At the time, this was a radically new method of distributing cryptographic keys.
  - The article inspired the development of a new class of enciphering algorithms - asymmetric ciphers.
  - Previously discovered by Malcolm Williamson (GCHQ in the UK) in 1973, but GCHQ chose not to make it public until 1997.

Diffie-Hellman key agreement

- Without prior arrangement, two parties can agree on a shared secret known only to them.
  - An eavesdropper cannot determine the shared secret by listening to the messages exchanged by the two parties.
- Setup:
  - Two system parameters p and g.
  - p is a large prime number.
  - g is an integer less than p such that, for every n between 1 and p-1, there is a k such that $n = g^k \bmod p$

Diffie-Hellman key agreement

Alice picks random integer a

Bob picks random integer b

$g^a \bmod p$

$g^b \bmod p$

Alice computes the shared secret
$(g^b)^a = g^{ab} \bmod p$

Bob computes the same secret
$(g^a)^b = g^{ab} \bmod p$.

Diffie-Hellman example (small numbers only)

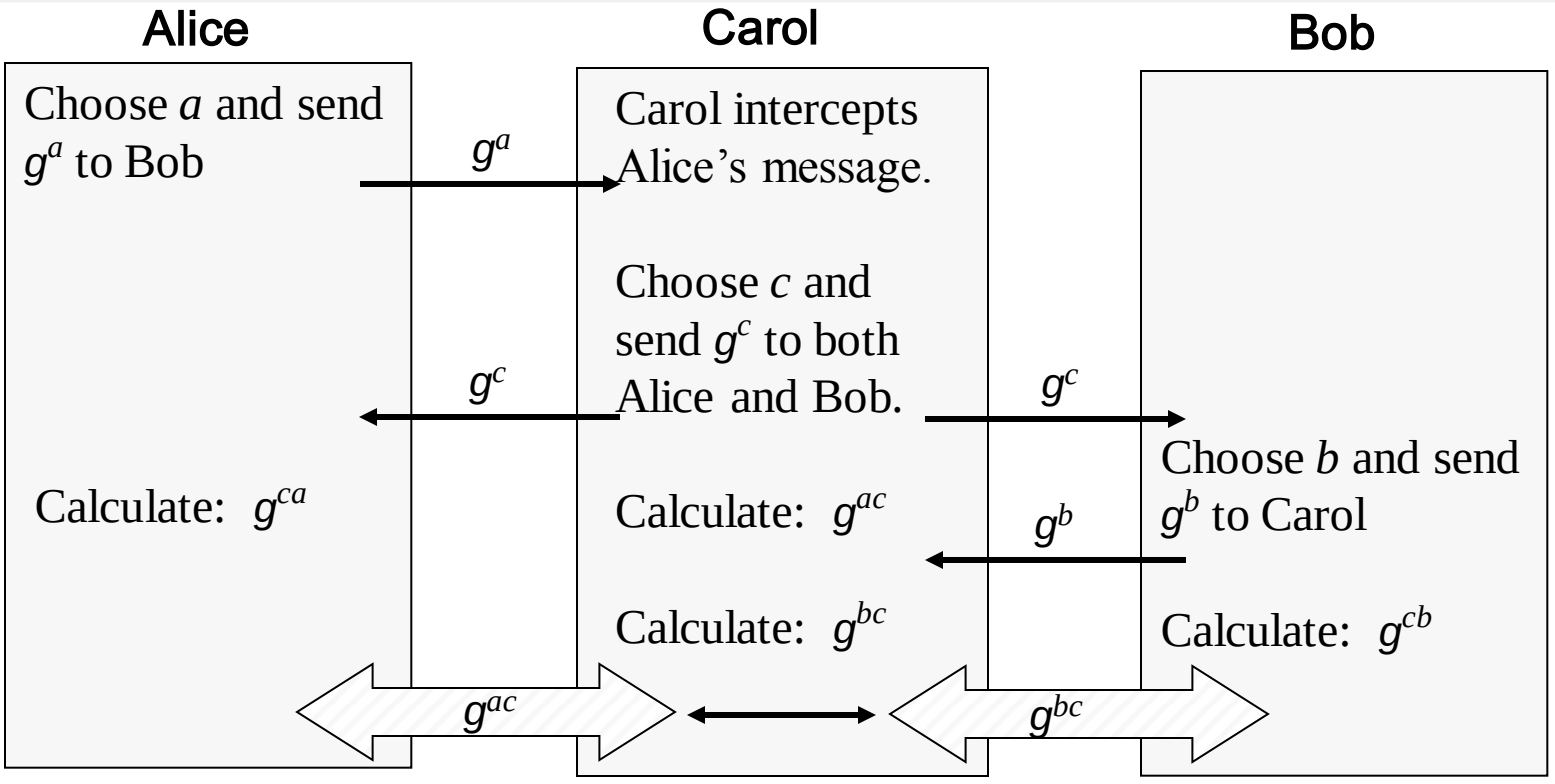
- Setup:
  - Alice and Bob agree on using $p = 71$ and $g = 7$.
- Protocol:
  - Alice selects a random integer, say 5, and sends to $7^5 \bmod 71 = 51$ to Bob.
  - Bob selects a random integer, say 12, and sends to $7^{12} \bmod 71 = 4$ to Alice.
  - Alice computes $4^5 \bmod 71 = 30$
  - Bob computes $51^{12} \bmod 71 = 30$

