FIT2093 Notes

Introduction to Cyber Security (Monash University)

Week 1 – Introduction to Cyber Security

What do we mean by cyber security?

- Body of technology, processes and practices designed to protect networks, computers, programs and data from attack, damage or unauthorised access.
- In a computing context, security includes both cyber security and physical security.

Defining security

- Cyber security in a nutshell is how to design systems that work even in the presence of malicious entities
- Security of a system, application or protocol is always relative to
  - A set of desired properties
  - An adversary with specific capabilities

Types of threats

- Unauthorised Information Disclosure – Unauthorised reading – Private data revealed to attacker
- Deception/alteration of information – unauthorised writing – false attacker data accepted
- Disruption of normal operation – incorrect/failed computation – also called denial of services

Types of vulnerabilities

- Design/Logic vulnerabilities:
  - Flaws in the logic/protocol of the software
- Implementation vulnerabilities – software bugs

Types of attacks aiming at information disclosure

- Exposure
- Interception/eavesdropping
- Inference
- Intrusion

Types of attacks aiming at deception/alternation

- Fabrication/Masquerade
- Modification
- Repudiation

Types of attacks aiming at disruption

- Incapacitation/ interruption
- Corruption
- Obstruction

Network security attacks

- Classified at passive or active
- Passive attacks are eavesdropping – hard to detect
- Active attacks modify/fake data – had to prevent

Passive attacks

- Release of message contents – opponent learns contents of sensitive transmissions
- Traffic analysis – can occur even when contents of messages are masked (e.g encryption)
- The goal is to obtain information of breach confidentiality of property such as a password

Active attacks

- Involve modification of data stream or creation of false data
  - Masqurade – when one entity pretends to be another r
  - Replay – passive capture of data and subsequent retransmission
  - Modification of messages – a legitimate message is altered, delayed or reordered

    - Denial of service prevents or inhibits normal use or management of communication facilities or the disruption of an entire network

Confidentiality

- Preserving authorized restrictions on information access and disclosure, including means for protection personal privacy and proprietary information
- Sensitive information can only be accessed by authorised parties
- Access can be in the form of reading, copying or distributing
- A loss of confidentiality is the unauthorized disclosure of information

Integrity

- Protecting information from unauthorised modification
- To ensure the authenticity of the data.
    - Data should be genuine, not only appear to be
    - Authenticity refers to the truthfulness of origins
- Ensuring information non-repudiation
    - Prevents either sends or receiver from denying a transmitted message

Availability

- The information should be accessible and useable without delay upon demand by an authorised entity

Assets of a computer system

What are we protecting?

- Hardware
    - Computer
    - Network
    - Infrastructure
- Software
- Data/Information
- Communication facilities and networks

Scope of computer security

1. Access to the data must be controlled (protection)
2. Access to the computer facility must be controlled (user authentication)
3. Data must be security transmitted through networks (network security)
4. Sensitive files must be secure (File security)

How is the protection achieved??

- Prevention – avoid the breach in security
- Detection – investigate a security breach
- Recover

Security protection

- Physical security protection – protecting IT infrastructure from physical damage and access by unauthorised party
- Logical Security Protection (Also known as information security) – protection of the information to preserve confidentiality, integrity and availability of information

Security mechanism

- Designed to detect, prevent or recover from an attack

Levels of impact

- Low: Little adverse effect of organisational operations, assets or individuals

- Moderate: serious adverse effect
- High: severe of catastrophic adverse effect

Managing IT Security in Organisation

- Identity management
- Vulnerability management – firewalls
- Threat management – intrusion detection
- Trust management

Principles of Security

- Principle of easiest penetration – an intruder will use any means of penetration
- Principles of timeliness – items only need to be protected until they lose value
- Principles of effectiveness – controls must work, and they should be efficient, easy to use and appropriate

Security Functional Requirements

- Technical measures
    - Access control, identification and authentication, system and communication protection, system and information integrity
- Management controls and procedures
    - Training, physical protection, risk assessment, maintenance
- Overlap
    - Incident response, media protection, configuration management

X.800 Security Architecture

- Systematic way of defining requirement for security and characterising approaches to satisfy them
- Defines
    - Security attacks – compromise security
    - Security mechanism 0 acts to detect prevent or recover from an attack
    - Security service – counter security attacks and enhances the security of the system.
- Security Services
    - Authentication – assurance that communicating entity is the one claimed
    - Access control
    - Data confidentiality – protection from unauthorised disclosure
    - Data integrity – recieved as sent
    - Non- repudiation – protection against denial by one of the parties in communication
    - Availability – accessible. /useable
- Security Mechanism
    - Specific mechanisms
        - Encipherment
        - Digital signatures
        - Data integrity
        - Routing control
    - Pervasive security mechanisms
        - Security labels
        - Event detection
        - Audit trails
        - Security recovery
    - Specific security mechanisms are protocol layer specific, whilst the pervasive security mechanisms are not

