Data Integrity, Digital Signature and Entity Authentication

Security Engineering, Week 4

Dr Arman Khouzani, Dr Na Yao

Learning outcomes

- Appreciate there are different levels of data integrity.
- Identify the different properties of a hash function.
- Recognise the significance of the length of the output of a hash function.
- Explain how to use a MAC to provide data origin authentication.
- Compare different ways of providing both confidentiality and data origin authentication.

Learning Outcomes

- Explain general requirements for a digital signature scheme.
- Explain two different methods of creating a digital signature scheme based on RSA.
- Compare different techniques for providing freshness.
- Recognise a number of different approaches to providing entity authentication.
- Appreciate the limitations of password-based approaches to providing entity authentication.
- Explain the principle behind dynamic password schemes.

Data Integrity

Data Integrity - Introduction

- Data Integrity: assurance that data has not been altered in an *unauthorised* (which includes accidental) manner.
- Data integrity is not concerned with the *prevention* of alteration of data, but provides a means for *detecting* whether data has been manipulated in an unauthorised way.

- Can encryption provide data integrity?

Four different levels of Data Integrity

- **Accidental errors**
  - likely to occur through noise in a communication channel
  - error-correcting codes, simple checksums
- **Simple manipulations**
  - where an attacker cannot ‘shortcut’ the computation of a new digest by manipulating old ones
  - *hash functions*
- **Active attacks**
  - prevent an attacker from creating a ‘valid’ integrity digest on some data for which they have not previously seen an integrity digest
  - requires data origin authentication: *MAC*
- **Repudiation attacks**
  - protects against a creator of an integrity digest who attempts to later deny they created the digest
  - *Digital signature*

Hash functions

- *Cryptographic* hash functions
- A hash function accepts a variable-size message *m* and produces a *fixed-size* **message digest** *h*(*m*) as output.
- A hash function does *not* use a secret key.
- The sender sends the message and message digest so the receiver can authenticate the message.
- Hash functions are *publicly computable*. We always assume an attacker knows the details of a hash function.

Practical properties of a hash function

1. Compresses Arbitrary Long Inputs into a Fixed Length Output
2. Easy to Compute

Security Properties of a hash function

1. Preimage Resistance

   Given an output (hash) value z, it should be a difficult operation to find any input value x such that $h(x)=z$.

   Output $h(x)$ of a function h is often called the *image* of x, and hence x is referred to as a *preimage* of $h(x)$.

2. Second Preimage Resistance

   given an input x and its output (hash) value $h(x)$, it should be a difficult operation to find any other input value y such that $h(y)=h(x)$.

3. Collision Resistance

   for a hash function h, it is hard to find two different inputs x and y such that $h(x)=h(y)$.

Application of Hash functions

- **Application Requiring Preimage Resistance**
  password storage protection
- **Application Requiring Second Preimage Resistance**
  Software downloading with a hash of the executable code
- **Application Requiring Collision Resistance**
  Bob writes an IOU message, sends it to Alice, and she signs it. Bob finds two messages with the same hash, one of which requires Alice to pay a small amount and one that requires a large payment. Alice signs the first message, and Bob is then able to claim that the second message is authentic.

Attacking hash functions - The birthday paradox

- From the attacker's perspective, finding collisions is normally the 'easiest' attack to conduct.
- how many bits long should the hash of a message be in order for it to be regarded as secure?
- What is the probability that in this lecture there are two of us which have the same birthday. (there are approximately 90)
- Consider a small group of, say five. The number of ways which five dates can be chosen is

  365 x 365 x 365 x 365 x 365

- Of all of these ways only

  365 x 364 x 363 x 362 x 361

are such that no two dates are the same.

The birthday paradox

- The probability that five dates chosen at random is

$$\frac{(365 \times 364 \times 363 \times 362 \times 361)}{365^5} = 0.97$$

- Hence the probability that in a group of five persons two of them have the same birthday is 3%.
- Doing the same procedure with 140 persons the probability of two people having the same birthday is 99.9999999999%

The Birthday paradox

- A related question is how big a group of people has to be to have a **50%** chance that there are two people with the same birthday. Following the above approach we have that the group should have at least **23 people**.

The Birthday Attack

- The birthday attack is based on the birthday paradox
  - Number of days per year ⬅➡ number of possible hashes
  - How many pairs of modified original/fraudulent messages we need to create to have a 50% chance that they have the same hash.
  - For an *m*-bit hash we need $2^{m/2}$ modified messages.

The Birthday Attack

- An attacker needs to find a message M’ such that the hash of this message is the same as the hash of the original message M, that is H(M’) = H(M).

The Birthday Attack

- Alice is ready to ‘sign’ a message by encrypting, using her private-key, the m-bit hash code of the message.
- Eve generates $2^{m/2}$ variations of the message, all of each convey the same meaning.
- Eve also generates an equal number of messages all of which are variations of the fraudulent message that Eve wants to substitute for the original one.
- Eve searches in the two sets of messages a pair that produce the same hash code. i.e. [hash(variation of original) = hash(variation of fraudulent)]
- The probability success is greater than 0.5.

