Introduction to Cryptography and Security Mechanisms

Dr Keith Martin
McCrea 349 01784 443099
keith.martin@rhul.ac.uk

Before we start…

Quiz 1

If two parties engage in a process that results in mutual entity authentication then at the end of the process the two parties can be reasonably confident that:

A All subsequent messages that they exchange come from one another

B The next messages that they exchange come from one another

C The messages that they have just exchanged came from one another

D Some messages that they recently exchanged were created by one another at some time in the past

Quiz 2

If an attacker intercepts a response that is issued during a successful challenge/response exchange and then tries to replay it a later date when a fresh challenge is issued, which of the following is the most likely reason for why the attacker will not succeed in being authenticated?

A The attacker does not know the correct PIN

B A response is only ever valid once

C The response needs to be accompanied by evidence of freshness

D The response might match the fresh challenge, but the probability of this happening is low

Quiz 3

Which of the following is **not** a problem with logical time-stamps (sequence numbers)?

A Synchronising logical time at either end of the communication link

B Preventing attackers from working out the next logical time-stamp

C Maintaining the integrity of logical time-stamps

D Deciding on procedures for coping with lost messages

Introduction to Cryptography and Security Mechanisms:

Unit 9

Digital Signatures

Dr Keith Martin
McCrea 349 01784 443099
keith.martin@rhul.ac.uk

Learning Outcomes

- Explain the concept of a digital signature
- Recognise that not all digital signatures rely on public key cryptography
- Appreciate the role that hash functions play in creating digital signatures
- Demonstrate how digital signatures can be created and verified using RSA
- Differentiate between digital signatures with appendix and digital signatures with message recovery
- Distinguish between the properties of digital and hand-written signatures
- Identify some of the main ways in which digital signature schemes can be attacked

Sections

1. Digital signature overview
2. Hash functions
3. Digital signature algorithms
4. Security issues

1. Digital signature overview

Informal definition

Informally, a **digital signature** is a technique for establishing the origin of a particular message in order to settle later disputes about what message (if any) was sent.

The purpose of a digital signature is thus for an entity to bind its identity to a message.

We use the term **signer** for an entity who creates a digital signature, and the term **verifier** for an entity who receives a signed message and attempts to check whether the digital signature is “correct” or not.

Digital signatures have many attractive properties and it is very important to understand exactly what assurances they provide and what their limitations are.

While data confidentiality has been the driver behind historical cryptography, digital signatures could be the major application of cryptography in the years to come.

Electronic signatures

The European Community Directive on electronic signatures refers to the concept of an **electronic signature** as:

> data in electronic form attached to, or logically connected with, other electronic data and which serves as a method of authentication

What different things can you think of that might satisfy this rather vague notion of an electronic signature?

Advanced electronic signatures

The European Community Directive on electronic signatures also refers to the concept of an **advanced electronic signature** as:

an electronic signature that is:

1. uniquely linked to the signatory
2. capable of identifying the signatory
3. created using means under the sole control of the signatory
4. linked to data to which it relates in such a way that subsequent changes in the data is detectable

Security requirements

We will define a **digital signature** on a message to be some data that provides:

- **Data origin authentication of the signer**
  - A digital signature validates the message in the sense that assurance is provided about the integrity of the message and of the identity of the entity that signed the message.
- **Non-repudiation**
  - A digital signature can be stored by anyone who receives the signed message as evidence that the message was sent and of who sent it. This evidence could later be presented to a third party who could use the evidence to resolve any dispute that relates to the contents and/or origin of the message.

