# What is security?

## Security concepts

Communications systems



Transmission of information; messages need to be encoded

Possible noise in transmission; Error-correcting codes can answer, received message correctly? & can incorrect message be recovered?

Possible interception in transmission.

Message concealment:

Steganography:

Message-hiding so existence is not detected.

Cryptography:

Message disguised so unintelligible even if seen.

Security services

How to send information in such a way that it remains “secure” even if seen?

For Computer Security, the most common approach is to define it in terms of ‘CIA’:

▶ Confidentiality: the prevention of unauthorised disclosure of information.

▶ (Data) Integrity: the prevention of unauthorised modification of information.

▶ Availability: the prevention of unauthorised withholding of information or resources.

These are called security services.

Other security services

▶ Data origin authentication: assurance that information comes from reputed source unaltered.

▶ Entity authentication/identification: identities of sender/receiver confirmed and active involvement confirmed.

▶ Non-repudiation: prevents denial of previous actions.

More:

▶ Accountability, reliability, privacy, . . .

Data origin authentication necessarily includes data integrity but has no guarantee as to when the data was created.

For non-repudiation, we generally want a third party to be able to verify actions if there is some dispute.

Accountability; In practice not all improper actions can be prevented.

Thus users must be held accountable for their actions, including system misuse.

This is typically done by

▶ securely identifying users,

▶ audit trail of security-relevant events,

▶ log files.

The aim is to help resolve “who did what”.

Reliability; security is related to reliability and safety, dealing with systems which must perform properly in adverse conditions.

▶ The word dependability is sometimes used to encompass both security and reliability.

▶ Not only are the goals of security and dependability related, but similar methods are often used for system evaluation.

In the past the word privacy was often used as a synonym for confidentiality.

However, today the word is usually used to refer to privacy of personal data, or Personally Identifiable Information (PII).

Privacy covers a range of topics, including:

▶ Giving users control over their own PII.

▶ Requirements on data holders to look after PII properly.

## Assessing security

It is sometimes necessary for security-related products to be assessed against defined criteria.

▶ For example, a government department may not wish to use a product unless it has been certified as meeting pre-specified requirements.

Certain industry sectors also specify certification processes.

▶ For example, the payment industry with its Payment Card Industry Data Security Standard (PCI DSS).

In assessing a secure system, two different aspects of the system need to be considered:

▶ Functionality: what security facilities are provided by the system.

▶ Assurance: what guarantees are offered that the security functionality performs as claimed.

These two aspects are reflected in ‘Security Evaluation Criteria’, from the Orange Book onwards.

The work involved in providing a high level of assurance in security features is proportional to the complexity of those features. Hence, if a high level of assurance is required, it makes sense to minimise complexity.

## Security threats

Another way of defining security is to first perform a threat analysis.

▶ Relevant to all systems and not just computers.

▶ Security for the system can then be defined in terms of combating the perceived threats.

▶ Not all threats are always worth defeating (cost versus benefit).

ENISA (European Union Agency for Cybersecurity):

A threat is “any circumstance or event with the potential to adversely impact an asset through unauthorised access, destruction, disclosure, modification of data, and/or denial of service”.

Risk analysis

Essence of risk analysis:

▶ Assess the importance of each threat.

▶ Assess whether or not it should be combated.

Not just listing possible threats but:

▶ assessing likelihood of their being realised, and the cost to the system if this happens.

▶ The cost of living with some threats may be less than the cost of prevention.

## Providing security

Designing security into a system from the start is much better than adding security as an afterthought.

Security by design

Security needs to be integral part of system design/lifecycle.

Integration can be at many levels:

▶ Security could be a required feature.

▶ Unit test could include security testing.

▶ Security development could be extra step in the process.

Security controls

The protection of data in a computer system is often achieved using rules on system behaviour.

These rules may

▶ limit ways in which data is handled.

▶ limit which operations can be performed on which data.

▶ limit which users can perform certain types of action.

So security controls can focus on data, operations, or users (or some combination).

A typical IT system can be modelled using five layers:

▶ Application programs;

▶ Services;

▶ Operating system;

▶ Kernel (of the operating system);

▶ Hardware.

Security controls can be located in any of these ‘layers’.

Mechanisms close to hardware are typically more generic and computer-oriented, whilst those close to the application are typically more user-oriented.

Rules are not the only types of security control, one must not forget physical security.

▶ When providing security for data communications, the main tool is typically the use of cryptography and cryptographic protocols.

An attacker with access to a system layer below where a protection mechanism is located can bypass the control.

For example:

▶ Given ‘system privilege’ access to the operating system, application program controls can be bypassed (e.g. by directly accessing protected files).

▶ Given access to system hardware, the logical access controls of the operating system can be bypassed.

Security policies

A Security Policy is a set of rules specifying how security should be enforced within a domain (e.g. a department or company).

It is a statement that defines the security objectives of a computer system.

▶ States what needs to be protected.

▶ Indicates how this is to be done.

▶ Plan for what to do when a violation occurs.

Security policies should be:

▶ Easy to read and remember.

▶ Unambiguous.

▶ In line with the culture of the organization.

▶ In support of good productivity and innovation.

▶ Able to adapt to changes in working environment.

## Laws and regulations

Computer Misuse Act 1990

Computer Misuse Act 1990 (”CMA”) is the main UK legislation relating to offences or attacks against computer systems.

▶ Provision for securing computer material against unauthorised access or modification.

Some offences under the CMA:

▶ ”Causing a computer to perform a function with intent to secure unauthorised access to computer material.”

▶ ”Unauthorised acts with intent to impair the operation of a computer.”

▶ ”Unauthorised acts causing, or creating risk of, serious damage, for example, to human welfare, the environment, economy or national security.”

Data Protection Act 2018

The UK Data Protection Act (updated 2018) gives holders of personal data an obligation to protect the accuracy and privacy of personal data.

▶ Personal data is any information relating to an identified or identifiable living individual.

▶ An identifying characteristic could include a name, ID number or location data.

Some offences under the DPA 2018

▶ ”Destroying or falsifying information and documents etc”.

▶ ”Unlawful obtaining etc of personal data”.

▶ ”Re-identification of de-identified personal data”.

▶ ”Alteration etc of personal data to prevent disclosure to data subject”.

GDPR

The General Data Protection Regulation is a EU regulation on data protection and privacy for all individuals in the EU.

▶ Adopted in 2016, enforced in May 2018.

▶ “Data protection by design and by default”: pseudonymization, anonymization, no use without explicit informed consent, revocation of consent possible, ...

▶ From 1 January 2021, the DPPEC (Data Protection, Privacy and Electronic Communications (Amendments etc) (EU Exit)) Regulations 2019 merged the DPA 2018 and the EU GDPR to form a new, UK-specific data protection regime.

# Cryptography

## Cryptosystems/ciphers

A diagram of a system

Description automatically generated  
  
  
  
  
  
  
  
Plaintext: the raw data to be protected, the input to a cipher.

Ciphertext: the result of applying an encryption algorithm to plaintext.

Encryption key: a value known to the sender. Used as input to the encryption algorithm to compute the ciphertext from the plaintext.

Encryption algorithm: rules that takes a plaintext and an encryption key, and outputs a ciphertext.

Decryption key: a value known to the receiver. Used as input to the decryption algorithm to compute the plaintext from the ciphertext.

Decryption algorithm: rules that takes a ciphertext and a decryption key, and outputs a plaintext.

The interceptor: also known as the eavesdropper, or the attacker and the adversary more generally.

▶ it attempts to determine the plaintext. (What else?)

▶ it can always see the ciphertext.

▶ it may know the encryption and decryption algorithms.

▶ it does not have the decryption key.

Conventions

Entities:

▶ Sender: Alice.

▶ Receiver: Bob.

▶ Interceptor: Oscar (opponent), Eve (eavesdropper).

▶ Generally assume insecure/public channel.

▶ Kerckhoff’s assumption: Cryptographic algorithm not required to be secret.

▶ Assume interceptor knows cryptosystem.

▶ All security resides in the key.

▶ Also assume interceptor has ciphertext and some corresponding plaintext.

Attack models: Amount of information Oscar has.

▶ Ciphertext only: knowledge of ciphertext alone.

▶ Known plaintext: corresponding plaintext and ciphertext.

▶ Chosen plaintext: chosen plaintext and corresponding ciphertext. Adversary has access to encryption machine.

▶ Chosen ciphertext: chosen ciphertext and corresponding plaintext. Adversary has access to decryption machine.

Modern cryptosystems are generally required to be at least secure against a chosen plaintext attack.

Attacker capabilities: what Oscar can do

Passive attacks:

▶ Unauthorised access to data.

▶ Eavesdropping.

Active attacks:

▶ Tampering with data.

▶ Deletion of data.

▶ Tampering with origin of data.

▶ Preventing access to data.

▶ . . .

Attack goals: what Oscar wants

▶ Determine the plaintext.

▶ Determine the key.

▶ Modify the message.

▶ Masquerade as Alice.

▶ Other subtle aims?

A common adversary: attacker-in-the-middle

▶ We assume the adversary controls the entire communication channel.

▶ The adversary can read all traffic.

▶ The adversary can also drop, delay, modify any network traffic.

Common criteria for evaluating security of cryptosystems:

▶ Computational security: There is no known method of breaking the cryptosystem with a ”reasonable amount” of computational resources.

▶ Provable security: Breaking the cryptosystem is ”equivalent” to solving a hard computational problem.

▶ Unconditional security: The cryptosystem cannot be broken even with infinite computational resources.

Cover time: Length of time we believe our cryptosystem will resist a particular attack.

Exhaustive key search

Exhaustive key search/brute force: An attacker can always try every key.

Attacker has some ciphertext.

1. Pick a decryption key.

2. Decrypt ciphertext using that decryption key.

3. Check if the resulting plaintext ”makes sense”.

4. If not, pick another decryption key. If it does then the decryption key is a candidate decryption key.

5. Stop when there is confirmation that the candidate key is the correct decryption key.

Very large key spaces make this infeasible; modern ciphers have enormous key spaces.

Key spaces usually expressed in bits; number of digits in base 2.

A white background with black text

Description automatically generatedFeasibility of different numbers of operations:

Worst case assumptions

assume attacker (cryptanalyst) has:

▶ Full knowledge of the encryption algorithm.

▶ A number of ciphertexts all computed using the same secret encryption key.

▶ Some known plaintext, that is, part or all of some plaintexts corresponding to known values of ciphertext.

Possible extra assumption – chosen plaintext.

Given the worst case assumptions, it is important for a user to try to find ways to break the cryptosystem.

▶ That is, the cryptosystem designer must play the role of the cryptanalyst.

▶ In practice, cryptosystems are used which are believed to be strong.

All this means is that the best attempts of experienced cryptographers cannot break them.

Caesar cipher

Julius Caesar coded his personal correspondence with a substitution algorithm as follows:

Each letter of the message was substituted by one three positions further down the alphabet.

Nowadays, any cipher in which a letter is substituted with a fixed shift is called Caesar’s cipher.

A diagram of a mathematical equation

Description automatically generated

We convert message M into a number.

We assume the key K is also a number (in this case K = 3).

Encryption is modular addition. Decryption is modular subtraction.

”Wrap around”: For example, 4 + 1 mod 5 = 0, 6 + 4 mod 7 = 3.

A close-up of a chart

Description automatically generated

A math problem with numbers and equations

Description automatically generated with medium confidence

A white background with black text

Description automatically generated

Caesar’s cipher can be cryptanalyzed easily using exhaustive key search, i.e., try every possible key until a meaningful plaintext is obtained.

▶ Indeed, there are only 26 possible keys.

▶ We can say ”Caesar’s cipher is not computationally secure against a ciphertext only attack, given a sufficient amount of ciphertext”.

Substitution ciphers

A Caesar cipher is actually just one special type of simple substitution (monoalphabetic) cipher.

These ciphers have as a key, a permutation of the letters of the alphabet.

Encryption involves replacing each letter by its permuted version, and decryption involves use of the inverse permutation.

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer code

Description automatically generated

e.g.,

Philosopher Al-Kindi, born in Baghdad in 801, described the first cryptanalytic tool: frequency analysis.

Key idea:

If we know what language a ciphertext belongs to, we can analyse a plaintext written in the same language, and count how many times each letter appears. We can rank letters in order of frequency, and map these to the ranking we perform on the ciphertext.

Substitution (monoalphabetic) ciphers cannot be secure, despite large key space.

▶ Preserve language statistics.

Polyalphabetic cipher:

▶ Plaintext letter sent to many ciphertext letters.

▶ Flattens letter frequency graph.

▶ Key must define unique plaintext.

Vigenère cipher

In the 16th century, Blaise de Vigenère created the most famous polyalphabetic cipher, which became known as le chiffre indechiffrable (French for 'the indecipherable cipher'), since it took over 300 years to break it.

▶ Map letters to numbers, as usual.

▶ Choose a keyword K of length m.

▶ Add the keyword to the plaintext, that is, encrypt m letters at a time.