The Birthday Attack

- For the 64-bit hash code, Eve needs to create 2^{32} variations of the message (2^{32} = 4,294,967,296)

- Eve presents the valid variation of the original message to Alice for signature. After Alice signs the hash, the hash is attached to the fraudulent version of the message. And that is it.

The Birthday Attack

- For Eve it is not difficult to create variations of the message. Here is an example (32 versions of simple message).

Dear Anthony,

This letter is to introduce you to Mr. Alferd P. Borton,

I am writing to you - -

the new chief ...

newly appointed senior

Well-known Hash functions

- **MD5**: The hash function MD5 has been one of the most widely deployed hash functions, and was adopted as Internet Standard RFC 1321. It is a 128-bit hash function and is often used for file integrity checking. In 2004, collisions were found in MD5, and MD5 is no longer recommended for use.
- **SHA-1** family: The *Secure Hash Algorithm* was published by NIST in 1993 as a 160-bit hash function. It has been the 'default' hash function of the late 1990s and the early 2000s and used in numerous security applications, including SSL/TLS and S/MIME. In 2005, a technique was proposed for finding collisions for SHA-1 after 2^{63} computations, much faster than a birthday attack. There have been a number of improved attacks.

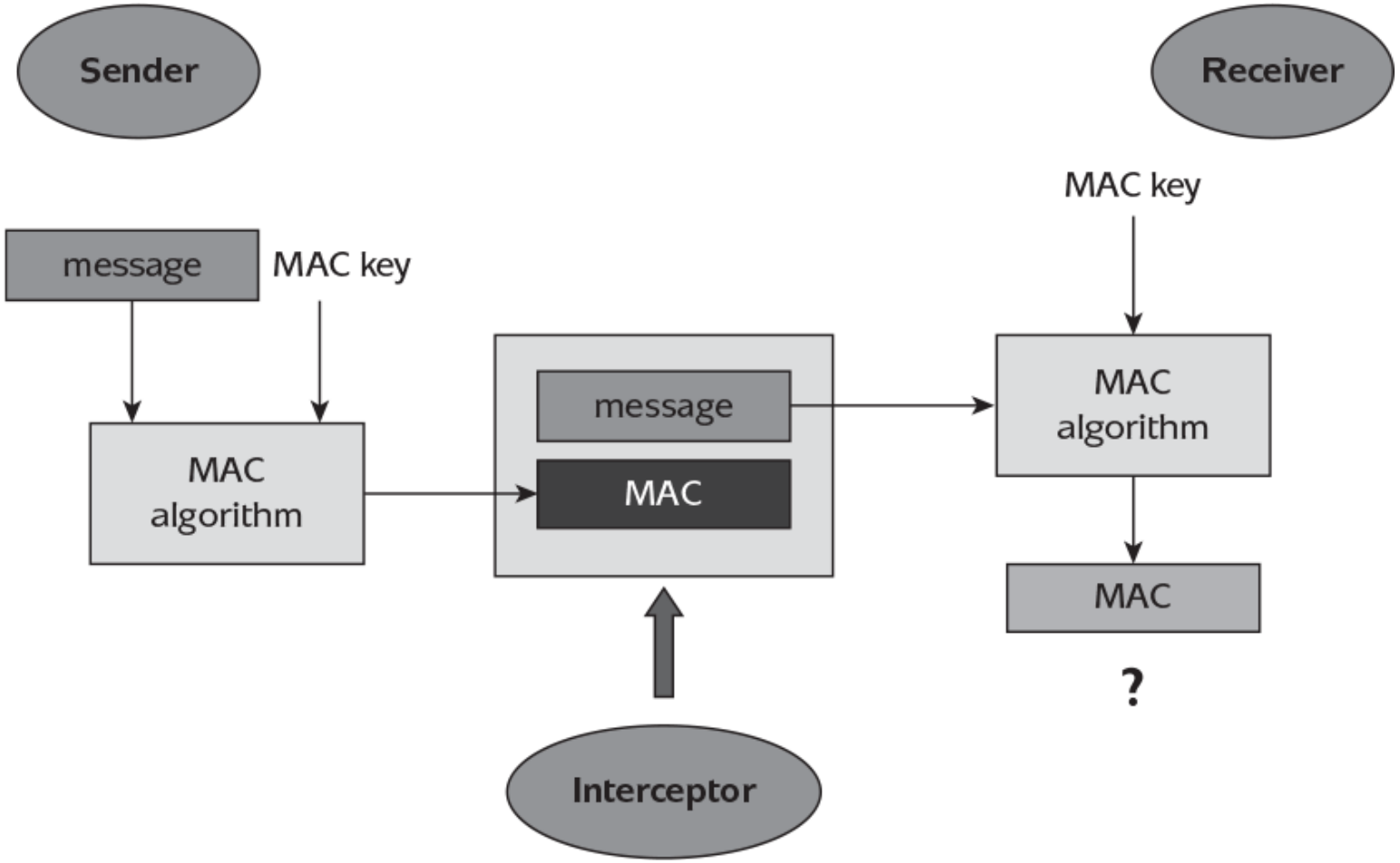
Well-known Hash functions 2

- **SHA-2** family: NIST published four further SHA variants, each of which is labelled by the number of bits of the hash output: SHA-224, SHA-256, SHA-384, and SHA-512 (collectively they are referred to as SHA-2). SHA-256 is the minimum security level recommended for many applications.
- **SHA-3**: In 2006 NIST initiated an 'AES style' *SHA-3 competition* for new hash functions which would provide a genuine alternative to the SHA-2 family. The winning hash function "*Keccak*" was announced in 2012 and SHA-3 was formally published In 2015.