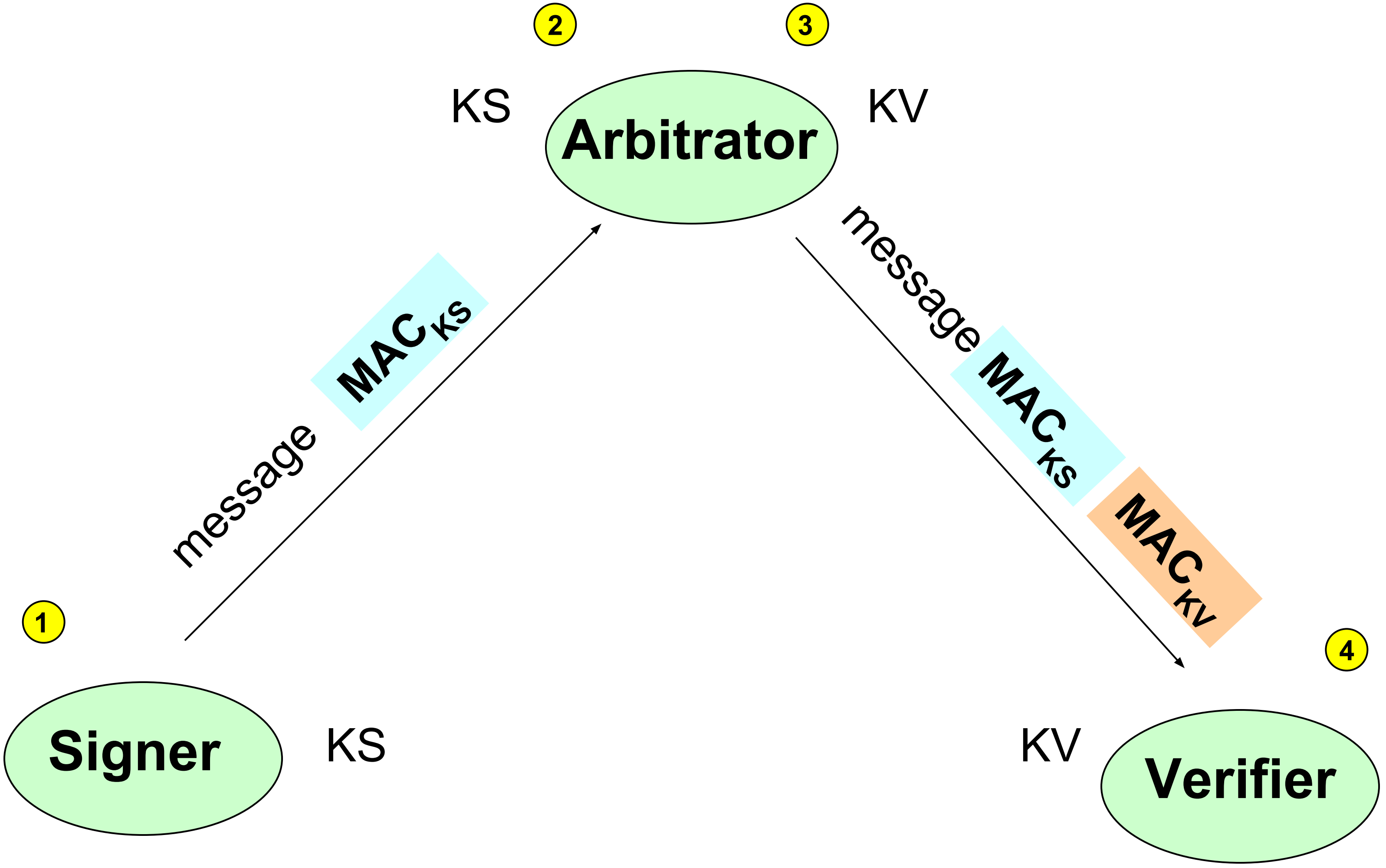
Input to a digital signature

- **The message**
  - Since a digital signature needs to offer data origin authentication (and non-repudiation) it is clear that the digital signature itself must be a piece of data that depends on the message, and cannot be a completely separate identifier.
  - It may be **sent** as a separate piece of data to the message, but its computation must involve the message.
- **A secret parameter known only by the signer**
  - Since a digital signature needs to offer non-repudiation, its calculation must involve a secret parameter that is known only by the signer.
  - The only possible exception to this rule is if the other entity is totally trusted by all parties involved in the signing and verifying of digital signatures.

Properties of a digital signature

- **Easy for the signer to sign a message**
  - There is no point in having a digital signature scheme that involves the signer needing to use slow and complex operations to compute a digital signature.
- **Easy for anyone to verify a message**
  - Similarly we would like the verification of a digital signature to be as efficient as possible.
- **Hard for anyone to forge a digital signature**
  - It should be practically impossible for anyone who is not the legitimate signer to compute a digital signature on a message that appears to be valid. By "appears to be valid" we mean that anyone who attempts to verify the digital signature is led to believe that they have just successfully verified a valid digital signature on a message.