A yellow and black text with numbers

Description automatically generated

In the 19th century, British mathematician Charles Babbage decided to approach the challenging task of deciphering this polyalphabetic cipher.

A screenshot of a white paper with black text

Description automatically generatedKey observation: Two identical segments of plaintext will be encrypted to the same ciphertext when they occur at a distance which is a multiple of m.

1. Determine the length m of the keyword

2. Use frequency analysis to determine plaintext.

One-time pad (Vernam cipher)

If the key is as long as the message, AND the key is completely random, AND each key is used only once, THEN

This cipher has “perfect secrecy” - you don’t know more about the plaintext after seeing the ciphertext.

One-time pad as motivation: replace random key with “pseudorandom” keystream generated from short key

Symmetric key cryptography

Symmetric cipher systems are generally classified as stream ciphers or block ciphers:

Block ciphers encrypt block-wise:

▶ Use fixed function.

Stream ciphers encrypt one bit at a time:

▶ Use a time-varying function.

▶ Widely used for mobile communications.

▶ Example: Vigenère cipher and rotor machines, acting on English alphabet rather than bits.

#### Stream cipher

A computer screen shot of a computer

Description automatically generated

Plaintext is represented as a sequence of bits.

▶ A keystream generator is chosen. It takes a secret key as input.

▶ It outputs a sequence of bits - the keystream.

▶ The plaintext is encrypted by adding (modulo 2) the keystream to give the ciphertext.

A math equations and numbers

Description automatically generated with medium confidenceEg.,

Generally fast. Encryption operation is elementary, and highly secure keystream generators can be implemented to operate at very high speeds.

Require less memory: appropriate when no buffering is available and/or characters need to be processed on-the-fly (important for telecommunications).

Dedicated stream ciphers are particularly suitable for:

▶ Hardware-oriented algorithms, with exceptionally small footprint (e.g. gates, power consumption), for constrained environments (e.g. RFIDs)

▶ Software-oriented algorithms, with very high speed (e.g. routers).

Low error propagation: advantageous if transmission errors likely. No protection against message manipulation. Same key used twice gives the same keystream; a security risk to be avoided - hence a need for “message” or “session” keys.

Influenced by one-time pad. One-time pad has key management problems: key too long.

▶ Stream cipher is an attempt to emulate one-time pad: Use shorter key to generate long keystream.

▶ Design of good keystream generator is fundamental.

▶ Keystream must appear “random”.

Is a stream cipher secure against a known plaintext attack?

▶ If you know some plaintext and the corresponding ciphertext you know some of the keystream.

Is a stream cipher secure against a ciphertext only attack?

▶ If you can predict the keystream you can obtain the plaintext.

Is a stream cipher secure against an active attack such as tampering of data?

▶ Modification of ciphertext undetectable unless there are redundancies in the plaintext.

##### Periodicity & randomness

Any deterministic sequence is ultimately periodic.

▶ Sequence must ultimately consist of repetitions of some finite sequence.

▶ Shortest repeated sequence is called a cycle.

▶ Length of a cycle is called the least period. No periodic sequence is truly random.

▶ Require “unpredictability” of “short” subsequences. Such sequences are called pseudorandom sequences. Sequences we use have large periods (e.g. 1050 > 2150).

▶ Apply statistical tests to sections to check they appear random - local randomness. There are standards specifying tests; Passing tests necessary but not sufficient.

##### Linear equivalence

Any periodic binary sequence can be generated by linear methods: by computing each element of the sequence as a linear combination of preceding elements in the sequence.

Indeed it can be generated using a linear feedback shift register (LFSR).

Linear equivalence of a sequence: size of the smallest LFSR that can generate the sequence.

▶ The smaller an LFSR the fewer bits one would need to know to determine the entire sequence.

▶ Hence we would like high linear equivalence for a secure keystream.

##### Properties of secure keystreams

▶ Long period: a short period and some corresponding amount of plaintext will reveal the keystream.

▶ Pseudorandomness properties: must “look” like a truly random sequence.

▶ Large linear equivalence: a sequence with small linear equivalence means an attacker needs fewer bits to ”break” the sequence.

These conditions are necessary but not sufficient to guarantee security. More generally: it must be computationally infeasible to gain information about one part of the keystream given other parts of the sequence.

##### Keystream generation

Approaches: one-way functions, intractable problems from number theory, LFSRs.

Produce pseudorandom sequences with large periods (264 bits or more). LFSRs are basic building block, used in most published designs.

Need to be easy to implement, efficient. long sequences, good statistical properties (Can be analysed algebraically to give security assurances)

LFSRs: basic building blocks in most stream ciphers.

▶ Well-suited for hardware implementation (e.g. LFSR with 200 stages fast and compact to implement).

▶ Produce sequences of large period and good statistical properties.

▶ Easy to analyse algebraically.

Unfortunately, the entire sequence can be compromised by knowledge of a tiny part of the sequence.

▶ In general, used in combination to introduce non-linearity.

##### Other generators

▶ Block cipher modes of operation: CBC, OFB, CFB.

▶ Table-driven ciphers: RC4 (now considered insecure), SEAL, HC-128.

A diagram of a block diagram

Description automatically generated

#### Block ciphers

Widely used. Encrypts data a block

(e.g. 64-, 128- or 256-bit blocks) at a time.

Primitive for many cryptographic functions:

▶ stream ciphers.

▶ hash functions.

▶ message authentication codes (MAC).

A block cipher is a simple substitution cipher. The alphabet consists of n-bit blocks.

Vulnerable to dictionary attacks.

▶ Prevent by using large block size so such attacks ineffective.

▶ Length of plaintext block is important to security.

▶ Should be at least 64 bits long, preferably 128 bits. E.g., Data Encryption Standard (DES) uses 64-bit blocks, Advanced Encryption Standard (AES) uses 128-bit blocks

Ciphertext is a permutation of n-bit blocks.

▶ Potentially a lot of permutations.

▶ Modern practical block ciphers have h-bit keys.

Number of keys = 2h. Often h = n.

DES has n = 64 bits blocks, but the key space has size 256.

AES has n = 128 bits blocks but the key lengths are 128-bit and 192-bit and 256-bit.

##### Iterated ciphers

All modern block ciphers are iterated ciphers.

Iterated ciphers: repeated use of a round function. A round function takes an n-bit block to another n-bit block. Each round uses a round key - subkeys derived from the key using a key schedule.

To allow decryption, for every subkey the round function must be invertible.

▶ Need method to specify a “good” round.

▶ Enough rounds can give a secure cipher.

##### Substitution-permutation (SP) networks

A technique for good round functions.

Split n-bit block into t sub-blocks of s bits.

For each round:

▶ Substitute the s-bit sub-blocks separately.

▶ Permute all n bits of the block.

Rationale:

▶ Permutation diffuses changes of substitution.

▶ Use good encryption functions on small sub-blocks, then diffuse effect to whole block.

▶ Iterate to ensure changes fully diffused.

##### Feistel ciphers

Feistel ciphers are iterated ciphers.

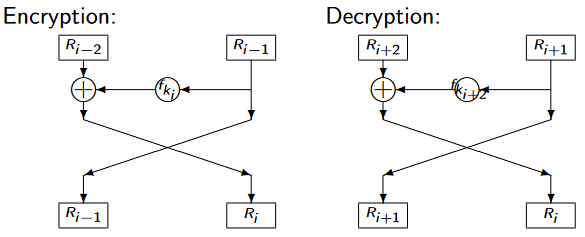
▶ Even block size 2n, data block split into two halves (L0, R0).

▶ r rounds: for round i, input (Li−1, Ri−1), output (Li , Ri ).

▶ Plaintext (L0, R0), ciphertext (Rr , Lr ) (swapped).

▶ Key schedule: from key k, define r subkeys k1, . . . kr, one for each round.

Round function fki : Zn2 → Zn2 for round i:

▶ Li = Ri−1.

▶ Ri = Li−1 ⊕ fki (Ri−1).

Decryption same process as

encryption, subkeys in reverse

order.

Image shows rounds; after

Completing final round, swap L & R

Round function fki need not be invertible.

Round function often SP Network.

Key schedule:

▶ Different key parts used in different places.

▶ DES: 56-bit key gives 16 48-bit round subkeys.

▶ AES: 128-bit key gives 11 128-bit round subkeys.

##### The Data Encryption Standard (DES)

DES is a 16-round Feistel cipher. Plaintext and ciphertext are 64-bit blocks. Keys are 56 bits long; 56 bits is widely regarded as insufficient for many modern applications. No longer recommended but variants are still widely used; historical importance and influential design.

▶ 64-bit plaintext blocks and 56-bit keys (plus 8 parity bits).

▶ Subkeys are each of 48 bits.

▶ Designed around manipulation of bits.

▶ Efficient implementation.

▶ XOR operations, small look-up tables or hard-wired simple circuits.

DES ideas are basis for other block ciphers.

###### Controversy

Should there be a standardised encryption algorithm? DES was first cryptographic algorithm to be standardised.

Advantages of standardisation:

▶ solves problems of compatibility.

▶ lower cost of implementation.

Disadvantage of standardisation:

▶ if cryptanalyst breaks widely used standardised algorithm then they would gain access to many users’ messages.

Now generally accepted that standardisation is needed.

Should the standardised encryption algorithm be the DES? Small key space; 56-bit key not thought to offer security, even in 1970s.

DES key space has size 256. Search can be done in software or hardware. Easy to parallelize.

A close-up of a list

Description automatically generated

###### Double DES

Why not use DES twice with two keys?

Double DES: y = e(K1,K2)(x) = eK2 (eK1 (x))

Exhaustive key search:

▶ One key: Length 56 bits, so size of key space is 256.

▶ Two keys: Length 2 × 56 = 112, so size of key space is 2112.

▶ So will take 2112 operations to search? Secure?

Meet-in-the-middle attack

Given a plaintext-ciphertext pair (m, c),

1. Compute m′i = eki (m) ∀ 256 keys ki

2. Store (m′i , ki ) in a table

3. Decrypt c under all 256 possible keys kj. For each decryption,

▶ check table to see if dkj (c) = m′i .

▶ if so (ki , kj ) is a candidate key

4. If more than one match, use another plaintext-ciphertext pair to see which of the matches work for this pair, and so on.

Meet-in-the-middle” attack on DES requires

▶ 256 pre-computations,

▶ Storage for 256 words,

▶ Another 256 decryption,

▶ One table-lookup.

Only 2×single DES encryptions: so 257 operations.

▶ Yet key length suggests work effort of around 2112. operations

▶ Double DES not really used.

“Meet-in-the-middle” attack works for any repeated encryption of any block cipher.

A white paper with black text

Description automatically generated

###### 2-key triple DES (2TDES)

To get round the problem of short DES key, triple-DES is widely used as replacement.

Two-key variant: use two DES keys (k1, k2). c = e(k1,k2)(x) = ek1(dk2(ek1(m)))

Note decryption in the “middle” operation.

This is equal to single DES if k1 = k2. This simplifies migration.

There is also a 3-key variant of triple-DES (3TDES).

Simple “meet-in-the-middle” attack not possible; cannot split encryption into two independent parts due to repeated use of k1. A different attack reduces security to about 80 bits. 2TDES still widely used in many important applications.

##### Advanced Encryption Standard (AES)

Standardisation of the DES was withdrawn in 1990s.

Key size and block size too small. In 1997-2000: NIST held competition to find replacement. Result was Advanced Encryption Standard (AES).

Block cipher with 128-bit blocks and 128/192/256-bit key versions.

Aims:

▶ Public selection process. (selection of types of AES)

▶ Public design, and freely available for public use.

▶ Faster than 2TDES.

Rijndael chosen as selected AES in 2000. It has good performance and the design is very structured - aids in security analysis and performance.

Widely deployed in hardware and software. Also deployed in low-cost environments such as RFID.

▶ Flexible: different block sizes and key sizes

##### Modes of operation

A block cipher can be used in a variety of ways to encrypt a data string.

Soon after DES was made a US Federal Standard, another US standard appeared giving four ‘recommended’ ways of using DES for data encryption. These modes of operation have since been standardised internationally (for use with any block cipher). More recently, a fifth mode has also been standardised (and widely used).

FIPS (Federal Information Processing Standards) 81 has 4 modes of operation for DES

▶ ECB: Electronic Codebook

▶ CBC: Cipher Block Chaining

▶ CFB: Cipher Feedback

▶ OFB: Output Feedback

The more recent fifth mode is counter (CTR) mode

###### Electronic codebook (ECB) mode

A diagram of a block diagram

Description automatically generatedSimplest use of a block cipher

Divide data m into q n-bit plaintext blocks m1, . . . , mq

ci = eK (mi), i = 1, . . . , q. (Last block padded if necessary.)

Decryption: mi = dK (ci), i = 1, . . . , q

Information leakage: Identical plaintext blocks give identical ciphertext.

Also allows cut-and-paste, replay and dictionary attacks.

▶ May improve security by padding in each block.

▶ Reordering of blocks not detected (can’t know if attacker did that).