Diffie-Hellman example (small numbers only)

- For this small example, an attacker who captures 51 and 4 can construct a small table of values as shown here and determine the values for *a* and *b*.
- In this case, *a=5* and *b=12*.
- For large values of *p*, constructing such a table is computationally too difficult.
- The problem of determining *a* given g^a is referred to as the discrete log problem.

i	g^i	i	g^i
0	1	9	47
1	7	10	45
2	49	11	31
3	59	12	4
4	58	13	28
5	51	14	54
6	2	15	23
7	14	16	19
8	27	17	62

Man-in-the-Middle with Diffie-Hellman



Carol now shares keys with both Alice and Bob. Alice sends confidential messages believing that she is communicating with Bob. Carol can decrypt these messages, read them and re-encrypt the message for Bob and send it on. Likewise, Bob sends messages to Alice unaware of Carol.

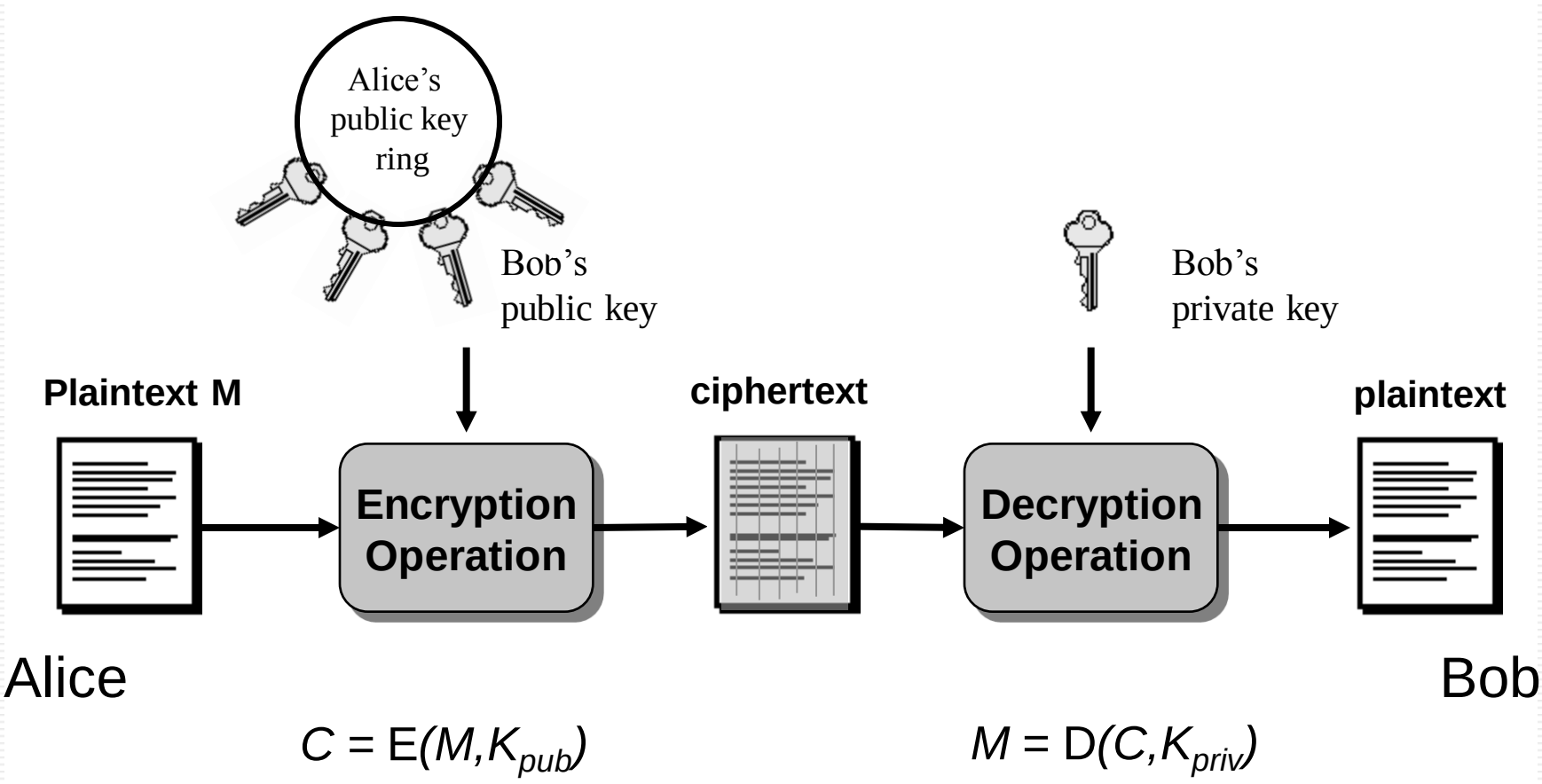
Man-in-the-Middle and Diffie-Hellman

- Basic problem is lack of authentication
- Alice and Bob have no idea who is their communication partner
- Some authentication information must be added such as a digital signature.

Diffie-Hellman Applications

- IPSec (IP Security)
  - IKE (Internet Key Exchange) is part of the IPSec protocol suite
  - IKE is based on Diffie-Hellman Key Agreement
- SSL/TLS
  - Several variations of SSL/TLS protocol including
    - Fixed Diffie-Hellman
    - Ephemeral Diffie-Hellman
    - Anonymous Diffie-Hellman

Asymmetric Encryption: Basic encryption operation



Asymmetric Ciphers: Basic encryption operation

- Encryption: $C = E(M, K_{pub})$
- Decryption: $M = D(C, K_{priv})$
- The public key of the recipient is used for encryption.
- The private key of the recipient is used for decryption.
- Only the private key must remain confidential (i.e. known only to its owner).
  - Anyone is allowed to know the public key.
  - Only the owner may know the associated private key.
- Key distribution is seemingly simple
  - But how do you know that you have the right public key?
  - Will return to this problem in Key Management lecture