Message authentication codes (MAC)

- MAC provides data origin authentication which, as we mentioned earlier, is a stronger notion than data integrity.
- MAC is a cryptographic checksum which is sent along with a message in order to provide an assurance of data origin authentication.

MAC – basic model



Message Authentication Code (MAC)

- Message Authentication involves generating a small block of data (the MAC) that is appended to the message.
- The MAC depends on the contents of the message and a secret key MAC(m) = $f(k_{AB},m)$, where m is the plaintext, k_{AB} is the shared key between users A and B.
- The message plus MAC are transmitted to the receiver.
- The receiver, which knows the secret key, calculates the MAC of the message and compares it with the MAC just received.
- Any changes in the message are noticed if the MAC calculated by the receiver and the MAC from the sender mismatch.

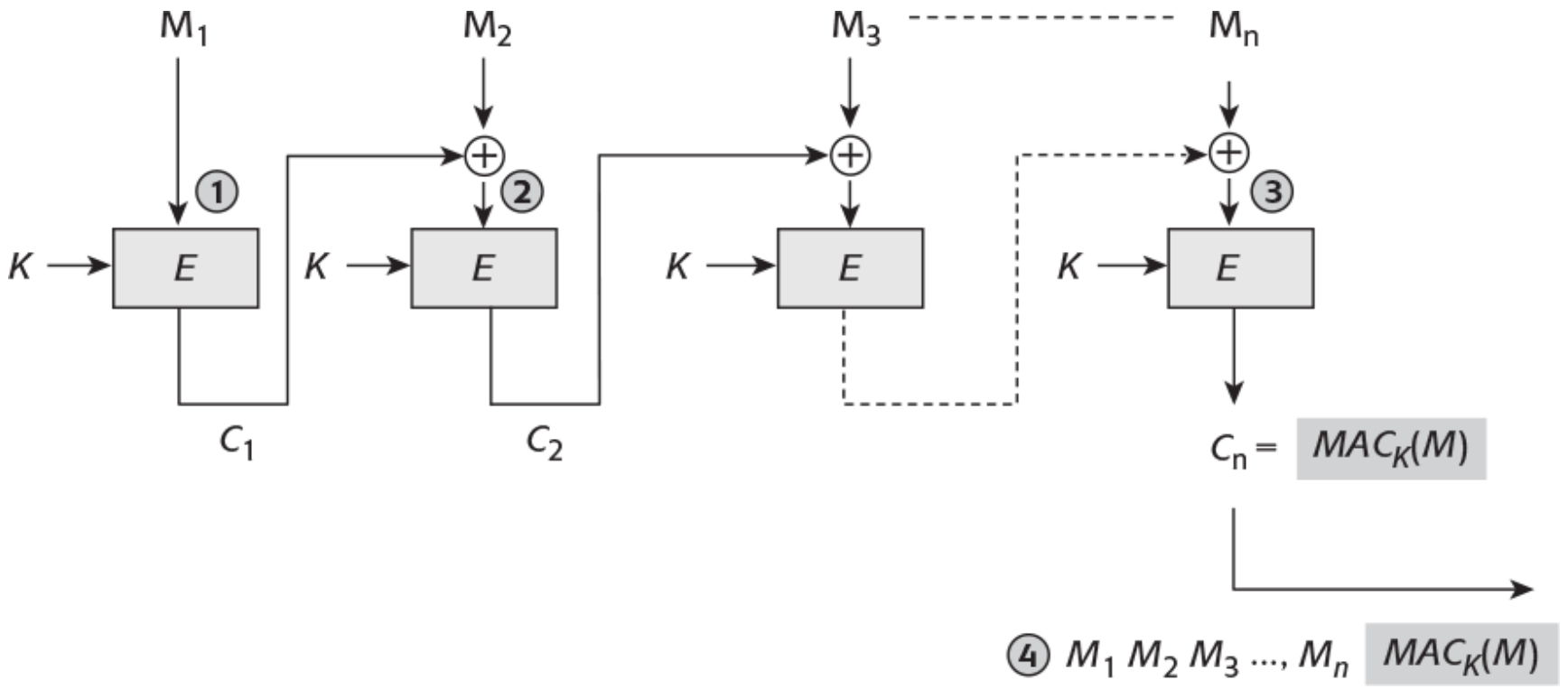
Authentication

- If the received MAC matches the calculated MAC then The receiver is assured that the message
  - has not been altered (integrity of the message).
  - is from the alleged sender (originator identity validation).

MAC algorithms

- Can we use Symmetric Encryption algorithms to construct MAC?
- After all, the sender and receiver need to share a secret key…

CBC-MAC



- The MAC in CBC-MAC is the last ciphertext block of a computation which is very similar to encryption of the message using the CBC mode of operation.
- For CBC-MAC, the process is started by using the first message block instead of creating an IV.
- In CBC-MAC we discard all the intermediate ‘ciphertext’ blocks.