▶ Modification of block not detected (for random data).

▶ No Error propagation: Errors in single ciphertext block affect decryption of that one block only.

###### Cipher block chaining (CBC) mode

CBC aims to hinder information leakage.

Uses “random” ciphertext to vary input.

Plaintext block xor-ed with previous ciphertext block before encryption.

Plaintext m must be made into a series of n-bit blocks (if necessary with padding added to the last block)

A diagram of a computer

Description automatically generated

Encryption:

ci = eK (ci−1 ⊕ mi ), i = 1, . . . , q

where c0 = IV, an n-bit initialisation vector.

Typically IV is different for every message.

A diagram of a block diagram

Description automatically generatedDecryption: let c0 = IV, and mi = ci−1 ⊕ dK (ci)

No Information leakage:

if yi = yj, xi = yi−1 + dK (yi) different from xj = yj−1 + dK (yj).

(i.e., same plaintext blocks (in the same message) get encrypted

to different ciphertext)

Error propagated in transmission: error in ciphertext yj effects

encryption of xj+1.

IV integrity must be protected, otherwise can make predictable changes to x1 when decrypting

###### Counter Mode (CTR)

The NIST document published December 2001: Special Publication 800-381 defines five modes for AES: FIPS 81 and Counter Mode.

A Stream cipher mode; Block cipher used to generate a keystream sequence.

Encrypt n-bit counter values, the ciphertexts from this encryption form the keystream.

A diagram of a flowchart

Description automatically generatedAllows for random access.

Encryption:

ci = mi + eK (ctr + (i − 1))

Where ctr is a counter which is a string the

same size as the block size.

Decryption: mi = ci + eK (ctr + (i − 1))

###### Other modes of operation

Output feedback mode (OFB): A Stream cipher mode.

Repeatedly encrypting an initialisation vector get keystream.

Cipher feedback mode (CFB): A Stream cipher mode.

Plaintext not directly encrypted. Encryption of a register is added to plaintext. Ciphertext is used to update register.

Public key (asymmetric key) cryptography

#### Issues in symmetric cryptography

The same key is used to encrypt and decrypt. This means:

▶ Need mutual trust: anything the sender can do, the receiver can do.

▶ Need key establishment: The sender and receiver need to agree on a symmetric key in advance.

▶ Compromise of dK or eK compromises system

In 1976 a revolutionary idea by Diffie and Hellman suggested using two different keys; the birth of public-key cryptography.

#### Overview

RSA invented by Rivest, Shamir and Adleman in 1977.

In a public key encryption system, the sender has the encryption key and the receiver has a (distinct) decryption key. Knowledge of the encryption key does not reveal knowledge of the decryption key.

Every user generates a key pair (encryption and decryption keys) and makes public their encryption key.

Everyone can send them a secret message, but only they can decrypt it.

▶ eK can be different from dK.

▶ dK computationally infeasible to derive from eK.

▶ eK can be made public so anyone can encrypt.

▶ Only the person who knows dK can decrypt.

▶ Authenticity of the public key should be provided

Public key cryptosystems are never unconditionally secure.

▶ If ciphertext c is observed, can try all possible m using public encryption rule eK till m is found such that eK (m) = c.

▶ Hence study computational security.

A diagram of a diagram

Description automatically generated

PK is public, so anyone can encrypt to Bob.

SK is private to Bob, so only Bob can decrypt.

So now security depends entirely on the secrecy of the private key SK .

In particular, it must be hard to compute SK from PK .

It should be computationally easy for an entity A to generate key pair (PKA, SKA).

It should be easy for any entity B knowing PKA and a message m, to generate the corresponding ciphertext, c = ePKA(m).

It should be computationally easy for entity A to decrypt ciphertext using SKA: m = dSKA(c).

It should be computationally infeasible for an attacker, knowing PKA and c, to determine SKA or the original message m.

##### Trapdoor one-way function

One-way functions are easy to compute but “hard” to invert.

Genuine receiver needs to be able to decrypt. Trapdoor information allows decryption.

Example: f(x) = xa mod n, n a product of two large primes.

f is invertible for most odd a.

But f −1 believed to be hard to compute. E.g., find x such that x3 = 648220 mod 2379409.

But knowing factors of n makes computing f −1 feasible.

#### Public key cryptosystems

##### RSA

Encryption and decryption use modular exponentiation. Security based on difficulty of factoring large integer.

###### The integer factorisation problem

The basic idea of RSA relies on the difficulty of finding the prime factors of large numbers.

Given N = pq, find p and q. It is easier to multiply two numbers than it is to reverse the process.

For example,

▶ If p = 47 and q = 71 then n = p × q = 3337. This is easy to calculate.

▶ If p × q = 3337 with p > 1 and q > 1 then what are p and q?

▶ Multiplication is feasible but factoring is considered to be a difficult problem.

We work with integers modulo n, written Zn = {0, 1, 2, . . . , n − 1}.

Addition, subtraction, multiplication are all performed modulo n:

▶ For any integer a, we can divide a by n to get a remainder r .

▶ We can write a = ns + r with 0 ≤ r < n. (s is the quotient.)

▶ We say a is congruent to r mod n.

▶ We write a = r (mod n).

▶ Every integer is congruent to some r mod n for 0 ≤ r < n.

To multiply two integers a, b modulo n:

▶ We multiply a and b as integers.

▶ If the product is between 0 and n − 1 (inclusive) we are done.

▶ Otherwise we divide it by n and take the remainder. We call this ”reducing modulo n”.

▶ Similarly with addition and subtraction.

Examples:

▶ 8 × 4 = 2 (mod 5).

▶ 10 − 20 = 4 (mod 7). (-10 = (7 \* -2) + 4)

Modular exponentiation xa (mod n) works similarly:

▶ You could calculate xa and then reduce modulo n.

For very large numbers this can be very inefficient.

▶ We could instead reduce modulo n after every multiplication.

There are efficient algorithms to calculate modular exponentiation efficiently.

A number p is a prime number if it is divisible only by 1 and itself.

The greatest common divisor of two numbers a and b is the largest positive integer dividing both a and b.

We write gcd(a, b). Example: gcd(24, 60) = 12, gcd(13, 15) = 1.

We say that a and b are coprime (or relatively prime) if gcd(a, b) = 1. Example: 13 is coprime to 15.

###### Generating an RSA key pair

1. Generating the modulus. Choose two large primes (at least 512 bits long), p and q. Let N be the product of p and q, so N = pq.

2. Generating e. Select a number e, where 1 < e < (p − 1)(q − 1) and (e, (p − 1)(q − 1)) are coprime.

3. Forming the public key. (N, e) are set to be the public key and can be made available to anyone.

4. Generating the private key. Compute the private key d from p, q and e,

such that ed = 1 mod (p − 1)(q − 1).

Textbook RSA Encryption and decryption

RSA encryption:

▶ Let the message M be a number less than N.

▶ Compute C = Me mod N.

RSA decryption:

▶ Let the message C be the ciphertext.

▶ Compute M = Cd mod N.

Follows from mathematical fact

###### Security

What does it mean to break a public key encryption scheme?

▶ An attacker gets hold of information about the secret key from public information.

▶ An attacker gets hold of information about the plaintext from public information.

The only practical way we know how to break RSA is to find p and q from N, and then compute d by solving the equation de = 1 mod (p − 1)(q − 1).

▶ If we can factor N (that is, we can find p and q) then we can calculate d.

▶ If factorisation of N is not hard then RSA can be broken.

The problem of factoring gets harder and harder as N gets bigger.

▶ 2005: 500-bit numbers were the largest which could be factored.

▶ 2009: RSA 768-bit was factored.

▶ 2020: RSA250 (829-bit) was factored.

Typical length of N: 1024 bits, 2048 bits, 4096 bits.

A cryptosystem is provably secure if its security relies on the hardness of a certain mathematical problem.

▶ Example: Finding the private key given the public key in RSA is equivalent to factoring a large integer.

Can an attacker with limited computational power learn any information about a plaintext given the ciphertext? (Roughly, “semantic security”.)

Given a ciphertext which is an encryption of two distinct plaintext chosen by the cryptanalyst, can the cryptanalyst tell which plaintext resulted in that ciphertext?

▶ If not, this is roughly what is known as “indistinguishability” of encryption.

▶ Indistinguishability gives semantic security.

▶ How does RSA perform under these notions of security?

Textbook RSA is deterministic: same plaintext always encrypt to the same ciphertext.

Suppose the attacker can choose two distinct plaintext m1, m2.

The attacker submits them to the challenger, who encrypts one of the plaintext and returns the ciphertext c.

The attacker has to determine which plaintext was encrypted.

The attacker simply encrypts m1 and m2 to see which one results in c.

###### RSA-OAEP

In practice, we do not deploy Textbook RSA.

Most critical alteration: introduce randomization to make RSA probabilistic.

Well-known way to do so: RSA-OAEP (Optimized Asymmetric Encryption Padding).

Assume you have M, (N, e) and two hash functions h1 and h2.

1. Select random r and compute A = h1(r) ⊕ M.

2. Compute B = h2(A) ⊕ r .

3. C = (A||B)e mod N.

To decrypt C:

1. Decrypt C using d: Cd = A||B

2. Hash A using h2 and XOR to B: B ⊕ h2(A) = r

3. Hash r using h1 and XOR to A: A ⊕ h1(r) = M

###### Properties

RSA is a special type of block cipher.

Typically much slower to implement than symmetric key block ciphers such as DES or AES.

Unsuitable for encrypting long messages.

Typically used for encrypting session keys for symmetric key ciphers.

##### Other public key cryptosystems

Most systems have been shown to have security flaws.

RSA remains a widely-used and well-trusted scheme.

In the long run, elliptic curve and lattice-based schemes look set to eclipse RSA.

###### The ElGamal public key cryptosystem

Second public key encryption method by ElGamal (1979).

▶ Encryption and decryption use modular exponentiation.

▶ Security based on discrete logarithm problem.

Properties of ElGamal:

▶ Randomised: the same plaintext does not give the same ciphertext.

▶ Ciphertext expansion: ciphertext is twice the length of plaintext.

Primitives

In Zp where p is a prime, a primitive element is a number whose powers give every nonzero element of Zp .

▶ For example, in Z7, the powers of 3 (mod 7) give all the nonzero elements:

31 = 3, 32 = 2, 33 = 6, 34 = 4, 35 = 5, 36 = 1.

▶ Another example: The primitive elements of Z13 are 2, 6, 7, 11.

Each nonzero element a of Zp has an inverse a−1:

▶ a × a−1 = 1 (mod p).

▶ For example, in Z7, 2−1 = 4 because 2 × 4 = 1 (mod 7).

The discrete logarithm problem

Prime p and primitive element α.

▶ Given β ∈ Zp, find integer a (0 ≤ a ≤ p − 2) such that β = αa mod p.

▶ a is the discrete logarithm of β to the base α, write a = logα b.

For example, 7300 = 600 (mod 601), so we say that 300 is the discrete log of 600 to the base 7, and write 300 = log7 600.

Finding discrete logarithm is believed to be hard if prime p is carefully chosen.

###### Elliptic curve cryptography

Elliptic curve cryptography (ECC) introduced in the mid-1980s by Miller and Koblitz.

▶ An elliptic curve E is the set of solutions (x, y ) to an equation of the form y 2 = x3 + Ax + B, together with an extra point ∞, the point at infinity. Define “addition” of points to turn E into an abelian group with identity ∞. (NOT EXAMINABLE.)

▶ Can be used in cryptosystems based on discrete log such as ElGamal.

▶ Offers comparable security to “classical” cryptosystems with much larger key sizes.

###### Lattice-based cryptography

A lattice is a set of points in n-dimensional space with regular repeated patterns (like a grid).

▶ You can describe this with a basis.

Typical problem: the Shortest Vector Problem.

▶ Given a lattice with an arbitrary basis, find the shortest nonzero vector in it.

▶ This is a hard problem and so far not susceptible to cryptanalysis using quantum algorithms

###### Post-quantum cryptography

Quantum computers can factor large integers and solve the discrete log problem efficiently.

▶ If scalable, fault tolerant quantum computers are built then public key cryptography based on these hard problems would be broken.

▶ Post-quantum cryptography: study cryptosystems based on computational problems not susceptible to quantum computing.

▶ Lattice-based cryptography is one such candidate.

##### Implementation issues

▶ Prime selection e.g., large RSA primes p, q.

▶ Efficient algorithms for intensive computation.

▶ Asymmetric schemes generally much slower than symmetric schemes.

▶ Not usually used to encrypt bulk messages.

▶ Often used for encrypting keys for symmetric schemes.

In practice many things need to be carefully specified to ensure security

e.g., the padding applied before encryption.

###### Impersonation

Communications between Oscar and Alice:

▶ Oscar → Alice: Bob’s supposed public key e′.

▶ Alice → Bob: message encrypted with e′.

▶ Oscar decrypts message with private key d′.

Communications between Oscar and Bob:

▶ Oscar → Bob: message encrypted with Bob’s public key e.