Computer security strategy

- Policy
    - What is the security scheme supposed to do
    - Codify in policy and procedures
- Implementation

    - How does it do it
    - Prevention, detection, response, recovery
- Correctness
    - Does it really work?
    - Assurance, evaluation

Week 2 – Cryptography – Concepts/Principles

Original message – plain text. Scrambled message – cipher text.

The aim of encryption in cryptography is:

- To generate the cipher text from place text and vice versa
- That is reversible if you know the secret scrambling information known as the key

Cryptography – code design   Cryptanalysis – code breaking

Cryptologist – both design and breaking

Classic Cryptography

Uses symmetric Key encryption, where the same key is used for encryption and decryption

Basic principles to hide a message

Substitution

- Systematically substitute letters or a group of letters with other letters or groups of letters

Transposition

- Rearrange the position of letters in the message according to some rules

Classical Cipher – Caesar Cipher

- Substitution cipher named after Julius Caesar
- Each letter is translated to a letter a fixed number of position after it in the alphabet table
- The fixed number of positions in the key.

Let x to be the position of the letter to be replaced and n to be the number of positions to be shifted in order to find the replacement letter. N is the key

En(x) = (x + n) mod 26

Dn(x) = (x-n) mod 26

Breaking the Caesar Cipher

By brute force exhaustive search for key. Lets say we want to encrypt “Alice”. There are only 26 possibilities which is easy to brute force, and only one of them will usually make sense

Monoalphabetic Substitution cipher

- Build a mapping table that maps a character with the replacement character
- How many mapping tables are possible for 26 upper case characters?
- 26! = 4 x 10^26
- Brute force search is infeasible

Statistical frequency analysis

- Observation: each plaintext letter always encrypts to the same character
- Letter frequency in the English alphabet (mainly vowels) is useful for breaking such encryption.
- By replacing the most common letter with E in the encrypted message we can begin to break down the coded message.
- This can be made more complicated by encrypting it twice with two different keys

Rail fence cipher – classical transposition cipher

- The letters are placed in multiple number of lines. The letters start from line 1, position 1, line 2 position 2, line 3 position 4, line 2 position 4 ect. Creates a zig zag pattern
- Key is rail fence, depth three

Transposition Cipher – more general

- Decide on a block size L (say L = 10 letters)
- Fix a permutation of length L the Key (5,3,1,4,6,8,0,9,7)
- Shift each letter by this amount

Why combine these two operations?

- Each methods contributes its own strength
- The product is more complex than one individually
- The number of possible product keys is like to be the product of number of keys of individual methods

What properties should scrambled messages have?

Confusion

- Seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possibl
- Achieved by complex substitution
- Difficult to deduce key

Diffusion

- The statistical structure of plaintext is dissipated in the long range statistics of the ciphertext
- Scrambling should spread the information from the plain text so that each plain text digit affects many cipher text digits
- This can be achieved through complex transposition

Avalanche Effect

- Is a property of any encryption algorithm such that a small change in either the plaintext or the key produces significant change in the cipher text
- A change in one bit of plaintext or one bit of key should change many bits in the ciphertext

Unconditional security/Computational security

- Ideally we would like to have unconditional security
- No matter how much computer power is available, the cipher cannot be broken. This is possible but comes at a cost
- In practice, it is usually sufficient to have computational security
- Given the attackers limited computing resources, the cipher cannot be broken (time, cost)

Security in practice: computational security via Transposition-substitution Product Cipher

- Need efficient transposition method
- Need to avoid large substitution tables, but still need a methods to substitute characters or bit

Representation of information to encrypt

- Sequence of characters
- Each keyboard character is represented by a 7bit/8bit ASCII code inside a computer
- Then a sequence of n characters is encoded into 0s and 1s
- n characters has 8 x n bits

BLOCK

- A long message can be broken down into equal sized blocks
- Exception – last block which is padded with blank characters

Scrambling a long message

Scramble each of the block of the message with block cipher

- Block size is constant
- Pad last block with special characters

Precise way of doing this depends on the mode of operation of block cipher

- Details of the mode are important for security
- Some natural modes are insecure and should not be used

Block Ciphers – building blocks for symmetric key encryption

Requirements

Design an encryption algorithm E(K,M) that:

- Given a secret key K of length n_K bits
- A message block M of length n bits
- Algorithm E applies efficient substitution and transposition operations to compute a ciphertext block C = E(K,M) of length N bits
- Use a suitable repeated combinate of confusion (substitution) and diffusion transposition) operations
- To scramble a block of plaintext bits into cipher text block of same length

Properties

- Decryption correctness: there is an efficient decryption algorithm D that correctly decrypts cipher texts produced by E
- Pseudorandom Function Computational security: E should have Confusion and Diffusion properties. "Cipher texts look random".