Arbitrated digital signatures



Arbitrated digital signatures

1. Explain why arbitrated digital signatures

   a) meet the security requirements

   b) have the properties that we required

   for a digital signature.

2. How does the verifier check the first MAC, computed using KS?

3. What is the main (practical) problem with implementing arbitrated signatures?

True digital signatures

The vast majority of digital signature techniques do not involve having to communicate through a trusted arbitrator.

A **true digital signature** is one that can be sent directly from the signer to the verifier. For the rest of this unit when we say “digital signature” we mean “true digital signature”.

True digital signature requirements	Public key encryption requirements
Only the holder of some secret data can sign a message	“Anyone” can encrypt a message
“Anyone” can verify that a signature is valid	Only the holder of some secret data can decrypt a message

A naive approach

1. Given the apparent symmetry of the requirements for public key encryption and digital signatures, propose a naïve approach to designing a digital signature scheme.
2. State two reasons why the above approach is naïve.

2. Hash functions

Hash functions

A **hash function** is a mathematical function that generally has the following three properties:

1. **Condenses arbitrary long inputs into a fixed length output**
   - You stuff as much data as you want into the function, and it churns out an output (or **hash**) that is always the same fixed length.
   - In general this hash is much smaller than the data that was put into the function.
   - Because the hash is a smaller thing that represents a larger thing, it sometimes referred to as a **digest**, and the hash function as a **message digest function**.

Hash functions

2. **Is one-way**
   - The hash function should be easy to compute, but given the hash of some data it should be very hard to recover the original data from the hash.
3. **It is hard to find two inputs with the same output**
   - It should be hard to find two different inputs (of any length) that when fed into the hash function result in the same hash (**collision free)**.
   - Note that it is impossible for a hash function **not** to have collisions. If arbitrarily large inputs are all being reduced to a fixed length hash then there will be lots of collisions. (For example - it is impossible to give each of 60 million people a different 4 digit PIN.) The point is that these collisions should be **hard to find**.

Hash functions?

Consider the following two mathematical functions and explain whether they satisfy each of the properties of a hash function or not:

- Multiplying two prime numbers together
- Reducing a number modulo n

Practical hash functions?

There are several hash functions in common use that are believed to be secure enough for general use.

Can you name them?

Hash functions and data integrity

A hash function provides a weak notion of data integrity.

If we had a list of MD5 hashes which contained information on all of our operating system files on our home computer you could verify the values of your files in the list and see which files have been changed or have been updated by say a virus.

BUT

If a virus replaced the system file it could also replace the MD5 values in your list with new ones and you would not be aware this had happened…

Hash function applications

Digital signatures with appendix: hash-functions are used to bind data together and make the signature process more efficient.

Password storage: hash-functions are sometimes used to store highly confidential data such as passwords.

Cryptographic protocols: hash-functions are often used within cryptographic protocols (including entity authentication protocols) to bind different data items together.

Hash-functions can be used as components from which to construct other cryptographic primitives .

Is a MAC a hash function?

- Have a fixed length output
- Rely on a symmetric key
- Provide data origin authentication (and data integrity)
- Typically constructed from block ciphers or hash functions

Is a MAC a hash function?

HMAC

$$\mathbf{MAC = h(\ K \ || \ h(\ K' \ || \ message\ )\ )}$$

3. Digital signature algorithms

Some caveats

We will focus this section on describing digital signatures based on RSA. Please note:

- Although we only describe in detail how to implement digital signatures using RSA, there are many other examples of (true) digital signatures that are not based on RSA
- The RSA public key cipher system has some special properties that allow it to be used for both encryption and digital signatures – not all public key cipher systems can be used to generate digital signatures, and neither can all digital signature algorithms be used as public key cipher systems
- We will see two different methods of implementing true digital signatures using RSA – these two techniques can also be used for some other digital signature algorithms
- The process described here is simplified – please consult relevant standards before making an actual implementation