Asymmetric Ciphers: Key Establishment Problem

- Key distribution is much simpler as anyone may know the public key.
- If there are *n* participants, then a total of *n* key pairs must be created.
- Each party need only create an asymmetric key pair and publish the public key.
- The problem now is how to ‘publish’ the public key. We will return to this problem later.

Asymmetric Ciphers: Examples of Cryptosystems

- RSA: best known asymmetric algorithm.
  - RSA = Rivest, Shamir, and Adleman (published 1977)
  - Historical Note: U.K. cryptographer Clifford Cocks invented the same algorithm in 1973, but didn't publish.
- ElGamal Cryptosystem
  - Based on the difficulty of solving the discrete log problem.
- Elliptic Curve Cryptography
  - Based on the difficulty of solving the EC discrete log problem.
  - Provides same level of security with smaller key sizes.

ElGamal Encryption

- Published in 1985
- Relies on difficulty of discrete logarithms for security
- Ciphertext is twice the length of plaintext
- Ciphertext is randomised - same plaintext always has different ciphertext

ElGamal Encryption

- System parameters p and g same as for Diffie-Hellman
- Alice picks a random integer a as her fixed private key and publishes her public key $g^a \bmod p$.
- To encrypt a message m to Alice, Bob picks a random integer r , computes $(g^a)^r = g^{ar} \bmod p$ and sends the ciphertext $(g^r \bmod p, m \times g^{ar} \bmod p)$ to Alice.
- Alice computes the shared secret $(g^r)^a = g^{ar} \bmod p$ and recovers $m = (m \times g^{ar} \bmod p) / (g^{ar} \bmod p)$

Asymmetric Ciphers: RSA Cryptosystem

- Published by Rivest-Shamir-Adleman, MIT, 1977.
- Public-key cryptosystem and digital signature scheme.
- Based on difficulty of factorising large integers.
  - To be precise, it is no harder to break than factorising
- Calculations are performed using modular arithmetic.
- Previously discovered by Clifford Cocks (GCHQ in the UK) in 1973, but GCHQ (equivalent to the NSA in USA) chose not to make it public until 1997.