Bit-wise exclusion- or (XOR)

0 XOR 0 = 0

1 XOR 1 = 0

1 XOR 0 = 1

0 XOR 1 = 1

Substitution

- If you take two sets of bits, and take the first set as the data block and the second and the substitution table, the SUBSITUTION OPERATOR is XOR
- The second sequence of bits needs to be pre-defined (KEY)

Week 3 – Symmetric- Key encryption

A symmetric key cipher is composed of two algorithms. Enc and Dec. The same key K is used for both. K has to be distributed beforehand.

Block cipher

Design an encryption algorithm E(K,M) that:

- Given secret key K of length N_K bits
- A message block M of length n bits

Properties

- Decryption correctness
    - There is an efficient decryption algorithm D that correctly decrypts the cipher text
- Pseudorandom Function Security
    - E should have the confusion and diffusion properties
    - Cipher text should look random

Feistel Cipher

Many modern block encryption algorithms have the structure of a Feistel Cipher. This is an example of a product cipher with repeated substitution and transposition

Fiestel cipher depends on:

- Block size: larger block means more security
- Key size: larger key size means greater security
- Number of rounds: multiple rounds offer increasing security, usually 16
- Subkey generation algorithm: greater complexity will lead to difficult cryptanalysis
- Round function: greater complexity will lead to difficult cryptanalysis

Data Encryption Standard (DES)

- The most widely used symmetric encryption scheme
- DES is a block cipher
    - Processed in 64Bits
    - 56bit key
    - 8 parity bits are stripped off key
    - 16subkeys are generated for the 16 rounds

DES Subkey Generations Round 1

- Drops the 8 parity bits from the 64Bits
- Permutates the bits then divides it into 2, 28 bits
- Rotates the left bits by a single bit
- Permutates and extracts 48 bits as the subkey

Shifts one bit in round 1, 2, 9 and 16. Every other round is two bits

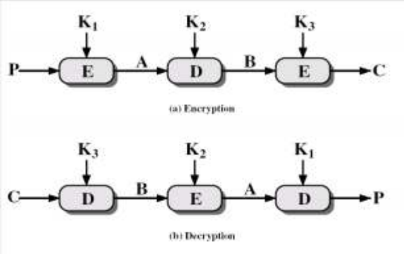
DES Round

- Each of the 16 rounds takes a 64 bit block as input and produces the same length output
- The output from the previous round is the input to the next round, and the final output from round 16 is the ciphertext

Encryption algorithms: Triple DES

- Apply DES algorithm 3x
- Use three keys and three executions of the DES algorithm (encrypt, decrypt, encrypt)

C = $E_{K3}[D_{K2}[E_{K1}[P]]]$



(a) Encryption

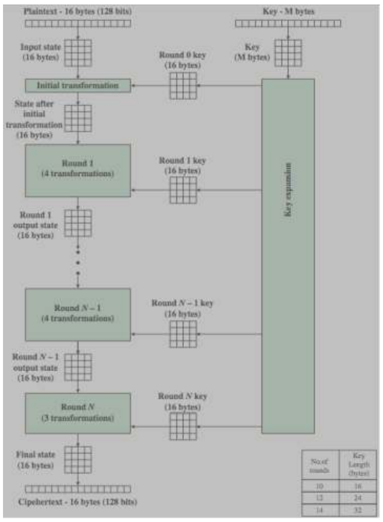
(b) Decryption

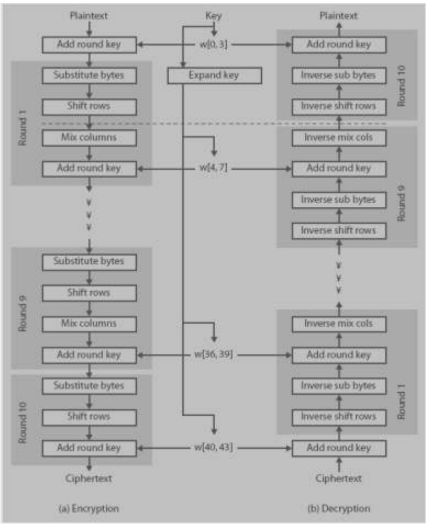
Advance Encryption Standard

- In 1997, US National Institute of Standards and Technology issued call for proposals of algorithms for Advanced Encryption Standard (AES) to replace DES based on:
  - Security, computational efficiency, memory requirements, hardware and software suitability, flexibility
  - Selected in 2000, standardised in 2001