Motivating different types of signature

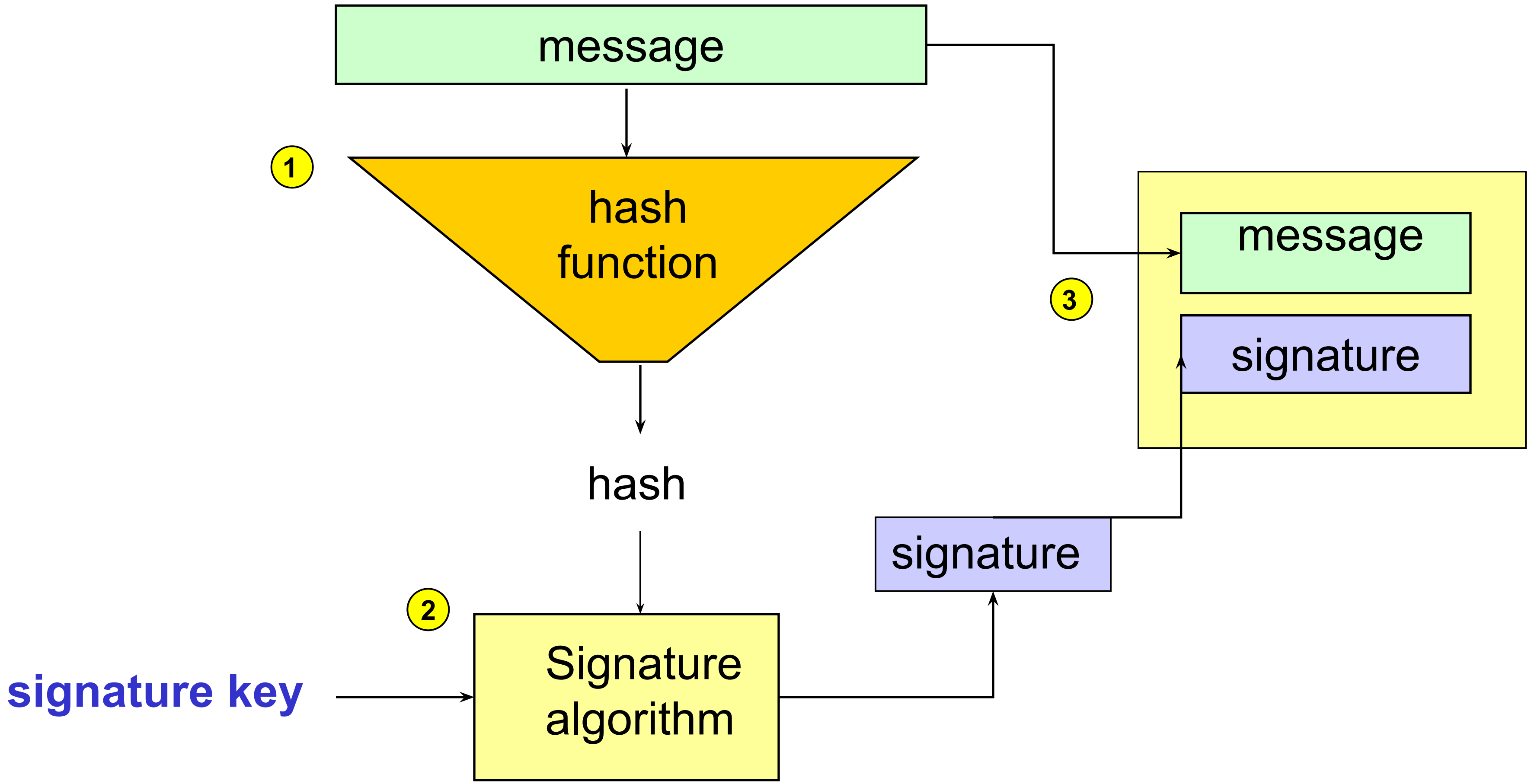
Suppose that you receive a digitally signed message that you are expected to be able to verify. Imagine that the message that is being signed is a random binary string.

To verify it you apply a publicly known process that does not involve any secret parameters.

Remember that an attacker could modify the signature on its way to you, changing a few bits here and there.

How are you going to know, just by performing a verification of the digital signature, what message the signature applies to and hence whether the signature is valid?