HMAC (*Hash-based Message Authentication Code*)

- HMAC is a MAC derived from a cryptographic hash function, that is, it incorporates a secret key into an existing hash algorithm.
- Let h be a hash function, and let K_1 and K_2 be two symmetric keys. Then the MAC on message M is computed as follows:
  1. compute the hash of K_2 concatenated with the message; in other words compute $h\ (K_2\ ||\ M)$; and
  2. compute the hash of K_1 concatenated with the output of step 1; in other words compute: $h\ (K_1\ ||\ h\ (K_2\ ||\ M))$.

Security of HMAC

The security of HMAC depends on:

- The security of the keys.

HMAC employs two symmetric keys. Thus, the length of an HMAC key can be regarded as the sum of the lengths of these two keys.

- The security of the underlying hash function.
- The length of the MAC output.

Authenticated Encryption

- So far, we look at data origin authentication without confidentiality.
- Not a reasonable assumption!
- Most cases, we require confidentiality as well as data origin authentication. We will refer to this combination of security services as *authenticated encryption*.
- Using separate primitives
  - **MAC-then-encrypt**
  - **Encrypt-then-MAC**
- Authenticated-Encryption primitives
  - Galois Counter Mode (GCM)

- Can MAC provide non-repudiation?

Digital Signature

Digital Signatures scheme

We will define a *digital signature scheme* to be a cryptographic primitive providing:

- **Data origin authentication of the signer**. A digital signature validates the underlying data in the sense that assurance is provided about the integrity of the data and the identity of the signer.
- **Non-repudiation**. A digital signature can be stored by anyone who receives it as evidence.

Digital Signature

A digital signature on some data will need to be computed from:

- **The data**.
- **A secret parameter known only by the signer**.

Basic Properties of a Digital Signature Scheme

- Easy for the signer to sign data.
- Easy for anyone to verify a message.
- Hard for anyone to forge a digital signature.

Digital signature schemes based on RSA

Comparison of requirements between digital signature and public-key encryption schemes

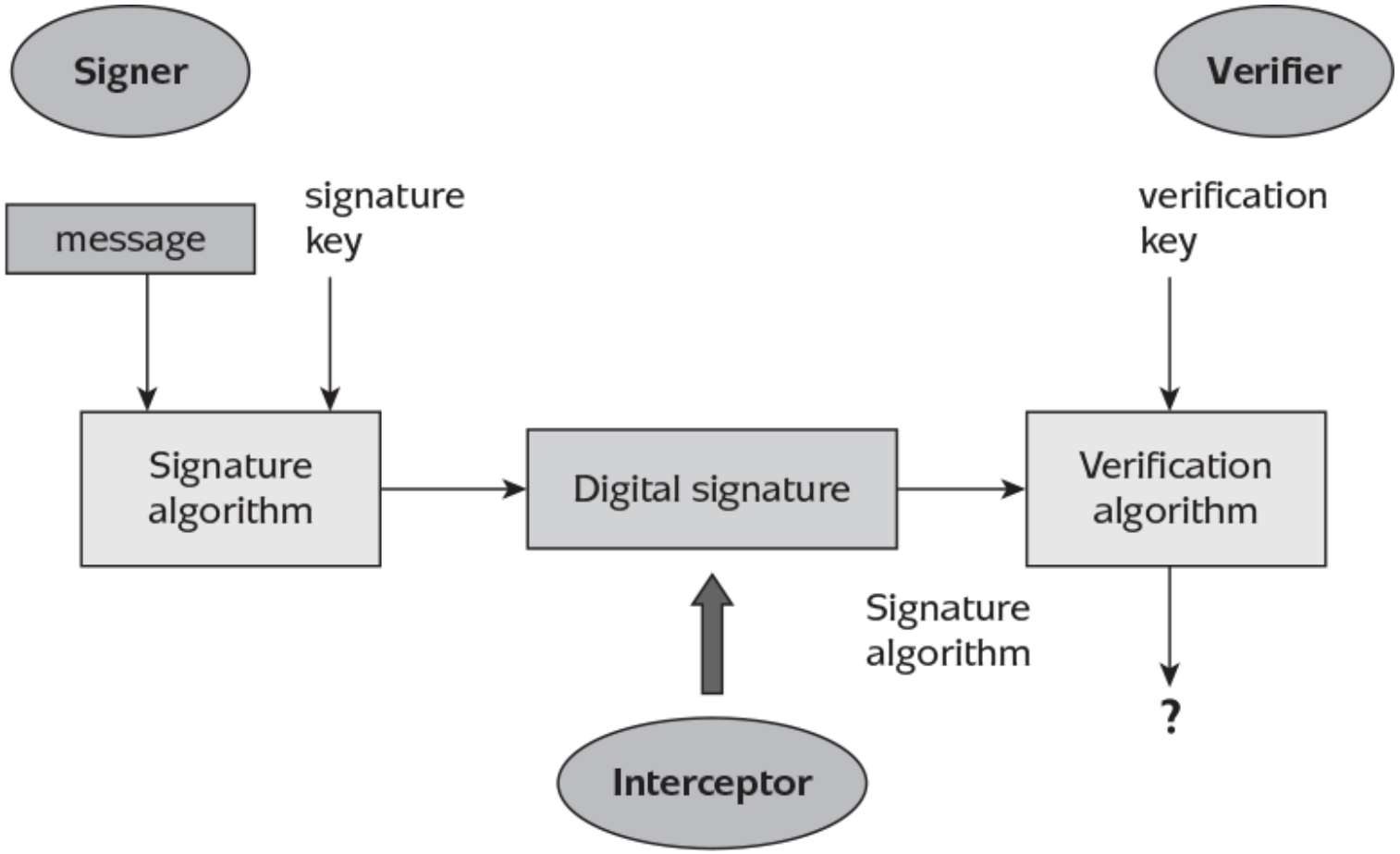
Digital signature scheme requirements	**Public-key encryption scheme requirements**
Only the holder of a secret can digitally sign some data	‘Anyone’ can encrypt some data
‘Anyone’ can verify that a digital signature is valid	Only the holder of a secret can decrypt some encrypted data