Alice and Bob believe message is secret but Oscar reads everything.

Alice must be sure Bob’s public key is authentic.

Symmetric cryptography: parties need secret channel to agree to a key.

Asymmetric cryptography: parties do not need secret channel to agree key.

Key distribution problem replaced by authenticity issue

▶ Usual solution: certificates

Key certificates

One solution is to certify public keys.

Trusted Third Party (TTP) issues key certificates:

▶ TTP checks Bob’s identity carefully.

▶ TTP certifies Bob and his public key.

▶ Alice can check Bob’s certificate.

▶ TTP can “sign” Bob’s key.

▶ TTP can use own public key to do so.

▶ Who certifies TTP’s public key?

▶ Hierarchy of TTPs

▶ Public key verification is a major issue in implementation of asymmetric schemes.

# Integrity mechanisms

Due to active attacks, receiver needs to be assured of

▶ data integrity

▶ data origin authentication (also known as message authentication)

▶ entity authentication(also known as identification)

Non-malicious threats:

▶ noise and accidental errors

Non-cryptographic means (no secret keys or secure channels):

▶ Parity check

▶ Checksums: generalisation of parity check

▶ Error-correcting codes

Active attacks:

▶ Alteration or replacement of a message;

▶ Deletion of a message;

▶ Insertion of a false message;

▶ Replay of an old message;

▶ Changing the order of a message;

▶ Falsifying the origin of a message.

## Manipulation detection codes (MDCs)

A Manipulation Detection Code (MDC) is a special type of integrity mechanism designed for use with a cipher. The MDC is a fixed length string computed as a function of the whole data string (no key is used). The MDC is concatenated with the data and then the combination is encrypted.

MDC functions must be chosen carefully. For example, simply XOR-ing the message blocks together is likely to lead to problems. MDCs must be used with the right type of cipher. For example, MDCs do not work well when used with a stream cipher.

Hash functions

A cryptographic hash function h is a function that takes a message of arbitrary length and produce a hash value or hash code (or simply hash) or message digest of a specific size.

Used in integrity mechanisms e.g. HMAC and digital signatures (non-repudiation and integrity). Also used in commitment, message authentication, key derivation, etc..

Hash functions are simply a special type of MDC. They are considered a very versatile and useful tool in cryptography (binding data, source of pseudorandomness, ...).

#### Example of typical usage

For a file or message m, a hash is computed: x = h(m). The integrity of the hash x is protected (but not m).

Later, to check whether m has been altered: We calculate the hash of the file/message again to get x′. We compare x′ to see if it is equal to x. If x′ = x then we accept that m has not been altered. Problem of protecting the integrity of a large file/message becomes the problem of protecting the integrity of a small value.

#### Attacks

An adversary finds another message m′ such that h(m′) = h(m). The adversary replaces m with m′. The replacement will not be detected by comparing the hash.

An adversary finds two messages m, m′ such that h(m′) = h(m) = x. The adversary may store message m with x. Later the adversary may claim that the original message was m′.

Such an m′ can exist because h is a many-to-one function. We need m′ to be difficult to find. We can have at most computational security.

#### Properties

Hash functions should be:

▶ easy to calculate

▶ One-way (preimage resistant) hash function.

▶ Second preimage resistant hash function.

▶ Collision-resistant hash function.

##### Preimage Resistance

The function is one-way.

Given z, it is difficult to find any m with z = h(m).

Given z = h(m) it is not only difficult to find m, it is also hard to find any value that will hash to z.

Potential application:

▶ Password storage.

##### Second Preimage Resistance

Given x, it is difficult to find x′ != x such that h(x′) = h(x).

Given m and z = h(m) it is difficult to find another value m′ that will hash to z.

Potential application:

▶ Data integrity against accidental errors.

▶ Combine with other mechanism to provide stronger data integrity protection.

##### Collision Resistance

It is infeasible to find m, m′ such that h(m) = h(m′).

Collision resistance implies second pre-image resistance.

Potential application:

▶ Cryptographic commitment: should not be able to claim to commit to another value.

▶ Digital signature.

#### Issues

Birthday paradox attacks: If n possibilities then list of length √n has over 50% chance of a match.

(if you had a list of possible hashes that is of length √n, there is over a 50% change of it having the correct hash)

Practical hash functions at least 160 bits; Collision not expected for 280 hashes.

Problems found with many hash functions - hash functions are often weakness in signatures

#### Hash function construction

Hash function can be constructed using block ciphers. CBC mode; Last block is hash value for the message.

Dedicated hash function designs:

▶ SHA-3 Keccak.

▶ MD5.

Hash functions do not have a key (typically). Hash functions are publicly computable. (Anyone can compute it / knows how to, the function is not private).

##### Dedicated Designs

Several hash functions are widely used (all iterative in structure - multiple rounds).

The Secure Hash Algorithm (SHA-1), a US NIST and ISO standard, gives 160-bit outputs.

Note that SHA-1 stands for SHA revision 1 – the original SHA algorithm (SHA-0) was modified soon after publication. SHA-0 turns out to be relatively weak.

Other widely used schemes include MD4 (broken) and MD5 (weak). (MD = Message Digest Algorithm)

SHA-1 is widely used in security applications and protocols.

In 2005, security flaws were identified in SHA-1, namely that it might be possible to find a collision in substantially less than 280 operations. Despite this, it still appears to be substantially more robust than MD5 and is in use.

After publication of SHA-1 (and before problems with it were discovered), a need was foreseen for longer hash codes to be used.

▶ Series of schemes were standardised by NIST and ISO, with hash value lengths up to 512 bits.

▶ Schemes are collectively known as SHA-2.

▶ No known issues, but, since they use similar concepts to SHA-1, the premise is that they will eventually need replacing.

A new hash standard, SHA-3 - KECCAK, was recently developed. KECCAK selected to be SHA-3 through a competition like AES.

## Message authentication codes (MACs)

MAC: Message Authentication Code, a cryptographic checksum.

A MAC takes as input a message m and a key k and outputs a MAC or check value. This output is sent with the message.

We assume that the message is sent in the clear. A MAC only provides data origin authentication (message authentication). Therefore it provides data integrity. If confidentiality is also required, then the message will need to be encrypted.

How they work

1. Sender computes MAC, fk (m), where f is the MAC function, k is the secret key and m is the message.

2. Sender sends m||MAC (where || denotes concatenation of data items).

3. Receiver recomputes the MAC over the message and check it matches.

If the value calculated by the receiver matches the MAC value sent by the sender, the receiver may accept that data origin authentication has been provided.

▶ The data was created at some time in the past by the sender (unknown time).

▶ The data has not been altered.

If the value calculated by the receiver does not match the MAC value sent by the sender?

▶ The receiver knows then that some part of the transmitted information has been tampered with.

Security of MACs

We will always assume that an attacker knows the MAC algorithm. Security relies on the security of the key. What are the security objectives of MACs?

▶ Can you detect if someone has changed part of the message without authorization?

▶ Can you detect if someone has deleted part of the message without authorization?

▶ Can you detect if someone has inserted part of the message without authorization?

▶ Can you detect if someone has changed the origin of the message?

Suppose the attacker has several (message, MAC) pairs for the same key.

Possible attack objectives:

▶ Key recovery: learn secret key – strongest possible attack;

▶ MAC forgery: work out the correct MAC for a new message.

Types of MAC forgery:

▶ Selective forgery; Adversary can produce a (message, MAC) pair where the message is under the adversary’s control. If the adversary obtains the key then this is possible.

▶ Existential forgery: Adversary can produce a (message, MAC) pair but has no control over what the message is.

In some cases even existential forgery can be damaging.

Messages requiring message origin authentication tend to be highly structured and have high redundancies.

MAC Mechanisms

Various types of MAC mechanisms. Best known, and probably most widely used, are the CBC-MACs.

#### CBC-MACs

Generated using a block cipher in CBC mode.

CBC-MACs are the subject of international standards. ISO has produced a standard, ISO/IEC 9797-1, for CBC-MAC schemes. We will look at one of the options described in this standard.

To get a n-bit block cipher to give an m-bit MAC (m ≤ n):

1. Pad data to form a series of n-bit blocks,

2. Encrypt data using CBC mode,

3. Take final block as the MAC, after optional processing and truncation (if m < n).

A diagram of a diagram

Description automatically generated

##### CBC-MAC forgery; cut-and-paste attack

The security of CBC-MAC relies on the key k

(and the security of the block cipher).

Suppose CBC-MAC is used with no processing or

truncation. Suppose you see two valid message-MAC pairs

(m1, MAC1), (m2, MAC2):

MAC1 = ek(m1), and MAC2 = ek(m2)

Can you forge a valid MAC on a different message?

(Existential) forgery: MAC2 is a valid MAC on

the two-block message m′1, m′2

m′1 = m1

m′2 = m2 ⊕ MAC1.

The optional process can prevent this.

#### Padding

Three possible padding methods:

Method 1: add as many zeros as necessary to make a

whole number of blocks. This does not allow detection of

addition or deletion of trailing zeros (unless message

length known by recipient).

Method 2: add a single 1 followed by as many 0s as

necessary to make a whole number of blocks.

Method 3: as Method 1, and also add an extra block

containing the unpadded message length.

#### Optional processes

Two possible optional processes applied to Oq are:

Choose a key k1 and compute: O′q = ek(dk1(Oq)).

Choose a key k1 and compute: O′q = ek1(Oq).

Optional processes can make it more difficult for a cryptanalyst to do an exhaustive search for the key k. Both optional processes prevent attacks where MACs on a number of messages can be combined to compute a valid MAC on a bogus message.

#### Hash-functions

Collision-resistant cryptographic hash-function can be used to build a MAC function.

One possibility way is to concatenate the key with the message and then apply a hash-function;

MAC = h(K ||M).

HMAC (Hash-based MAC) is slightly more complicated (for protection against attacks).

HMAC = h( K1 || h( K2 || M )), where K1 and K2 are two variants of a secret key.

It is specified in Internet RFC 2104 and ISO/IEC 9797-2.

Encryption

Many cryptographic applications require both integrity and confidentiality. This combination of security services is known as authenticated encryption.

Typically, in these scenarios, we refer to message, requiring confidentiality and data origin

Authentication & associated data, requiring only data origin authentication & full message is the combination of message and associated data.

One approach to achieve authenticated encryption is to use two separate cryptographic primitives, i.e., a MAC and an encryption scheme (with different keys).

MAC-then-encrypt

1. MAC computed over full message.

2. Message and MAC encrypted (not associated data AD).

3. Ciphertext and AD sent to receiver.

Encrypt-then-MAC

1. Message encrypted.

2. MAC computed over ciphertext and AD.

3. Ciphertext, AD and MAC sent to receiver.

A second approach is to use a single cryptographic primitive providing both confidentiality and integrity.

This has several advantages:

▶ Key management (now using only one key!)

▶ Processing costs (now running a single dedicated primitive)

▶ Security (designs which can provide stronger security guarantees)

Notable example: GCM (Galois counter mode) (Section 6.3.6 of Everyday Cryptography.)

# Entity authentication and identification

Entity authentication gives an entity assurance of the identity of the entity it is communicating with, and that this entity is currently participating.

▶ Unilateral authentication: gives one entity assurance of the other’s identity, but not the other way round.

▶ Mutual authentication: provides both entities with assurance of each other’s identity.

We are interested in controlling access to communication and information resources over potentially insecure networks.

Entity authentication provides a fundamental service:

▶ It allows one host/user on the network to check with which other host/user it is communicating.

▶ Successful entity authentication can be a precursor to the use of more complex security services.

An identity: the individual attributes by which a person or thing is recognised.

A realm of identity: a context for which particular attributes apply.

A digital identity: links real world attributes to a ”cyber world” identity.

Need for digital identities:

▶ Control access to secure areas.

▶ Determine permitted actions (authorisation).

▶ Track actions in and across systems (?)

Identification: the ability to claim a unique identity.

▶ For example, a user enters a username and the system recognises the username as a legitimate one.

Authentication (identity verification): the ability to prove the claimed identity.

▶ For example, after the user enters a username the user enters a password.

▶ The system authenticates the user by verifying that the password is correctly associated with that username.

We can classify identity verification methods into four types:

▶ By something known.

▶ By something possessed.

▶ By physical characteristics.

▶ By result of involuntary action.

Multi-factor authentication is the use of two or more methods in a single authentication transaction.

We consider first two (something you have) and last two (something you are) together.

## Schemes and techniques for entity authentication

Verification by something known or possessed:

▶ Passwords/PINs.

▶ Challenge-response schemes.

▶ Tokens.

Verification by physical characteristics:

▶ Biometrics

Vary in efficiency, involvement of trusted third parties (TTPs), security guarantees, storage of secrets.

Password schemes

Usual schemes to access computer systems.

▶ User and system share a secret password.

▶ Password typically 6 to 10 characters.

▶ To gain access, user enters (userid, password).

▶ System checks password against stored data for that userid.

How should lists of passwords be stored?