Algorithm uses

- 128 bit block cipher
- Supports three different key lengths: 128, 192, 256
- 10 or 14 rounds
- Offers flexibility
- Efficient implementation in both hardware and software


Plaintext - 16 bytes (128 bits)
Key - M bytes
Input state (16 bytes)
Round 0 key (16 bytes)
Key (M bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round N – 1 (4 transformations)
Round N – 1 key (16 bytes)
Round N – 1 output state (16 bytes)
Round N (3 transformations)
Round N key (16 bytes)
Final state (16 bytes)
Cipehertext - 16 bytes (128 bits)
No. of rounds
Key Length (bytes)
10
16
12
24
14
32



(a) Encryption

(b) Decryption

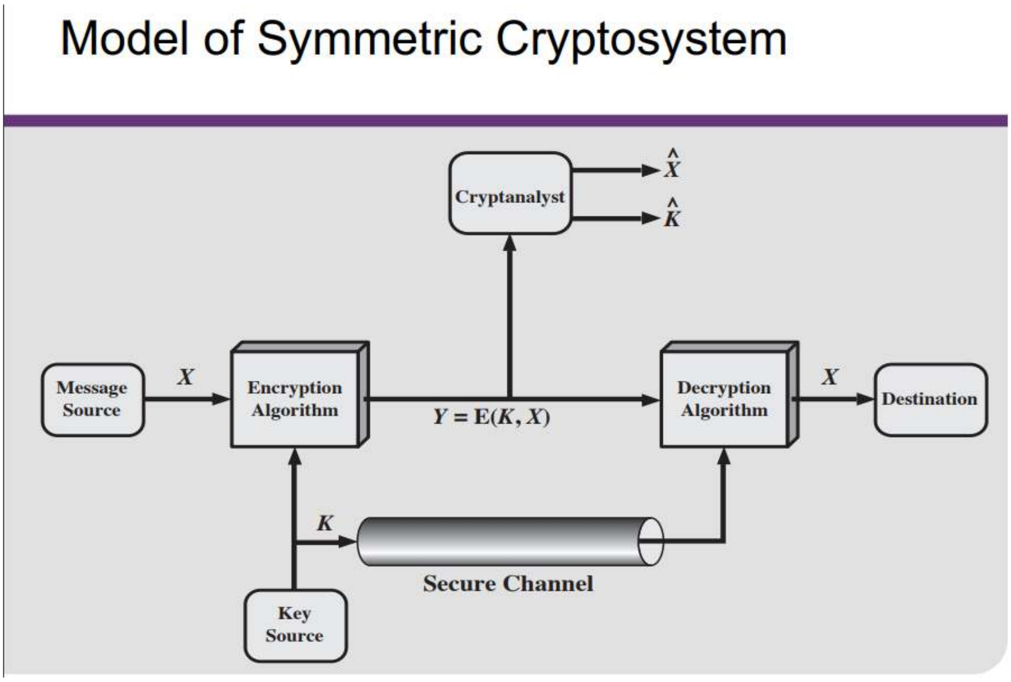
- Data block of 4 columns of 4 bytes
- Key is expanded to array of words
- Has 9/11/13 rounds which state undergoes
  - Byte substitution (Sbox for on everyone byte)
  - Shift columns (permutates bytes between groups/columns)
  - Mix columns (subs using matrix multiply of groups)
  - Add round key (XOR with key materia)

Symmetric-Key Encryption: Attack Models and Confidentiality Goal

The attack model for Symmetric Encryption

Confidentiality is the goal

- Assumption 1: The network is open and can be eavesdropped on
- Assumption 2: Key is agreed upon privately
- Assumption 3: Attacks knows algorithms but not key



Model of Symmetric Cryptosystem

Cryptanalysis

- Attack relies on the nature of the algorithm and some knowledge of the general characteristics of the plaintext
- Attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used

4 Types of Cryptanalysis

Ciphertext only attack

- Attacker has some samples of Cipher text

Known plaintext attacks

- Attacker knows part of the plaintext (Guesses part or is published later)

Chosen plaintext attacks

- Attacker can feed encryption algorithm and obtain the matching ciphertext

Chosen ciphertext attack

- Attacker can feed decryption algorithm with ciphertext and obtain matching plaintext

Brute force attack

- On average, half of all possible keys must be tried to achieve success
- Involves trying every possible key until cipher text is decrypted
- To supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed, and some means of automatically distinguishing the plaintext from garble is also needed.