Can we “swap” the public keys and private keys in the public-key encryption to create/verify digital signatures?

RSA is Special

- In general, we cannot obtain a digital signature scheme from an arbitrary public-key cryptosystem by swapping the public and private keys.
- RSA is ‘special’ in that it allows the digital signature scheme to be realised by essentially swapping the roles of private and public keys in an RSA cryptosystem.

Basic model of a digital signature scheme



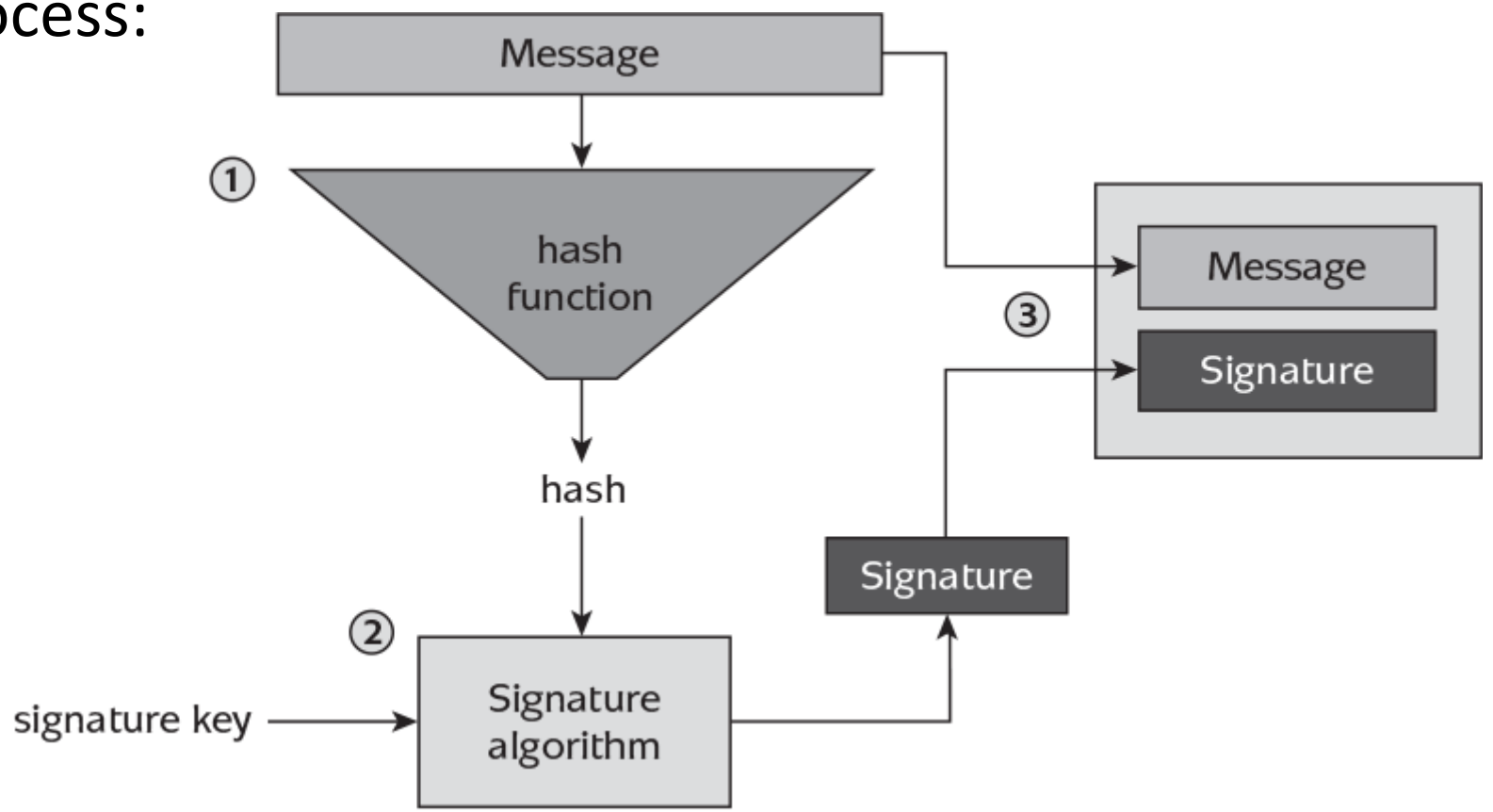
Two different approaches to Digital Signature Schemes

- How will the verifier know which data a digital signature matches?
- Or, how will the verifier be able to determine whether the output is the correct data?