Creating an RSA signature with appendix

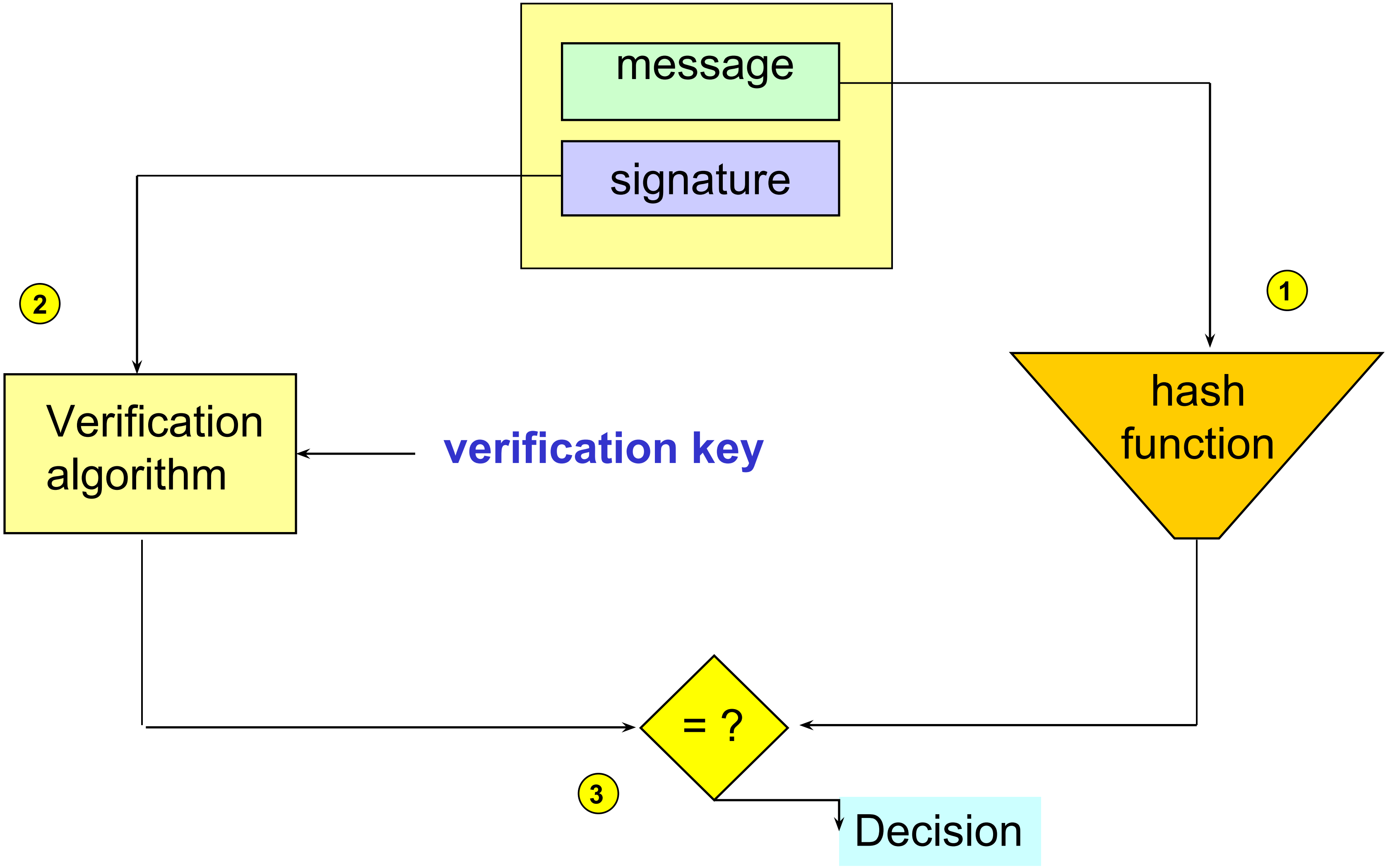


Hashing before signing

There are two reasons why a message is hashed before it is signed using RSA.

What are they?

Verifying an RSA signature with appendix



message

signature

1

hash function

2

Verification algorithm

verification key

= ?

3

Decision

RSA is special

You cannot obtain a digital signature scheme by swapping the roles of the private and public keys of any public key cipher system

You cannot obtain a public key cipher system by swapping the roles of the signature and verification keys of any digital signature scheme

Optional Task!