▶ If they are stored unprotected, then they are readable by privileged users or hackers.

▶ Usual solution – hide them using a one-way function (easy to compute, difficult to invert).

▶ Check password by applying function and comparing with list entry.

Unix uses a one-way function to protect its password list:

▶ Slow ”encryption” (25 iteration of modified-DES (now replaced)).

▶ Password salting.

Aim: to make the derivation of user passwords from possession of the password file as difficult as possible.

▶ Slow ”encryption” slows down attacks involving large number of trial passwords.

▶ Salting also makes pre-”encrypted” dictionary attack difficult and prevents entire list being attacked simultaneously.

#### Password salting

A password is modified by a random t-bit string called a salt before the one-way function is applied.

▶ The salt is stored in the password table with the output of the one-way function.

▶ When the user enters a password the system looks up the salt and applies the one-way function to the password as modified by the salt.

Salting does NOT increase the difficulty of exhaustive search on a particular user’s password.

▶ It prevents a dictionary attack against a large set of passwords simultaneously.

▶ The dictionary has to store 2t variations of each trial password.

#### PINs

An example of a password scheme

e.g., ATM cards.

▶ Userid is account information (stored on card), password is PIN (not stored).

▶ PVV (pin verification value) is a cryptographic function of userid & PIN. (PVV stored on card).

▶ To gain access, user enters (userid, PIN) by using card.

▶ System checks PVV against userid and PIN.

#### Drawbacks

▶ Replay attack.

▶ One-way identification: system does not identify itself to user.

▶ Exhaustive search (online): try passwords one at a time on actual system.

▶ Exhaustive search (offline): obtain “encrypted” password file, “encrypt” trial password and compare.

▶ Dictionary attack:

Create “encrypted” list of high-probability passwords.

Compare captured “encrypted” passwords from password file.

Dictionary can be used repeatedly.

If passwords are sent across insecure channel then they are vulnerable to interception.

▶ Simple encryption is no help: encrypted password could be replayed.

▶ One solution is to use a challenge-response process.

Strong authentication

In strong authentication, an entity ”proves” its identity to another by showing knowledge of a secret known to be associated with that entity without revealing the secret itself.

▶ Example: ”Challenge-and-response” authentication protocols with cryptographic mechanisms to protect messages in the protocols.

#### Challenge-and-response schemes

Alice proves her identity to Bob by demonstrating knowledge of a secret associated with her.

▶ Knowledge is demonstrated by responding to time-varying challenge.

▶ Response depends on the challenge and Alice’s secret.

##### Freshness and liveness

Replay attacks show that data origin authentication and integrity checking is not enough.

▶ Need a way of checking message freshness and liveness of principals.

▶ Message freshness: assurance that message has not been used previously, and originated within an acceptably recent timeframe.

▶ Liveness of principals: assurance that a principal has sent a protocol message within an acceptably recent timeframe.

▶ Liveness = message origin authentication + freshness.

▶ Method: use time-variant parameters.

###### Time-variant parameters

”Nonce”: ”number used once”.

A value that is used no more than once for the same purpose.

▶ Verifiable timeliness, uniqueness, identification of sequence of messages.

Main types: random numbers, sequence numbers, time stamps.

▶ Different properties and uses.

Random numbers provide uniqueness, timeliness, unpredictability & prevents replays.

Time stamps provide uniqueness and timeliness (detect replays and forced delays).

Requires securely synchronised clocks - non-trivial.

▶ Typical clock drift: one second per day on work station.

▶ Need window of acceptance for time difference of received message, to account for clock drift and message propagation time.

▶ Need a log of recent received messages to prevent replay attacks within window of acceptance.

A sequence number can be used to identify a message (Detect replays but not forced delays).

Can be used as ”logical time stamps”:

▶ Alice and Bob use a pair of sequence numbers NAB and NBA.

▶ When Alice sends Bob a message she includes the current value of NAB and increments it.

▶ Likewise for Bob with NBA.

▶ Need a pair of secret sequence numbers for every pair of communicating parties.

##### General password-based scheme

A close up of black text

Description automatically generatedIf M is an attacker we write M(A) to mean that M masquerades as A.

One-way function f must have property that knowledge of f(R, P), R and f do not reveal P.

▶ Insecure if not enough passwords (brute force search).

▶ Users must have means to compute f reasonably quickly.

##### Symmetric technique

Assume that Alice and Bob share a key K .

The claimant corroborates its identity by demonstrating knowledge of shared secret:

▶ In this case the shared secret is K.

▶ The demonstration of knowledge is the ability to encrypt a challenge using that key.

A → B : ”Hi Bob I’m Alice”

B → A: R (challenge)

A → B : eK(R||iB) (response)

where iB is a public identifier for B

##### Tokens

Challenge-response schemes are widely deployed in token-based technologies.

Tokens are well-established idea:

▶ Keys for doors, cabinets, cars, etc.

▶ Magnetic stripe cards, smart cards.

Problems with copying/cloning/forgery.

Alternatives: calculator-like devices with:

▶ Key-pad and display,

▶ Key/password storage,

▶ Cryptographic calculation facility.

###### Magnetic stripe cards

Very widely used. ISO/IEC 7810 specifies card dimensions and magnetic stripe format.

User ID on magnetic stripe.

▶ Usually used with PIN.

Off-line systems - PIN check data on card.

On-line systems - PINs verified centrally.

▶ Problems arise because of easy forging/copying.

▶ Hologram (on card) added to prevent changing embossed data.

▶ Many schemes devised to make forging/copying difficult.

###### Smart cards

Cards with on-board micro-processor, RAM and ROM.

More memory than magnetic stripe cards.

Communicate with reader via plated areas on card.

Copying much more difficult.

Positions/protocols standardised in ISO/IEC 7816, a multi-part standard.

1st generation cards had primitive processors and limited memory (8 kbytes).

2nd generation IC cards have more powerful processors and more memory.

If IC card contains cryptographic function, they can then be used in an identification process (e.g. challenge-response).

▶ Typically they also require PIN entry.

▶ Increasing range of applications as the available processing power and memory grow.

▶ Routinely used for debit/credit cards.

▶ Used widely in mobile phones to store user identity and user secret keys.

###### Hand-held ID devices

Calculator-like devices with:

▶ key-pad and display,

▶ key/password storage,

▶ cryptographic calculation facility.

No card reader required.

###### Use of tokens in challenge-response

The User and the Server share a secret key K .

The token contains the User PIN, K, and the encryption function shared with the Server.

An authentication attempt might go as follows:

1. The Server issues a challenge.

2. The User enters PIN into token.

3. If the PIN is correct the token is activated and the User enters the challenge from the Server.

4. The token encrypts the challenge using K and displays the result to the User.

5. The User responds to the Server with the result.

6. The Server checks that the challenge and the result are valid.

If so the User is authenticated.

###### S/KEY: Password generation

S/KEY is a public domain one-time password scheme (Internet RFC 1760).

▶ Based on repeated application of a one-way function f on a secret key S.

Notation: We write fk to mean apply f a total of k times.

The host and user agree on a 64-bit key S.

▶ S is derived from a password the user remembers.

Apply the one-way function f N times to S to get N one-time passwords:

▶ fN(S) is the 1st password.

▶ fN−1(S) is the 2nd password. And so on.

The host stores the values k and fk(S), k ≤ N. The host does not store S.

▶ When the user identity is to be verified, the host subtracts one from its current value of k, and sends this value as the challenge c = k − 1.

▶ The user responds with r = fc (S).

▶ The host verifies by checking whether f(r) = fk(S).

▶ If this check works, then the host replaces its stored value of fk(S) with fc(S). (decremented iteration)

▶ The secret S is not stored long-term in either the host or user systems.

###### Time-based 1-time passwords

Another well-established idea is to use a clock to generate one-time passwords

▶ At regular intervals, the clock value and secret key are input to a one-way function to generate a one-time password.

▶ Host will accept one password ‘either side’ of the current one. (To account for timing)

A diagram of a device

Description automatically generated

SecurID

SecurID: token generating one-time passwords as a function

A close-up of a security card

Description automatically generatedof an internal clock and a secret.

##### Asymmetric technique

RSA with modulus n.

▶ Alice has certified public key exponent e.

▶ Bob sends “Alice” a challenge y = xe mod n.

▶ “Alice” sends back yd mod n.

▶ If yd = x mod n then “Alice” knows d and Bob accepts “Alice” as genuine

Biometrics

Passwords may be revealed or guessed.

Tokens may be lost or stolen.

Personal characteristics may be harder to forge and have long history of use.

Device that measures characteristics must be trusted (e.g. physically secure) otherwise replay may be possible.

Biometrics is the measurement and statistical analysis of biological data.

▶ In IT, biometrics refers to technologies for measuring and analysing human body characteristics (physiological or behavioural) for authentication and/or identification purposes.

Definition by Biometrics Consortium: automatically recognising a person using distinguishing traits

▶ No human involvement.

▶ Comparison takes place in real time.

#### Distinguishing traits

Ideal identifier (distinguishing trait) should possess following properties:

▶ Universality: Nearly all people in the target population should have the characteristic.

▶ Uniqueness: The characteristic of everyone should be unique.

▶ Stability: The characteristic should neither change with time nor allow alteration.

▶ Collectability: The ease at which a person’s trait can be acquired, measured quantitatively, or processed further.

Other issues to be considered when a biometric system is being developed:

▶ Performance: The efficiency of the system in terms of accuracy, speed, and memory requirements.

▶ Acceptability: People should be willing to accept the biometric system in the particular application environment

▶ Forgery resistance: Not easy to fool the biometric system.

#### Biometric ATM example

Identification system using biometrics:

▶ You approach an ATM with NO card, NO claimed identity, NO PIN.

▶ The ATM scans your face and determines who you are and gives you access to your money.

Authentication system using biometrics:

▶ You approach an ATM and swipe a card or enter an account number.

▶ The ATM scans your finger print and uses it as a password to authenticate you as the rightful owner of the card and therefore give you access to your money.

#### Biometric system architecture

Major components of a biometric system:

▶ Data collection (make the measurement, capture biometric data presented);

▶ Signal processing (feature extraction and coding);

▶ Storage (for reference values (templates));

▶ Matching (compare measurement with reference);

▶ Decision (yes/no for authentication; identity for identification);

▶ Transmission (of measurements and decision).

A diagram of a decision making process

Description automatically generatedUser’s template: set of digital features

extracted from biometric input

#### Security of enrolment

Requirements for enrolment:

▶ Secure enrolment procedure;

▶ Users should use

authorised enrolment site.

▶ Verification of identity of

the user being enrolled.

▶ The enroller has proper

permission to access enrolment functions.

▶ Binding of the biometric template to the user being enrolled;

▶ Check of template quality and matchability.

Other considerations:

▶ Whether user already enrolled;

▶ Possible similarity to existing templates.

#### Transmission system

Subsystems are logically separate. Usually, there are separate physical entities in a biometric system.

Some subsystems may be physically integrated.

▶ Biometric data has to be transmitted between the different physical entities.

▶ Biometric data is vulnerable during transmission.

#### Data collection subsystem

Sensor reads biometric information from the user and converts it into a suitable form for processing.

▶ Sensors must be similar, so biometric features are measured consistently at all sensors

Must be able to deal with changes in data collection:

▶ Biometric feature may change.

▶ Presentation of the biometric feature at the sensor may change.

▶ Performance of the sensor itself may change.

▶ Surrounding environmental conditions may change.

#### Signal processing subsystem

Performs ‘feature extraction’.

▶ Receives raw biometric data from the data collection subsystem.

▶ Extract distinguishing features from the raw biometric data.

▶ Transforms the data into the form required by the matching subsystem.

▶ Filtering may be applied to remove noise.

#### Storage subsystem

Maintains reference templates for enrolled users.

▶ One or more templates for each user.

The template could be stored in:

▶ Physically protected storage within the biometric device;

▶ Conventional database;

▶ Portable token, such as a smartcard.

Collateral information (such as name, identification number, etc) binding the owner to their reference template may also be stored together with the reference template.

#### Matching subsystem

Receives processed biometric data from signal processing subsystem & biometric template from storage subsystem. Measures similarity of the claimant’s sample with the reference template.

E.g. uses distance metrics, probabilistic measures, neural networks, etc.

▶ Result is a number - the match score.

▶ Indicates how closely the sample and the template match.

#### Decision subsystem

The decision subsystem interprets the match score from the matching subsystem: A threshold is set: if the match score is above the threshold, the user is authenticated. Otherwise the user is rejected.

▶ Typically it is a binary decision: accept or reject.

▶ May require more than one submitted sample to reach decision.

Possible outcomes:

▶ A legitimate claimant is accepted.

▶ A legitimate claimant is rejected.

↑ False rejection/false nonmatch (Type I error).

▶ An impostor is rejected.

▶ An impostor is accepted.

↑ False acceptance/false match (Type II error).

Balance needed between the two types of error. Application and context dependent trade-off between them.

##### FRR and FAR

False Rejection Rate FRR: the probability of identifying a legitimate user as an imposter.