Asymmetric Ciphers : RSA Key Generation (corrected)

- Choose two secret large primes ***p*** and ***q***.
  - Calculate: ***n* = *pq*** which can be made public (not ***p*** and ***q***)
  - Calculate secret: **z = (p-1)(q-1)**
- Choose a public exponent ***e***
- Calculate the private exponent ***d*** such that ***ed* = 1 mod(*z*)**
  - requires using the secret knowledge of ***z*** (i.e. of primes ***p*** and ***q)***.
  - Anyone may have access to the public key **(*n*, *e*)**
  - The private key ***d*** must be kept confidential by the key owner.
  - Primes ***p*** and ***q*** must be kept confidential or destroyed
- The security of RSA rests on the difficulty of factorizing ***n***
  - so the prime factors ***p*** and ***q*** must be LARGE

Asymmetric Ciphers : RSA Encryption/Decryption

- Encryption function of message *m* (1 < m < n).
  - Calculate: $\boldsymbol{c = m^e \bmod n}$
- Decryption function of ciphertext *c*
  - Calculate: $\boldsymbol{m = c^d \bmod n}$

Asymmetric Ciphers : RSA "Raw" Example

- Key Generation:
  - Choose $p = 43$, $q = 59$
  - $n = pq = 2537$
  - Choose $e = 5$
  - Calculate d = 1949
- Encryption:
  - Let m = 50
  - $c = m^5 = 2488 \bmod 2537$
- Decryption:
  - $m = c^{1949} = 50 \bmod 2537$

Asymmetric Ciphers: RSA Formatting

- In practice messages are "pre-processed" before RSA encryption is applied
- Pre-processing involves the application of any of a number of formalised RSA formatting standards - padding, etc
- Formatting overcomes known problems with "raw" RSA encryption

Elliptic Curve Cryptography

- First proposed in 1985
- Uses algebraic group defined on set of points on an elliptic curve
- Any cryptosystem based on discrete logarithm problem (such as ElGamal) can work on elliptic curve
- Commercialised, particularly by Certicom, in 1990s
- Small key size and small ciphertext than RSA

Key length comparison:

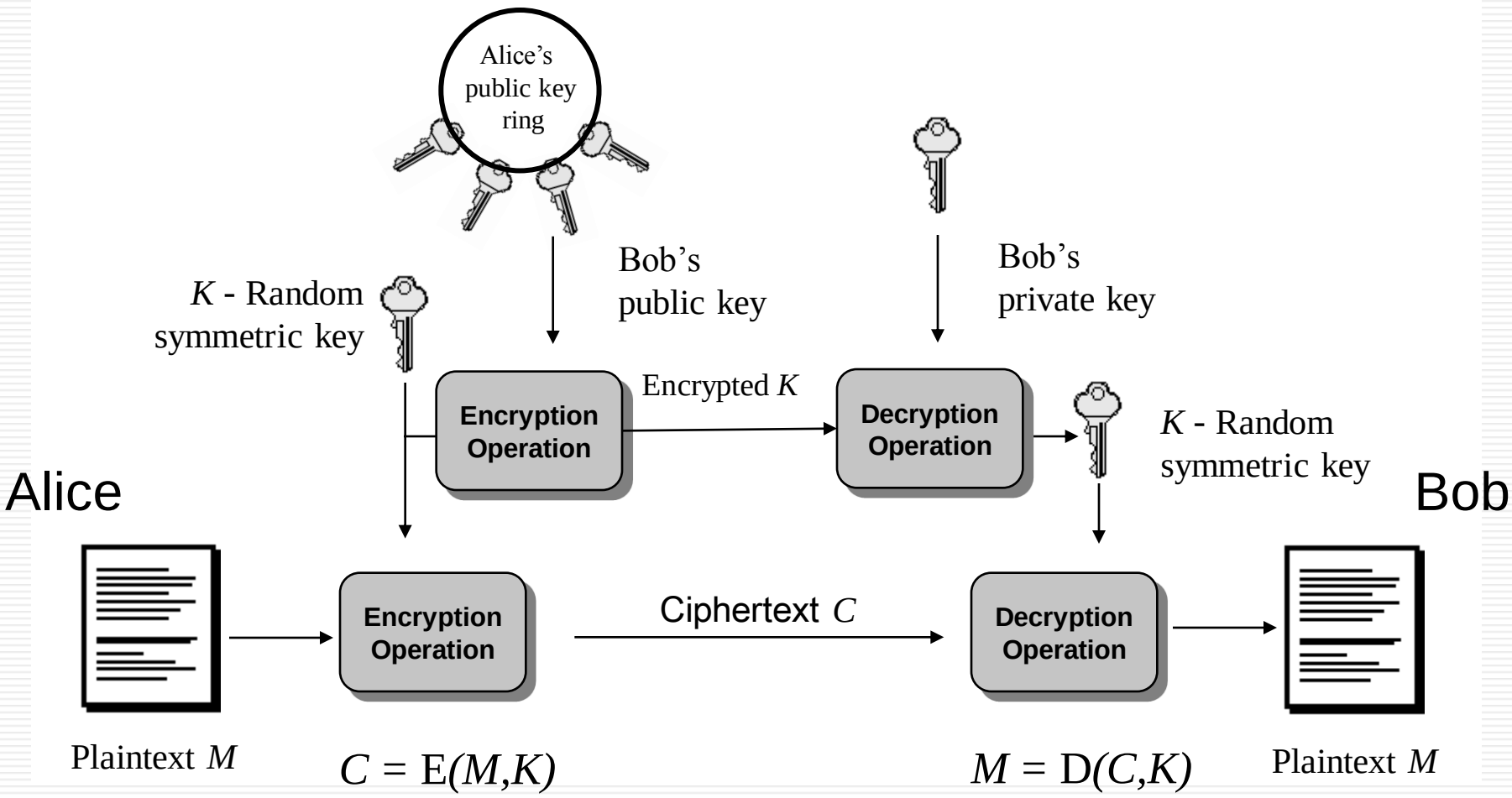
Symmetric and Asymmetric ciphers offering comparable security

AES Key Size	RSA Key Size	Elliptic curve Key Size
-	1024	163
128	3072	256
192	7680	384
256	15360	512