Express the operations involved in RSA signatures mathematically to check that the process of verifying an RSA signature with appendix does indeed work.

Now identify the special property of RSA that allows it to be used as both an encryption and a signature algorithm.

Key separation

In real applications you should avoid using the same RSA key pair for both encryption and for digital signatures.

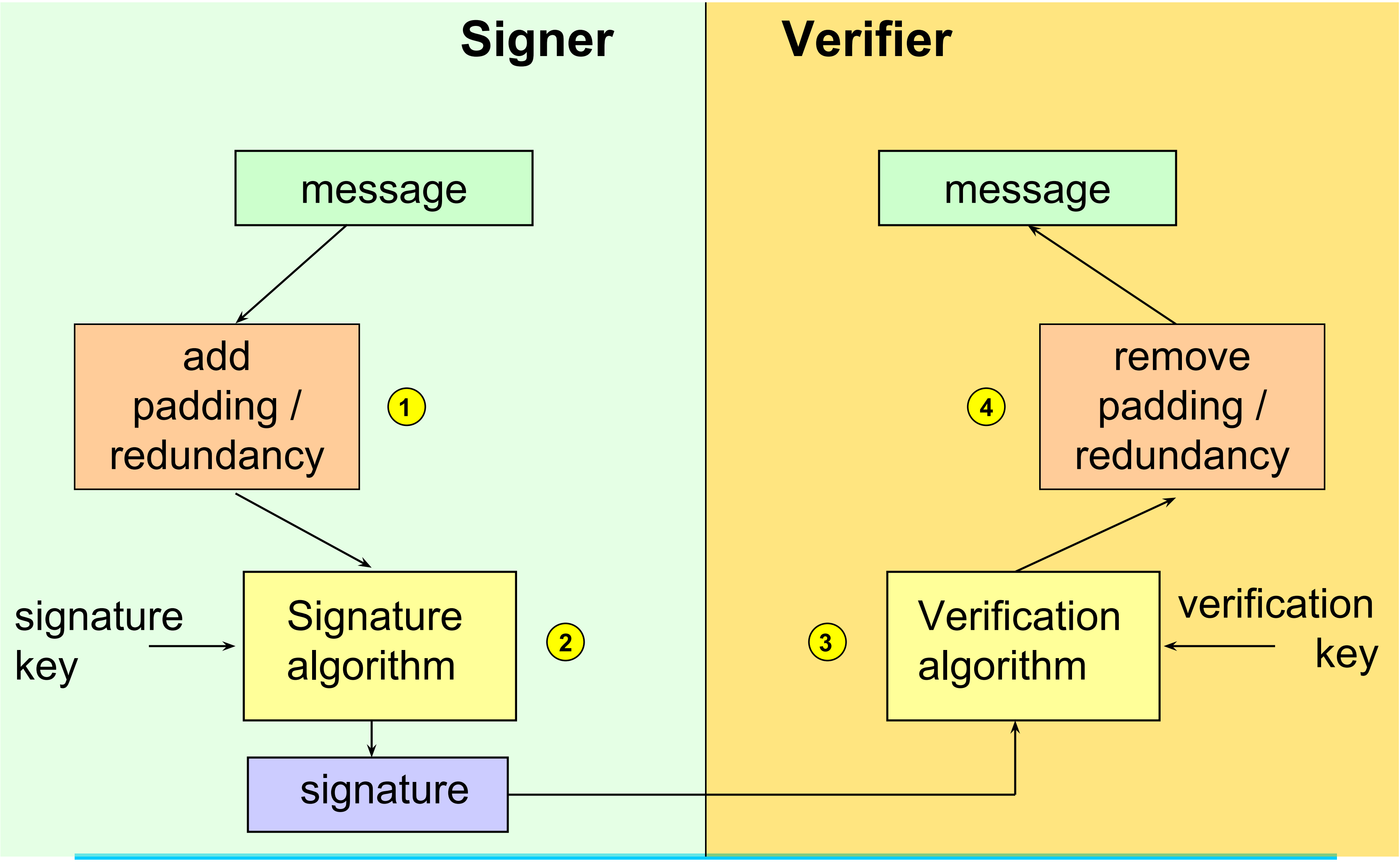
The reason is that good key management follows a principle known as **key separation**, where any cryptographic key has a specific role and is not used for different purposes.