Total false rejections

Total rejections

False Acceptance Rate FAR: the probability of identifying an imposter as a legitimate user.

Total false acceptances

Total acceptances

Decision threshold: a trade-off between FRR and FAR:

Lower FRR means higher FAR and vice versa.

Equal error rate (EER): FRR = FAR.

A line graph with a point

Description automatically generated with medium confidenceType I/FRR error curve Type II/FAR error curve ERR

A diagram of a normal and sharpness curve

Description automatically generated with medium confidenceA diagram of a function

Description automatically generated with medium confidence

#### Static vs dynamic biometrics

Static (also called physiological) biometric methods:

▶ Identification/authentication based on a feature that is always present and are mostly associated with physical shape of the persons’ body.

Example: Fingerprint recognition, retinal scan, iris scan, hand geometry, face scan.

Dynamic (also called behavioural) biometric methods:

▶ Identification/authentication based on a certain behaviour pattern associated with a certain person.

Example: Speaker recognition, keystroke dynamics

Regardless of technology, it is important to make sure that input at biometric sensor originates with live user. This is known as liveness detection.

#### Fingerprint recognition

Ridge patterns on fingers uniquely identify people.

Classification scheme devised in 1890s.

▶ Major features: arch, loop, whorl.

Each fingerprint has at least one of the major features and many ”small features” (so called minutiae).

▶ In an automated system, the sensor must minimise the image rotation.

▶ Locate minutiae and compare with reference template.

Universality: common (but with exceptions).

Uniqueness: high inter-user differences.

Stability: individual stability, though may be affected by skin condition and injury.

Collectability: simple and easy to use, non-intrusive.

Performance: sensors are comparatively low cost.

Acceptability: high acceptance due to simplicity, but association with forensic applications (crime detection).

Accuracy: high - claims of over 98% probability of correct authentication.

Liveness detection is important (detached real fingers and gummy bear fingers).

#### Eye biometrics

Two main types of eye biometric:

Iris scanning:

▶ Iris is the coloured portion of the eye surrounding the pupil.

▶ Complex iris pattern is used for authentication.

Retinal scanning:

▶ Retinal vascular pattern on the inside of the eyeball.

▶ Pattern of blood vessels used for authentication.

Retinal scanning:

Universality: common.

Uniqueness: high inter-user differences.

Genetic independence: identical twins have different retinal pattern.

Stability: highly protected internal organ, but may change during the life of a person.

Collectability: requires close proximity to sensor in appropriate position with no movement.

Performance: requires more complex sensors, but small templates allow efficient verification.

Acceptability: intrusive, perceived health threat.

Accuracy: potential for high accuracy.

#### Signature recognition

Handwritten signatures are an accepted way to authenticate a person.

▶ Automatic signature recognition measures the dynamics of the signing process.

▶ Signature generating process is a trained reflex and imitation is difficult, especially in real time.

Dynamic signature recognition: A variety of characteristics can be captured:

▶ angle of the pen;

▶ pressure on the pen;

▶ total signing time;

▶ velocity and acceleration;

▶ geometry.

Universality: common (but with exceptions).

Uniqueness: high inter-user differences.

Stability: variability between signing, evolve with time, affected by external and emotional condition.

Collectability: simple and easy to use.

Performance: sensors are comparatively low cost, but largish templates.

Acceptability: high acceptance due to traditional role, perceived to be non-intrusive.

Accuracy: not high, not resistant to professional forgery.

#### Speaker verification

Linguistic and speaker dependent acoustic patterns.

Speaker’s patterns reflect:

▶ anatomy (size and shape of mouth and throat),

▶ behavioural (voice pitch, speaking style).

Heavy signal processing involved (spectral analysis, periodicity, filtering etc.).

Three main types:

▶ Text dependent: predetermined set of phrases for enrolment and identification;

▶ Text prompted: fixed set of words, but user prompted to avoid recorded attacks;

▶ Text independent: free speech, more difficult to accomplish.

Universality: common (but with exceptions).

Uniqueness: high inter-user differences.

Stability: variability of voice, evolve with time, affected by external and emotional condition.

Collectability: use of existing telephony infrastructure, non-intrusive, easy to use.

Performance: sensors are comparatively low cost, but large templates, affected by background noise.

Acceptability: high acceptance, no negative association.

Accuracy: low, may be subject to pre-recorded attacks.

#### Choosing biometrics

▶ Does application need identification or authentication?

▶ Is the collection point attended or unattended?

▶ Are the users used to the biometrics?

▶ Is the application covert or overt?

▶ What are the storage requirement constraints?

▶ How strict are the performance requirements?

▶ What types of biometrics are acceptable to the users?

#### Biometric authentication issues

Technically complex and expensive.

Risks:

▶ Fake features: replacement with fake fingers, photos, voice recordings.

▶ Replay of recorded signal, bypassing the sensors.

▶ There must be means for a user to withdraw existing record and re-register using new (?) biometric information, in the case of an attacker obtaining biometric information of a legitimate user.

Common defence:

▶ Multimodal biometric system.

▶ Liveness detection: making sure that input at sensor originates with live user.

▶ Storing/transmitting sensitive information securely.

## Modern identity systems

The Laws of Identity

In May 2005 Kim Cameron, Architect of Identity, Microsoft Corporation, put forward 7 laws of identity.

▶ To provide a reliable way to establish who is connecting with what on the internet.

▶ <https://www.identityblog.com/?p=352>

Focus of the first three.

1. User control and consent.

2. Minimal disclosure for a constrained use.

3. Justifiable parties.

4. Directed identities.

5. Pluralism of operators and technologies.

6. Human integration.

7. Consistent experience across contexts.

#### 1. User control and consent

Technical identity systems must only reveal information identifying a user with the user’s consent.

▶ Promote user trust in the system.

▶ Provide privacy control.

#### 2. Minimal disclosure for a constrained use

The solution which discloses the least amount of identifying information and best limits its use is the most stable long term solution.

▶ Principles of ‘need-to-know’ and ‘need-to-retain’.

▶ Limit the impact/loss in event of security breach.

▶ Limit the risk, by minimising benefit for attackers

#### 3. Justifiable parties

Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship.

▶ Promote user trust in the system.

▶ Apply to identity providers – principals managing and releasing user information.

▶ Apply to relying parties – principals with dependencies on user information

Identity systems

Identity federation

▶ Links together user accounts from various systems.

▶ Allows a user to use the same identification data to access multiple systems across different enterprises.

Single sign-on (SSO)

▶ Allow users to access multiple services from only a single set of authentication credential.

▶ Can be a component of identity federation.

User/user agent: owner of digital identity/resources.

Service provider/relying party :

▶ provides services to users.

▶ requests authentication decision from identity provider.

Identity provider :

▶ manages identity information for other principals.

▶ provides methods to authenticate a user and generates claims regarding the attributes of the user.

#### SSO (Single sign-on)

##### Trust establishment

Manual registration: service provider visits identity provider and obtains identity provider information (such as its certificate).

Dynamic registration (eg. OpenID, OAuth):

1. User requests login on service provider (eg. sp.com) by submitting identifier (eg. user@idp.com).

2. Service provider identifies identity provider (idp.com) from user identifier.

3. Service provider dynamically registers with identity provider.

Dynamic registration may require more steps to ensure security.

##### Authentication token generation

1. The service provider forwards the user to the identity provider

(e.g., a HTTP redirect to a URL on idp.com).

2. The user logs in at the identity provider who gives the user an SSO token.

3. This token is given to the service provider.

##### Authentication token redemption/validation

The service provider receives the SSO token and uses it to confirm the authentication of the user.

▶ The token typically would contain the user’s identity, the recipient (the service provider), and some freshness parameter.

# Case study – GSM authentication

Global System for Mobile communications.

▶ ETSI (European Telecommunications Standards Institute) standard for 2G digital cellular networks.

▶ Security for digital mobile telecommunications.

▶ Mobile station (MS): Mobile phone.

▶ International mobile subscriber identity (IMSI): A number that uniquely identifies a user.

▶ Subscriber identity module (SIM): A smart card containing a user’s subscription information.

▶ Base station (BS): Handle traffic and signalling between a mobile phone and the network.

▶ Home public land mobile network (HPLMN): Home network.

▶ Home location register (HLR): A central database containing details of every SIM card issued by the mobile phone operator.

▶ Authentication centre (AuC): Handles authentication of SIMs.

▶ Visitor location register (VLR): A database of the MSs that have roamed into the network.

A diagram of a network

Description automatically generated

## GSM security features

Security requirement: as ”secure” as Public Switch Telephone Network.

▶ Entity authentication of user.

▶ Confidentiality over the air between mobile station and base station.

▶ Anonymity over the air.

Mobile operators need to be able to charge the right user for services.

Anonymity over the air

This is confidentiality of the identity of the user.

▶ Prevent linking user calls.

A diagram of a network

Description automatically generated▶ Use a temporary identity (TMSI) instead of IMSI.

## GSM set up

▶ Each MS has a home network (HPLMN).

▶ HPLMN has contractual agreement with MS.

▶ MS and HPLMN share IMSI.

▶ HPLMN maintains HLR:

▶ HLR records last known location of all its MS.

▶ HPLMN operates an AuC:

▶ AuC stores secret key information relating to each MS.

▶ Each MS has a SIM which stores IMSI and key information.

▶ At subscription, an individual subscriber authentication key Ki is allocated and stored in the MS’s SIM and in the AuC.

Roaming

When MS visits a network it communicates with the BS belonging to that network across the air interface.

A diagram of a network

Description automatically generated▶ Visited network maintains VLR which registers MS.

▶ MS authenticates itself to VLR.

▶ Challenge-response type authentication.

▶ Corroboration that MS’s IMSI is what is claimed.

#### IMSI authentication

1. MS sends authentication request and IMSI to VLR.

2. VLR forwards request and IMSI to HPLMN.

3. HPLMN returns authentication triplet ( RANDi , SRESi , Kc ) to VLR.

▶ SRESi = A3Ki ( RANDi )

▶ Kc = A8Ki ( RANDi )

4. VLR stores triplet and sends RANDi to MS.

5. MS computes RESi = A3Ki ( RANDi ) and Kc = A8Ki (RANDi) and sends RES to VLR:

6. VLR checks RESi = SRESi

#### Entity authentication of MS

Unilateral rather than mutual authentication: MS authenticated to VLR but VLR not authenticated to MS.

▶ This relies on the belief that only the home network AuC and the MS know Ki.

▶ A3 is used in the challenge-response protocol and is operator specific.

▶ Typically HPLMN would supply a number of triplets to VLR so that many authentications can be performed.

▶ Note that VLR does not know Ki.

#### Confidentiality over the air

In the challenge-response protocol, another value Kc is generated:

▶ The algorithm A8 is used to generate the encryption key Kc . Kc = A8Ki (RANDi).

▶ A8 is operator specific.

▶ Kc is used for encryption using the stream cipher A5.

#### Universal Mobile Telecommunications System (UMTS)

The next generation of mobile telecommunications – more functionality.

Security features:

▶ Mutual entity authentication.

▶ Use of public algorithms.

▶ Longer key lengths.

# Digital signatures

Electronic medium: digitise written signature and append to digital document?

▶ Can be removed and appended on other document easily.

▶ Forgery indistinguishable from original.

▶ Digital signature must be message-dependent (and publicly verifiable).

Cryptographic primitive providing

▶ Data origin authentication (data integrity + identity of signer).

▶ Non-repudiation.

Digital signatures are dependent on the message and a secret signing key known only to the signer.

▶ If dispute arises (repudiation or fraudulent claimant), an unbiased third party (arbiter) must be able to resolve matter.

▶ No requirement of access to signer’s secret key.

## Digital signature scheme

There are three basic properties a practical digital signature scheme should satisfy.

1. It should be easy (and efficient) for the signer to sign the data.

2. It should be easy for anyone to verify the message/signature pair.

3. It should be hard for anyone to forge a digital signature.

A diagram of a sign

Description automatically generated

M: Set of messages.

S: Set of digital signatures, usually fixed-length binary strings.

The signer A must have a signing, verification key pair (SK, VK ).

▶ The signing key SK is private to A, so only A can sign.

▶ The verification key VK is made available to whoever wants to verify the signature.

▶ The output of the verification algorithm is some data with which the verifier decides whether the signature is valid or not.

Signature generation

Signing/signature generation algorithm:

▶ Takes the signing key SK and the message M as input.

▶ Outputs a signature SSK(M).

▶ SSK(M) is the digital signature on the message M that is sent to the receiver.

Signature verification

Verification algorithm:

▶ Takes the public verification key VK and the signature S (and sometimes the message M) as input.

▶ Outputs a public value which allows the verifier/receiver to decide whether to accept the signature as a valid signature on the message.

▶ Anyone in possession of the verification key can verify the signature.

How does the verifier know which data the digital signature matches?

Two approaches:

▶ Send the verifier the data being digitally signed.

▶ Add redundancy to the data being signed.

In practice, digital signatures are almost always of the first type.