- **Send the verifier the data being digitally signed** (*digital signature scheme with appendix*)
  - Data being digitally signed is not, by default, confidential. In many situations, it is acceptable to send the data ("appendix") along with the digital signature.
- **Add redundancy to the data being signed** (*digital signature schemes with message recovery*)
  - Make the data being digitally signed 'recognisable' by adding redundancy before computing the digital signature

RSA digital signature scheme with appendix

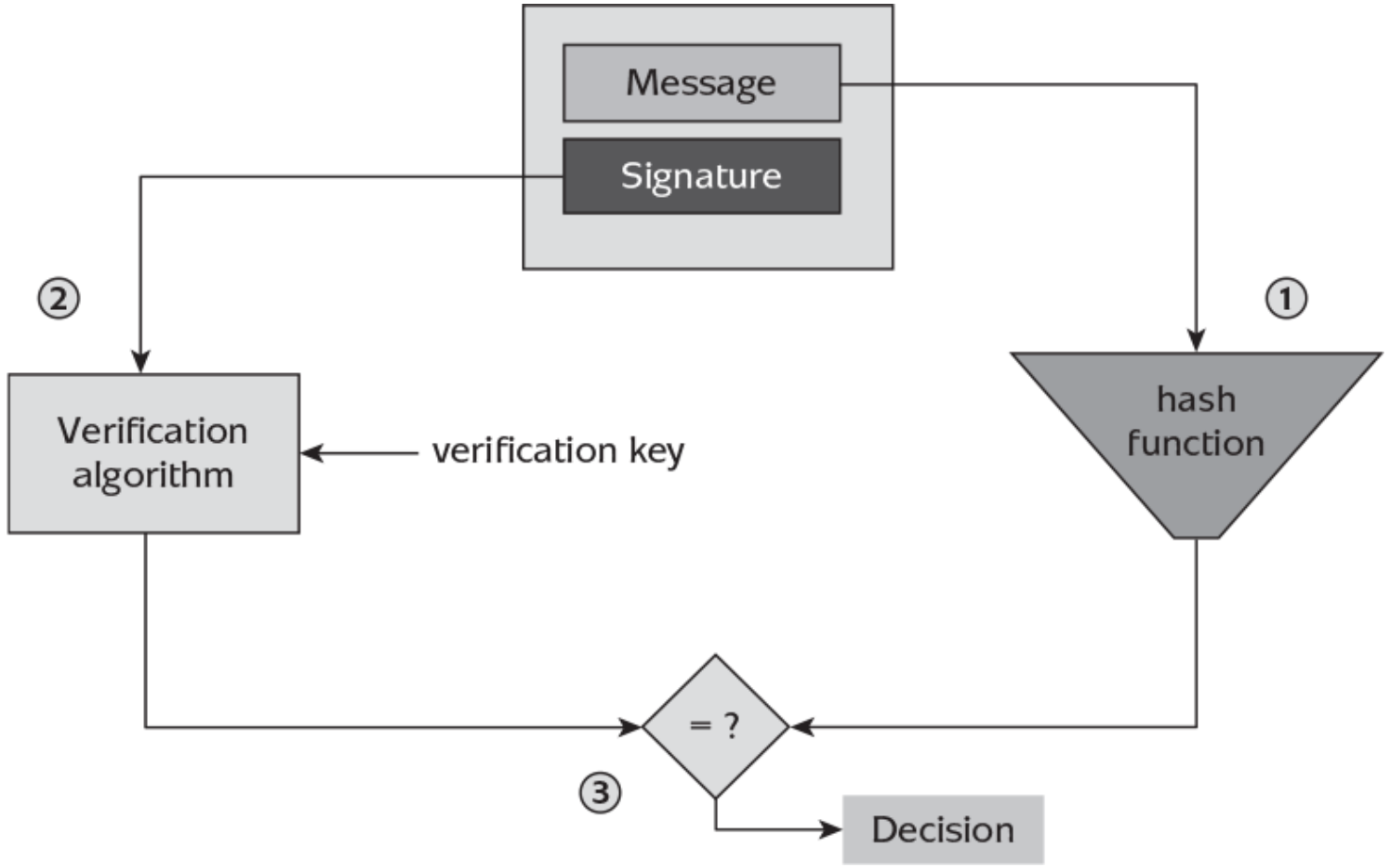
Signing process:



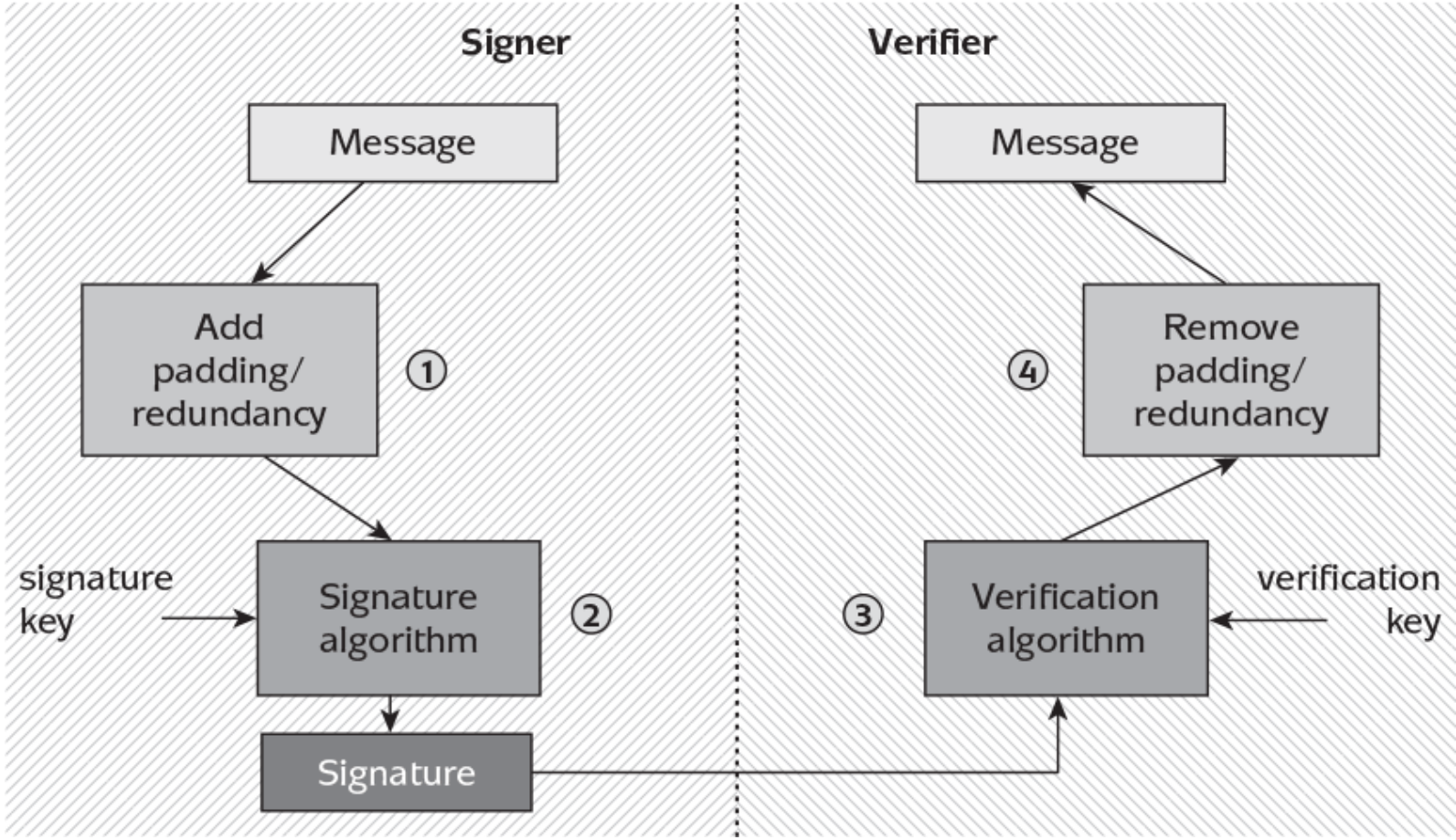
Hashing is needed for efficiency, preventing modification attacks and preventing existential forgeries.

RSA digital signature scheme with appendix

Verification process:



RSA digital signature scheme with message recovery (*digital signature schemes for short messages*)



Entity authentication

Entity authentication

- *Entity authentication* is the assurance that a given entity is involved *and currently active* in a communication session.
- *Replay attack*: an adversary captures a message, and then later replays it at some advantageous time.
- Entity authentication really involves assurance of ***identity*** and ***freshness*****.**
- *Freshness mechanisms* are techniques which can be used to provide assurance a given message is ‘new’, in the sense that it is not a *replay* of a message sent at a previous time.
- Does a message is ‘new’ imply the sender is ‘alive’?

Freshness Mechanisms

Clock-based mechanisms

- relies on the generation of some data that identifies the time the data was created. This is sometimes referred to as a ***timestamp***.
- Suppose Alice and Bob both have such a clock, and Alice includes the time on the clock when she sends a message to Bob. When Bob receives the message, he checks the time on his clock, and if it 'matches' Alice's timestamp, then he accepts the message as fresh.
- Requires clocks and synchronisation.

Freshness Mechanisms

Sequence numbers

- Logical time maintains a notion of the order in which messages or sessions occur and is normally instantiated by a *counter* or *sequence number*.
- Communicating parties need to maintain a database of the latest sequence numbers

Freshness Mechanisms

Nonce-based mechanisms

- *Nonces* (literally, ‘numbers used only once’): randomly generated numbers for one-off use.
- Alice generates a nonce at some stage in a communication session (protocol). If Alice receives a subsequent message containing this nonce, then Alice has assurance the new message is fresh, where by ‘fresh’ we mean the received message must have been created *after* the nonce was generated.
- Need to set a window of acceptance beyond which a nonce will no longer be regarded as fresh.
- Requires random generator
- Freshness requires a minimum of two message exchanges

Comparison of freshness mechanisms

	Clock-based	**Sequence numbers**	**Nonce-based**
Synchronisation needed?	Yes	Yes	No
Communication delays	Window needed	Window needed	Window needed
Integrity required?	Yes	Yes	No
Minimum passes needed	1	1	2
Special requirements	Clock	Sequence database	Random generator

Applications of entity authentication

- **Access control**

  An entity, must provide assurance of their identity in real time in order to have access. The entity can then be provided with access to the resources immediately following the instant in time they are authenticated.