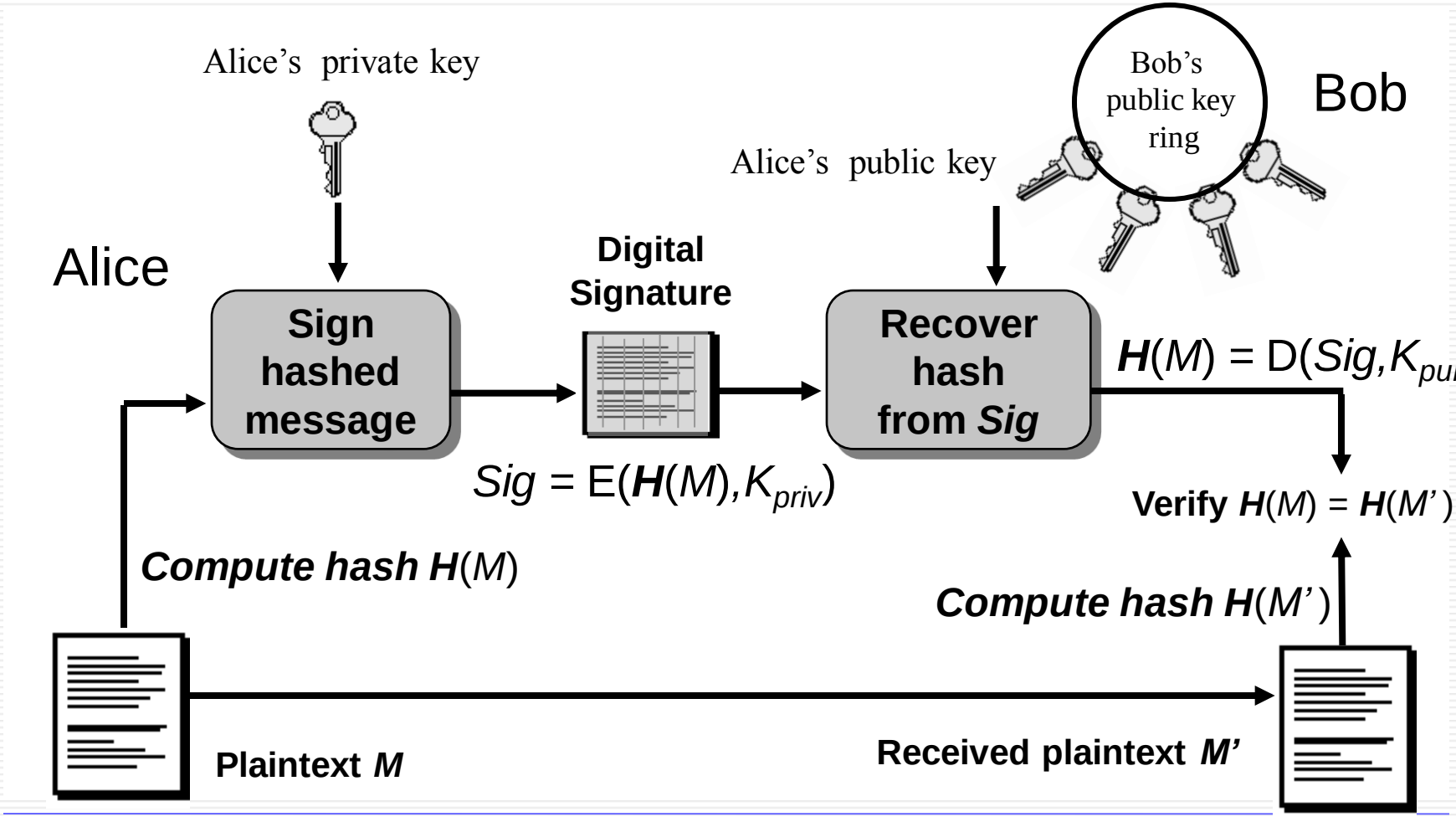
Hybrid Cryptosystems

- Symmetric ciphers are faster than asymmetric ciphers (because they are less computationally expensive), but ...
- Asymmetric ciphers simplify key distribution, therefore ...
- a combination of both symmetric and asymmetric ciphers can be used – a hybrid system:
  - The asymmetric cipher is used to distribute a randomly chosen symmetric key.
  - The symmetric cipher is used for encrypting bulk data.

Confidentiality Services: Hybrid Cryptosystems



Digital signature with asymmetric algorithm



Ciphers and security

- Cryptography provides no security if compromised key
- The security of the key involves:
  - Confidentiality:
    - the key must only be available to those authorized to use it.
  - Integrity:
    - if a key is altered, then the altered key will not be able to decrypt ciphertext formed using the original key.
    - If an attacker alters the key (replaces it with a key they know), then they will potentially have unauthorized access to all information that is encrypted with the new key.
  - Availability:
    - if an attacker destroys a key, all information currently encrypted with that key will be unavailable.

Ciphers and security

- Cryptographic keys represent sensitive information that needs information security
- We have replaced one security problem with another.
- Cryptographic techniques can be used to provide security for cryptographic keys. For example:
  - We can store our keys in a file which we encrypt (using another key) to provide confidentiality.
    - Forms a heirarchy of keys: master keys, session keys, etc
    - Like keeping your house key, car key, etc in a locked box
  - We can store hash values or MACs of key files to detect modifications, etc.

Ciphers and security

- A cipher must
  - be hard to cryptanalyse
  - use a sufficiently large key
- Algorithm secrecy makes cryptanalysis harder, but
  - can give false assurance, i.e. “security by obscurity”
  - challenging to keep cipher design confidential
  - safest to assume that attacker knows cipher

- Auguste Kerckhoffs proposed in 1883 that communication secrecy should only be based on the secrecy of the key
- Still, many organisations use secret algorithms.

Hopefully, you now know:

- What cryptography is
- When cryptography is used
- That there is a ‘perfect’ cipher, but that it has serious limitations
- That there are practical ciphers and how to use them
  - Symmetric ciphers (stream and block)
  - Asymmetric ciphers
- What hash functions are and how to use them
- That the security services ciphers provide depend on the security of the key