Security

Required property:

▶ It is computationally infeasible for anyone other than the signer to find a signature such that the verification algorithm gives true verification for any message.

Breaking digital signature schemes:

▶ Total break: Oscar finds Alice’s secret signing key so can sign anything.

▶ Selective forgery: Oscar can sign certain classes of messages chosen by him.

▶ Existential forgery: Oscar can sign at least one message, not of his choosing.

#### Attacks

Key-only attacks: Oscar knows only the public verification key.

Message attacks: Oscar has signatures corresponding to known messages:

▶ Known-message attack: Oscar has signatures to messages known to him but not chosen by him.

▶ Chosen-message attack: Oscar obtains signatures to a chosen list of messages before trying to break the scheme.

▶ Adaptive chosen-message attack: Oscar can request signatures from Alice depending on previous signatures/messages.

Digital signature schemes with appendix

#### Textbook RSA

RSA can be used to create a signature scheme with appendix.

Signer (Bob) generates RSA parameters (N, e) and (p, q, d).

▶ N = p × q where p and q are large primes.

▶ e is coprime to (p − 1)(q − 1).

▶ ed = 1 mod (p − 1)(q − 1).

Bob keeps d secret as his private signing key. Bob makes (N, e) the public verification key.

Bob (the signer) applies his private signing key d to generate the signature S. S = Md mod N.

▶ The message M and signature S are sent to the verifier.

▶ The verifier applies Bob’s public verification key (N, e) to S to check: Se = M mod N.

▶ The verifier accepts the message-signature to be from Bob if the equation holds, and rejects it otherwise.

e.g.,

Bob’s RSA signature scheme:

▶ Private key d = 23 to sign.

▶ Public key (N, e) = (143, 47).

To sign message m = 67:

▶ Bob computes s = 67d = 6723 = 111 mod 143.

▶ Bob sends (m, s) = (67, 111) to the verifier.

To verify message-signature pair (67, 111):

▶ Check that 111e = 11147 = 67 mod 143.

##### Security

Forgery without factoring: due to underlying multiplicative property of RSA.

y1 = x1d mod n w/ y2 = x2d mod n → y1y2 = (x1x2)d mod n

So if s1 is a signature on m1 and s2 is a signature on m2, then s1s2 is a signature on m1m2.

(Existential forgery using a known-message attack.)

Prevention: using appropriate redundancy function, or use hash function in conjunction with signature algorithm.

###### Appendix with hash function

RSA signature scheme:

▶ Private key d to sign.

▶ Public key (N, e) to verify.

To sign the document m:

▶ Signer uses hash function h to hash the document m so that 0 < h(m) < N.

▶ Signer computes the signature s = h(m)d mod N.

▶ Signer sends m and s to the verifier.

To verify whether s is a valid signature on m:

▶ Verifier uses (N, e) as verification key to check se = h(m) mod N.

▶ If the equation holds the verifier accepts that Bob is the originator of the message and the message has not been modified. Otherwise the claim is rejected.

###### Long messages

To apply an RSA signature to a long message m we must break m into blocks m1, m2, . . . so that mi < N, N being the RSA modulus. Worse, we must add a serial number and additional redundancy to each piece mi, reducing the amount of actual message in each decrypted block.

▶ This is time and resource consuming for long messages.

Standard technique: sign hash of message rather than message.

###### Message recovery

We can also use RSA with message recovery:

▶ Bob’s RSA signature scheme: Private key d to sign, public key (N, e) to verify.

▶ To sign message M: Bob computes the signature S = Md mod N.

▶ Bob sends S to the verifier.

To verify whether S is a valid signature and to recover M:

▶ The verifier uses Bob’s public verification key (N, e) to compute Se = M mod N.

▶ If M contains the correct redundancies then the verifier accepts that M is the original message, and Bob is the originator of the message and the message has not been modified. Otherwise the claim is rejected.

#### Digital signatures in practice

In practice, digital signatures are almost always of the type with appendix.

A close-up of a sign

Description automatically generated▶ The message must be sent with the signature, and the verification process takes as input both the message and the signature.

▶ Type with appendix also known as ”digital signatures without message recovery”.

▶ Requires the message M to be input to the verification process.

To sign:

1. The signer hashes the message M being signed.

2. The signer signs the hash. The signed hash is the digital signature on M.

3. The signer sends to the verifier the message M itself and the digital signature (signed hash).

##### Verification

The receiver receives a message M and a digital signature on M.

To verify:

1. The verifier hashes the message M.

2. The verifier verifies the digital signature on the hash.

3. If the verification passes, the verifier accepts that the signature is valid on the message.

##### Signatures with hash functions

Hash functions must not weaken the signature schemes:

Existential forgery using a known-message attack:

▶ Oscar has valid signed message from Bob (m, SB(h(m)) ).

▶ Oscar computes h(m) and finds m′ != m such that h(m′) = h(m). Then (m′, SB(h(m)) ) would be valid.

▶ Hence h should be second preimage resistant.

Existential forgery using a chosen-message attack:

▶ Oscar finds distinct m, m′ with h(m) = h(m′).

▶ Oscar persuades Bob to sign h(m) to get (m, SB (h(m))). Then (m′, SB (h(m))) would also be valid.

▶ Hence h should be collision resistant.

If the signature scheme (without hash function) allows existential forgery with key-only attack, then Oscar can forge a signature on some message digest z and find a message m such that h(m) = z.

Hence h should be preimage resistant.

Other signature schemes

Another important scheme is the ElGamal signature scheme (based on discrete logs, in multiplicative group of integers mod p, where p is a prime)

▶ Randomised scheme, modified and adopted as NIST standard Digital Signature Algorithm in the 1990s.

Post-quantum digital signature schemes: based on codes and elliptic curves.

#### If confidentiality is required?

Digital signature schemes with appendix does not provide message confidentiality.

▶ If confidentiality is required we could use a combination of signature scheme and encryption.

Generally recommend signing before encrypting, with sender and receiver identity included.

▶ Or use signcryption schemes.

## Public key management

As with public key encryption schemes, we have the problem of distributing the public key so that the recipient knows it is valid.

Certification authorities

One solution: everyone registers with a Certification Authority (CA) (a trusted third party).

▶ Every user submits their public key to their CA who makes sure that the users’ credentials are verified.

▶ The CA signs a public key certificate which binds a public key to relevant data.

▶ Everyone gets a copy of the CA’s public signature verification key by a trusted means.

A public key certificate binds a public key to relevant data:

▶ User name, User public key (for encryption or verification), Expiry date, . . .

CA concatenates all the relevant data and generates a signature on the data string.

▶ s = SCA(user name, user public key, expiry date, . . .).

▶ Anyone with the CA’s public key can verify the user’s public key certificate, thus getting a trusted copy of a user’s public key.

If more than one CA exists, cross-certificates are needed (one CA’s public key is signed by another CA).

## Signature-based entity authentication

An application of digital signatures

Challenge-response scheme with signatures.

▶ Use nonces (random numbers/time stamps/sequence numbers) for freshness.

Bob needs to have an authenticated version of Alice’s signature verification key in order for Bob to accept Alice’s signature and authenticate Alice.

▶ Similarly for Alice, if Alice wants to authenticate Bob.

e.g., mutual authentication:

Suppose Alice and Bob have each other’s authenticated signature verification keys.

Write SX(m) to mean X’s signature on m.

1. Bob → Alice: RB

2. Alice → Bob: RA || SA( RA || RB || iB )

3. Bob → Alice: SB (RB||RA||iA)

RA, RB are random numbers generated by Alice and Bob respectively.

▶ The random numbers provide freshness guarantees.

When Bob receives the message from Alice in Step 2, Bob must verify Alice’s signature. Similarly Alice must verify Bob’s signature in Step 3.

▶ The signatures provide (mutual) data origin authentication.

# Key establishment

In symmetric cryptosystems, parties must agree to a key before communication.

▶ Compromise of key compromises system.

Properties of keys:

▶ Large enough for security.

▶ Easy enough to handle.

A major issue in symmetric cryptography.

Public Key Cryptosystems (PKC): no need of secure channel for key exchange.

▶ Issue 1: authenticity of Bob’s public key in the presence of active adversary.

▶ Issue 2: Most PKC are slower than symmetric key systems

Users (or pairs of users) may have long-term keys:

▶ Precomputed and stored securely;

▶ Or computed from securely stored secret information (key pre-distribution).

Often used to transmit session keys;

▶ Short-term key for a particular session only.

▶ Can be updated frequently to limit amount of ciphertext (encrypted with one key) available to cryptanalyst, and to limit exposure if session key is compromised.

## Security protocols

A protocol is a set of rules for exchanging messages between two or more principals/participants over a network.

Rules cover:

▶ Message formats.

▶ How to handle the messages on receipt.

▶ How messages are interpreted.

Here we will see how basic cryptographic primitives (encryption, MACs, signatures) can be used to provide security services (authenticated key establishment) over insecure networks.

▶ Build on ISO 7498 OSI generic security architecture.

In a secure protocol:

When acting honestly, principals/participants achieve the stated aim of the protocol.

▶ Example 1: A authenticates B.

▶ Example 2: A and B sets up a fresh session key (with certain assurances).

Neither a passive eavesdropper nor an active adversary can defeat this objective.

▶ In Example 1: Oscar cannot successfully impersonate B to A.

▶ In Example 2: Oscar cannot persuade A and B to reuse an old session key.

Legitimate participants: conventionally called Alice, Bob, Carol, etc.

▶ In more complex protocols there may be a Trusted Third Party (TTP) who is trusted by the legitimate participants. Depending on the application they may be called Trusted Third Party, Trusted Authority, Trusted Server, Certification Authority, etc.

▶ There may be varying levels of trust, for example, trusted to relay messages correctly, trusted to generate keys, trusted to verify identities, etc.

There are two kinds of adversaries:

▶ Eve, a passive adversary, an eavesdropper. Eve can only read sent messages.

▶ Mallory/Oscar, an active adversary, who can:

▶ view, alter, delete, replay message.

▶ inject messages into the network.

▶ initiate protocol runs.

▶ impersonate a principal in a protocol run.

We assume that the underlying cryptographic primitives are secure:

▶ The (pseudo)random number generation is secure:

▶ Mallory cannot guess a random number chosen by another principal if it is selected from a sufficiently large space.

▶ The hash function is secure:

▶ Mallory cannot easily find preimages or collisions.

▶ The encryption algorithm is secure. For example,

▶ In a symmetric key cryptosystem, Mallory cannot deduce the key from observing plaintext-ciphertext pairs.

▶ In a public key cryptosystem, Mallory cannot deduce the private key from a public key.

▶ The signature scheme is secure:

▶ Mallory cannot deduce the signing key from the public verification key and from observing message-signature pairs.

From the Handbook of Applied Cryptography:

▶ Key establishment: process by which a shared secret key becomes available to two or more parties for subsequent cryptographic use.

▶ Key distribution: one party chooses a key and transmits it securely to others.

▶ Key agreement: the secret key is derived by all parties as a function of inputs by all parties.

▶ Key management: set of processes and mechanisms which support key establishment and the maintenance of ongoing keying relationships between parties.

▶ Eg. key generation, distribution, storage, update, destruction, etc.

## Authenticated key establishment

Entity authentication can only be achieved for an instant in time.

▶ Typically this is established at the start of a connection/session.

If we want security (confidentiality/integrity) for a whole session we need to establish a session key.

▶ A session key can be agreed as part of an authentication protocol.

▶ The session key can be bound to that protocol run.

▶ This can be done in an authenticated key establishment protocol.

TTP can be online or offline. Can be certification authority, vouching for authenticity of public keys or key generator or key escrow agent etc. Third parties have varying levels of trust.

Key establishment protocols

Many different scenarios and methods and models:

▶ Key pre-distribution: TTP distributes keying information ahead of time securely. Pairs of users can derive secret keys later on.

▶ Key transport/distribution: One party creates and transfers the key to other parties.

▶ Key agreement: Users agree on session key using interactive protocols, maybe based on symmetric-key or public-key schemes. Usually do not require on-line TTP.

#### Security goals

Implicit key authentication:

▶ No one other than specified party may gain access to a key

Key confirmation:

▶ Assurance that second party (possibly unspecified) has actual possession of a key

Entity authentication:

▶ Assurance of identity and liveness of communicating party

Explicit key authentication:

▶ both implicit key authentication and key confirmation

Other assurances:

▶ Key freshness: guarantee that new key is used.

▶ Key control: neither party can control/predict key value.

Other considerations:

▶ Efficiency: number of passes and bandwidth, complexity of computations.

▶ TTP requirement: on-line, off-line, or none; degree of trust.

Security under different attack models:

▶ Security if a session key is known?

▶ Security if long-term key is known?

Perfect forward secrecy: compromise of long-term keys does not affect security of short-term keys made before the compromise.

#### Session keys example

Two parties, Alice and Bob, share a secret long-term key k.

Session key establishment:

▶ Alice sends Bob k′ in the clear.

▶ Session key is k ⊕ k′.