- **As part of a more complex cryptographic process**

  Entity authentication is also a common component of more complex cryptographic processes, typically instantiated by a cryptographic protocol. An entity must provide assurance of their identity in real time in order for the extended protocol to complete satisfactorily

Passwords

- One of the most popular techniques for providing identity information.
- Problems with passwords
  - Length
  - Complexity
  - Repeatability
  - Vulnerability

Dynamic Password Scheme

- Limiting the exposure of the password, thus reducing vulnerability.
- Using the password to generate dynamic data which changes on each authentication attempt, thus preventing repeatability.
- Widely deployed in token-based technologies for accessing services such as internet banking or telephone banking.

Zero-knowledge mechanisms

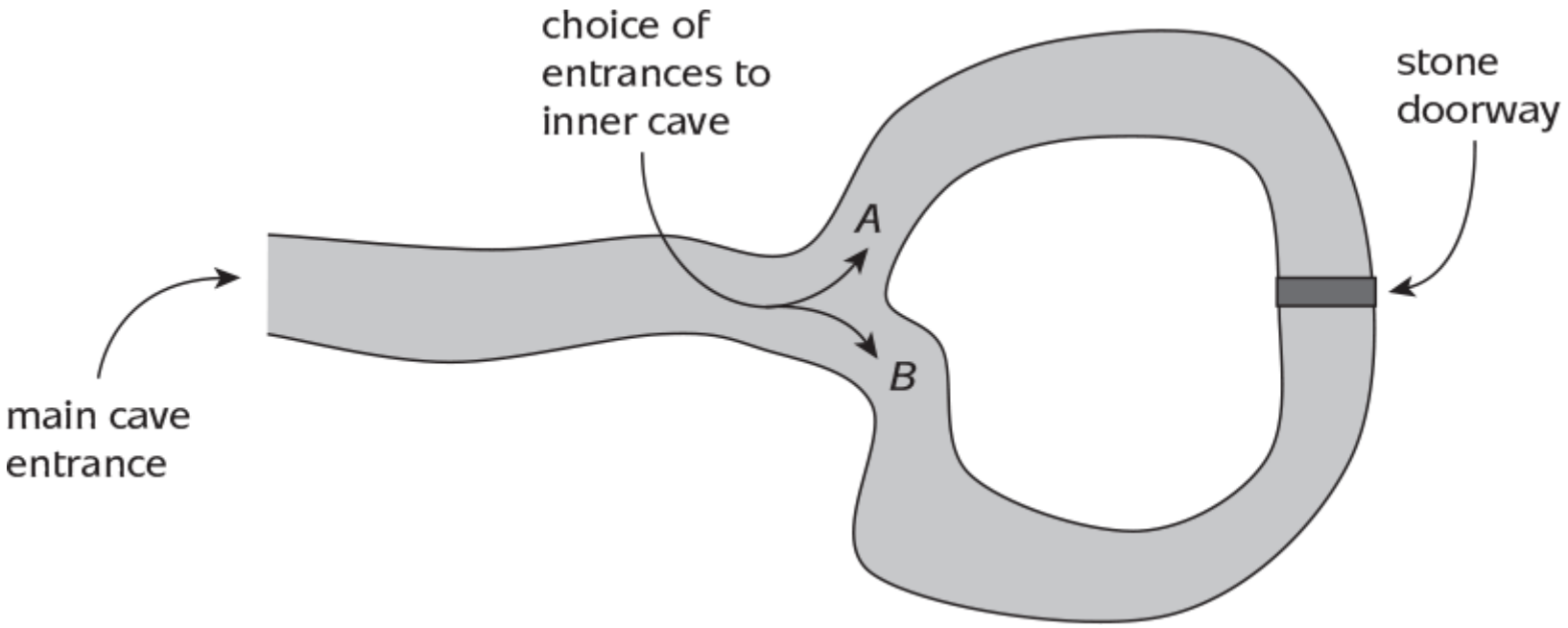
Two undesirable properties for current entity authentication:

- **Requirement for mutual trust**. Firstly, they are all based on some degree of trust between the entities involved.
- **Leaking of information**. Secondly, they all give away some potentially useful information on each occasion they are used. Conventional passwords are catastrophic in this regard since the password is fully exposed when it is entered.

Requirement for a zero-knowledge mechanism

- One entity (the *prover*) must be able to provide assurance of their identity to another entity (the *verifier*) in such a way that it is impossible for the verifier to later impersonate the prover, even after the verifier has observed and verified many different successful authentication attempts.

Zero-knowledge analogy



Summary

- Data Integrity
  - Hash functions
  - MAC
- Digital Signature
  - RSA digital signature schemes
- Entity Authentication
  - Freshness
  - Dynamic password scheme
  - Zero-knowledge mechanism