Thus, properly implemented versions of RSA that are to be used for both encryption and digital signatures should issue each user with **two** key pairs:

- a public / private key pair for encryption
- a verification / signature key pair for digital signatures

These different key pairs should be carefully managed to ensure that they are only used for the designated purpose.

RSA signatures with message recovery



Digital Signature Algorithm

Although there have been many different proposals for digital signature schemes, only two systems have thus far proved to be fairly popular.

RSA digital signatures are one, and the other is a digital signature scheme based on ElGamal that was proposed as the **Digital Signature Algorithm** (**DSA**) and standardised by the U.S. Government as the **Digital Signature Standard**.

The DSA is a digital signature with appendix, but it cannot be used as a public key encryption system in the same way that RSA can be – it is a dedicated digital signature scheme.

4. Security issues

Basis of signature security

1. On what basis does a digital signature offer data origin authentication?
2. On what basis does a digital signature offer non-repudiation?
3. How do the security properties of a MAC and a digital signature differ?

Hand-written v digital signatures

Hand-written signatures	Property	Digital signatures
	Uniqueness	
	Accuracy of creation	
	Consistency over messages	
	Storage	
	Physical aspects	
	Difficulty of forgery	
	Acceptability	
	Legal support	

Two generic attacks

- **Obtain someone else’s private signature key**
  - In a digital signature scheme “you are your private key”.
  - This is one aspect of the problem of **identity theft**.
- **Persuade others that someone else’s public verification key belongs to you**
  - Others will verify it and believe that the message was signed by you.
  - This is a particularly “neat” attack because you do not need to obtain that other person’s signature key
  - An interesting variant of this attack for hand-written signatures arises if you steal someone else’s mail when a new credit card is sent out to them – if you just sign this blank card then you can easily masquerade as them.

Two generic attacks

1. How would you go about “stealing” someone else’s private signature key?
2. How would you prevent someone “persuading others” that your public verification key is actually their’s?

Security of hash functions

Because a hash is shorter than the message, collisions are inevitable – we just want them to be hard to find.

H**ow long does a hash have to be before finding collisions is hard**?

Security of hash functions

Suppose that we sign the message **Keith owes Fred £10** by hashing it using a hash function that has a hash of just 2 bits:

there are only four possible hashes: 00, 01, 10 or 11.

Fred receives this signed message, and being a manipulative type he decides to change the message to **Keith owes Fred £100**. Of course Fred does not have Keith's signature key, so he cannot digitally sign this message. But he doesn't have to – he only has to sign the hash!

What is the probability that:

hash (**Keith owes Fred £10**) = **hash** (**Keith owes Fred £100**)?

Security of hash functions

Suppose the hash is 10 bits long – in other words about 1000 hashes

1000 requests for £200

1. Pay Fred Piper £200
2. Pay F. Piper £200
3. Pay F.C. Piper two hundred pounds
4. Pay F.C. Piper two hundred pounds only
5. Pay two hundred pounds to Mr Fred Piper
6. ….

1000 request for £8000

1. Pay Fred Piper £8000
2. Pay F. Piper £8000
3. Pay F.C. Piper eight thousand pounds
4. Pay F.C. Piper eight thousand pounds only
5. Pay eight thousand pounds to Mr Fred Piper
6. ….

Since there are only 1000 different possible values of the hash, there is a **very good chance** that there will be at least one match…

Security of hash functions

1. What attack can Fred now launch against a payment clerk?
2. What lesson have we learnt?
3. How can this attack be easily avoided in practice?

Secure hash functions

In practice a common practical length for a hash is about 160 bits. This makes finding collisions of the type just described extremely unlikely, and also represents a significant compacting of the original message length.

Much shorter hashes than 160 bits are insecure, as we have seen.

Much longer hashes than 160 bits might be secure, but are not as efficient.

Finding good hash functions has proven to be a significant challenge to cryptographers.

Summary

- Digital signatures are in some senses a complimentary technology to public key encryption, offering data origin authentication and non-repudiation of digital messages.
- Digital signatures have different properties and offer different guarantees to hand-written signatures.
- The security of digital signatures critically relies on the security of the keys that are used to create and verify them.