Weaknesses?

##### Key hierarchies

Two parties, Alice and Bob, share a master key kM.

Session key establishment:

▶ Alice sends Bob session key as ekM(kS)

▶ Bob decrypts and obtain session key kS

▶ Many level hierarchies are possible.

▶ Weaknesses?

#### Example using TTP

TTP is an agency trusted by all parties.

TTP can generate and convey keys. Each user has secret key agreed with TTP.

▶ Alice shares secret key kA with TTP.

▶ Bob shares secret key kB with TTP.

When Alice and Bob wish to communicate:

▶ TTP generates session key kS .

▶ TTP sends Alice ekA(kS) and Bob ekB(kS)

▶ Weaknesses?

#### Key pre-distribution

TTP distribute keying information securely ahead of time.

▶ Preload keys/keying material on to devices in controlled environment before deployment.

Pairs of users later determine key from keying information.

Evaluation criteria:

▶ How much information to be transmitted securely,

▶ How much information to be stored securely,

▶ Others: how much information to be published or broadcast, how much computation to be performed by TTP and users.

A trivial example: for each pair of users U, V , TTP chooses random key KUV and transmits it securely to U and V. Examples: Preloaded keys in SIMs on mobile phones, or Set top boxes for digital TV services.

Issues:

▶ Keeping track of device ownerships.

▶ Post deployment key management.

#### Key distribution

One party (could be TTP) chooses a session key and securely transfers it to the others.

Many different scenarios possible:

▶ Using symmetric key cryptosystems only, or PKC only, or a hybrid.

▶ Different levels of involvement of TTP and trust in TTP.

▶ Different levels of input to the session key.

Simple example using symmetric key cryptosystems and time-stamps:

▶ A and B share a long-term key K .

▶ A → B: eK( t || iB || Ks ) t is a time stamp, iB is an identifier for B, and Ks is a session key.

▶ A is authenticated to B and they now share a secret session key Ks.

▶ Implicit key authentication: No one other than B (and A) may gain access to Ks.

Simple example using public key cryptosystems and MACs:

A checks the authenticity of B’s public key PKB. A → B: ePKB (Ks).

Subsequent messages are encrypted or authenticated using Ks (or keys derived from Ks )

Assurances? B → A : data, MACKs(data)

▶ A is not authenticated to B.

▶ B is authenticated to A if subsequent messages are correctly encrypted/authenticated using Ks.

▶ Explicit key authentication if subsequent messages are correct: only B (and A) could have Ks and B does actually have Ks.

Example using TTP:

The “wide-mouthed frog protocol”:

▶ Alice shares a key KAT with TTP.

▶ Bob shares a key KBT with TTP.

▶ If Alice and Bob wish to communicate, then Alice chooses session key KAB and TTP transfers it to Bob securely.

1. Alice → TTP eKAT ( tA || IDB || KAB )

2. TTP → Bob eKBT ( tT || IDA || KAB )

#### Kerberos

Kerberos is a TTP-aided authentication protocol.

▶ Can achieve mutual authentication and key establishment.

▶ The name also refers to software implementing that protocol, currently Kerberos V5 Release 1.2.

▶ Also the name of a project at MIT which devised the protocols (properly called Project Athena).

▶ Standardised in RFC 1510 – Kerberos V5 (1992).

▶ Version of Kerberos incorporated in Windows and used in many versions of Unix

Authentication of Client (C) to Server (S) done as a two-stage process.

Authentication Server (AS):

▶ Mutual authentication with Client at login based on a shared long-term secret.

▶ Gives client *ticket granting ticket* and a short-term key for use between Ticket Granting Server and Client.

Ticket Granting Server (TGS):

▶ Performs mutual authentication with Client based on the short-term key and ticket granting ticket.

▶ The TGS then issues tickets giving Client access to further Servers that demand authentication.

Two TTPs: Authentication Server (AS) and Ticket Granting Server (TGS)

▶ A user only needs to load their long-term secret key into the client host for the minimum time.

▶ Once the short-term key is established this long-term secret key can be erased from the client host.

A diagram of a server

Description automatically generatedAll further client interactions are with TGS and servers.

This minimises the risk of exposure of the long-term secret key.

Messages 3 & 4 can be repeated without

repeating messages 1 and 2.

Messages 5 & 6 can be repeated without

repeating messages 3 and 4.

C and AS share long-term key KAS,C derived from C’s password.

Optionally, S can send C another message to authenticate itself to C.

A close up of a logo

Description automatically generatedA close-up of a document

Description automatically generated

A close-up of a ticket

Description automatically generated

A close up of a text

Description automatically generated

Kerberos uses symmetric encryption and Manipulation Detection Codes (MDC).

▶ The MDC is computed on the data to be encrypted, and then the concatenation of the MDC with the data is encrypted.

▶ Specifically, Kerberos version 5 (as originally in RFC 1510) uses DES and MD4 or MD5.

▶ Release 1.2 of Kerberos Version 5 implements triple DES (3DES).

##### Issues

Revocation: ticket granting tickets valid until they expire, typically 10 hours.

Within realms (domains), long-term keys need to be established between AS and TGS, TGS and Servers and AS and clients.

Synchronous clocks are needed, and must be protected against attacks.

Cache of recent messages to protect against replay

AS and TGS must be trusted by clients not to eavesdrop.

▶ Can be extended to include keying material to establish additional secret not chosen by AS or TGS.

Client-AS long-term key often still based on password entry – vulnerable to guessing.

Short-term keys and ticket granting tickets located on largely unprotected client hosts.

Denial of service possible? E.g. on the clock service or on the TGS.

##### Windows network authentication

Microsoft has adopted and extended Kerberos to provide network authentication in Windows.

First extension: support for public key encryption to protect client/AS messages (rather than password-based long-term key).

Second extension: use Kerberos (normally empty) data authorisation field to transmit access privileges.

Message formats proprietary to Microsoft.

Non-standard extension to Kerberos makes it hard to interoperate Microsoft & non-Microsoft implementations.

## Key agreement protocols

Key agreement: secret key derived by all parties as a function of inputs by all parties.

Diffie-Hellman key exchange:

Allow two parties who have not met in advance or shared keying material to establish shared secret by public exchange of message. The first practical solution.

A white background with black text

Description automatically generatedDiffie-Hellman key exchange

A table with numbers and symbols

Description automatically generated

#### The Diffie-Hellman problem

Outsider knows p and α, yA = αXA mod p, yB = αXB mod p. Determine k = αXA x XB mod p from this information.

Example: Prime p = 59, primitive element α = 2

▶ Given 2xA = 47 mod 59, 2xB = 33 mod 59, find 2xA x xB mod 59

▶ Diffie-Hellman problem believed to be hard.

##### The discrete log problem

Prime p and primitive element α.

▶ Given β ∈ Zp , find integer a (0 ≤ a ≤ p − 2) such that β = αa mod p.

▶ Can try to calculate xA from αxA mod p - the discrete log problem.

Solution to discrete logarithm gives solution to Diffie-Hellman problem.

#### A diagram of mathematical equations Description automatically generatedIntruder-in-the-middle

Station-to-station (STS) protocol

Public information: prime p, primitive element α.

▶ User U: signature generation function SU, signature verification algorithm VU certified by TTP.

▶ K = αXA x XB mod p can be calculated by Bob after the first message, and Alice after the second message.

Achieves key agreement, mutual entity authentication, explicit key authentication.

1. Alice → Bob yA = αxA mod p

2. Bob → Alice yB = αxB mod p, eK( SA(yB ||yA) )

3. Alice → Bob eK( SB(yA||yB) )

# Computer Security

Computer security is the protection of the items that a person values, called the assets of a computer or networked systems.

Assets: hardware, software, data, reputation, etc.

Value is subjective; based on the user’s perspective. Computer security deals with the prevention and detection of unauthorised access by users of a system.

Authentication: This verifies user identities.

Access control (authorisation): Limits access to programs and resources to those authorised to have access (the main focus of this part of the course).

Memory protection: Segmented virtual memory model prevents a process reading or overwriting memory used by other processes.

## Access Control

A generic term for the process that controls interactions between users and resources.

An access control system implements a security policy determined by

▶ organisational requirements;

▶ statutory requirements, for example, covering Personally Identifiable Information (PII), including medical records.

A white background with black text

Description automatically generatedA white background with black text

Description automatically generated

A close-up of a white background

Description automatically generated

Authorization

A diagram of a system

Description automatically generatedA close-up of a text

Description automatically generated

A close-up of a computer

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white paper with black text

Description automatically generated

A close up of a text

Description automatically generated

A white background with black text

Description automatically generated

A close-up of black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generatedAccess Control Models

A close-up of a computer code

Description automatically generated

#### Access control matrices

A close-up of black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

#### A white background with black text Description automatically generatedAccess control lists

A white background with black text

Description automatically generated

#### A close up of a text Description automatically generatedCapability lists



A close-up of a white background

Description automatically generated

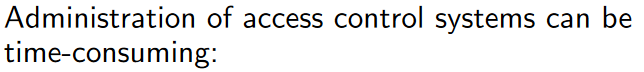
#### A diagram of lines with arrows and letters Description automatically generated with medium confidenceA white background with black text Description automatically generatedInformation flow models

A white background with black text

Description automatically generated

A white text with black text

Description automatically generated

RBAC

A white background with black text

Description automatically generatedA close up of words

Description automatically generated

A close-up of a sign

Description automatically generated

A close up of a white background

Description automatically generated

A diagram of a role model

Description automatically generated

A diagram of a system

Description automatically generated

A close up of black text

Description automatically generated

A white background with black text

Description automatically generated

A diagram of a company

Description automatically generated

#### Hierarchical RBAC

A close up of black text

Description automatically generated

A white background with black text

Description automatically generatedA white background with black text

Description automatically generated

A white paper with black text

Description automatically generated

#### A white background with black text Description automatically generatedA white background with black text Description automatically generatedConstrained RBAC

A close-up of a white background

Description automatically generated

A diagram of a system

Description automatically generatedA white text with black text

Description automatically generated

A white paper with black text

Description automatically generated

#### Administration

A close-up of black text

Description automatically generatedA white background with black text

Description automatically generated

A white background with black text

Description automatically generated

# Network Security

A white background with black text

Description automatically generated

A white paper with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

A close up of black text

Description automatically generatedA white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generatedA white background with black text

Description automatically generated

A white background with black text

Description automatically generated

## Security Services

A close-up of a white background

Description automatically generated

A white background with black text

Description automatically generated

A close-up of a white background

Description automatically generatedAuthentication

A close-up of black text

Description automatically generatedAccess control

A close-up of a white background

Description automatically generatedData confidentiality

A white background with black text

Description automatically generatedData Integrity

A white background with black text

Description automatically generated

Non-repudiation

## A white background with black text Description automatically generatedSecurity mechanisms

Encryption

A close-up of black text

Description automatically generated

A close-up of a white background

Description automatically generatedDigital signature

A white background with black text

Description automatically generatedAccess control

A white background with black text

Description automatically generatedData integrity

A close-up of a white background

Description automatically generated

Authentication exchange

A close-up of black text

Description automatically generatedTraffic padding

A white background with black text

Description automatically generatedRouting control

A close-up of black text

Description automatically generatedNotarisation

## A diagram of a network Description automatically generatedLayered Models

A white text with black text

Description automatically generated

A white background with black text

Description automatically generatedIPsec

A close-up of a computer screen

Description automatically generated

A white paper with black text

Description automatically generated

A white paper with black text

Description automatically generated

A diagram of a computer

Description automatically generatedA white background with black text

Description automatically generatedTLS

A white paper with black text

Description automatically generated

## A white background with black text Description automatically generatedIntrusion prevention and detection

A diagram of a network

Description automatically generated

Firewalls

A close up of a text

Description automatically generated

A close-up of a white background

Description automatically generated

A screenshot of a computer error

Description automatically generated

#### Types

##### Packet filtering

A white background with black text

Description automatically generated

###### A white text with black text Description automatically generatedA close up of a text Description automatically generatedStateless packet filtering

A close up of a text

Description automatically generated

###### Stateful inspection

A white text on a white background

Description automatically generated

##### Proxy servers

A white paper with black text

Description automatically generated

###### A white text on a black background Description automatically generatedApplication gateways

###### Circuit-level gateways

A white background with black text

Description automatically generated

A white background with black text

Description automatically generatedIntrusion detection

A white background with black text

Description automatically generated

A close up of a text

Description automatically generated

A white background with black text

Description automatically generated

A white paper with black text

Description automatically generated

## A white background with black text Description automatically generatedA white background with black text Description automatically generatedSoftware Security

A white text with black text

Description automatically generatedA white paper with black text

Description automatically generated

A white background with black text

Description automatically generatedA white text with black text

Description automatically generated

A white background with black text

Description automatically generated

A white paper with black text

Description automatically generated

A white background with black text

Description automatically generatedA white text with black text

Description automatically generatedA white text on a white background

Description automatically generatedA white text on a white background

Description automatically generatedMalware