Modes of operation: how to encrypt arbitrary length messages with a block cipher

- Block ciphers process data in blocks
- For longer messages, these must be broken up and possibly pad the end of it with bits

- There are 5 modes of operation for application of block cipher ECB, CBC, CFB, OFB, CTR

Electronic Codebook Book (ECB) mode

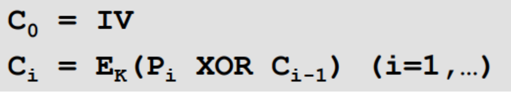
- Message is broken into independent blocks which are encrypted
- Each block is a value which is substituted
- Each block is coded independently of the other

Limitations

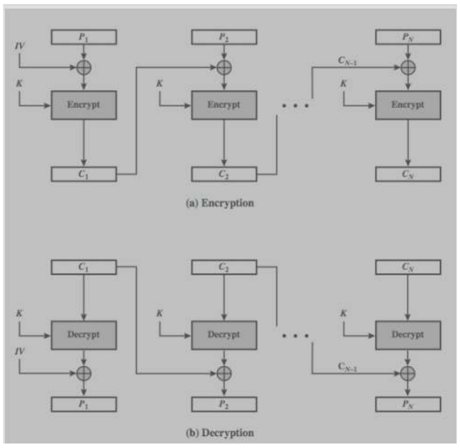
- Message repetitions may show in cipher text is aligned with message blocks
- Weakness is due to the encrypted message blocks being independent, should be avoided

Cipher block chaining (CBC)

- Message is broken into blocks
- Linked together in encryption operation
- Each previous cipher blocks is chained with current plaintext block
- Uses initial vector IV to start processes. This IV is random. In the next iteration instead of XOR iv with plaintext, you XOR, plaintext 2 with previous cipher text
- Used for bulk data and authentication



$$C_0 = IV$$
$$C_i = E_K(P_i \text{ XOR } C_{i-1}) \quad (i=1,\ldots)$$



(a) Encryption

(b) Decryption

Each IV should be random and only be used once (for one message only)

Message Padding

- At end of message to handle a shorter last block
- Options
  - Padd with known non-data value such as nills
  - Or pad with 1 followed by as many 0s as needed
  - Bad with xerows followed by a count of pad size (000006)

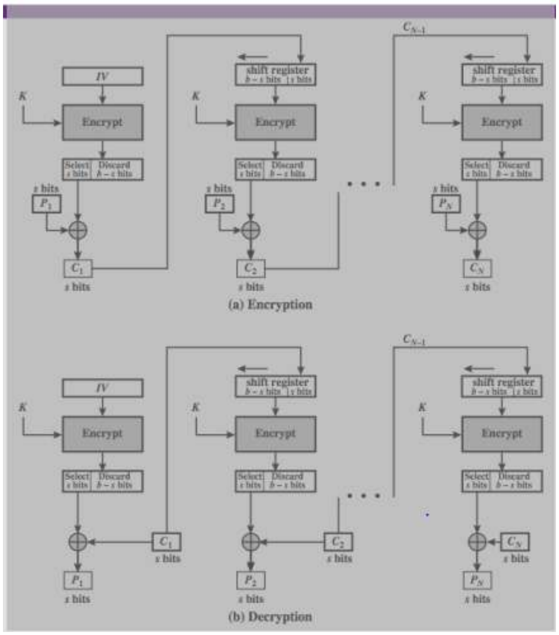
Advantages and Limitations of CBC

- A ciphertext block depends on all the blocks before, cannot run in parallel
- Change to cipher text block affects decryption of that block and also the next block
- Need IV which must be known to send as receiver, independent and random for each encryption
- SECURE if IV is random

Cipher Feedback (CFB)

- Message is treated as a stream of bits
- Added to the output of block cipher

- Result is a feedback into the next stage
- Allows of any number of bits to be fed back



(a) Encryption

(b) Decryption

a. Random number selected as initialization value

b. Key to encrypt the IV

c. The Block cipher E algorithm

d. The shift register to select number of bits based on s value*

e. P stands for plaintext (message being sent)

f. XOR

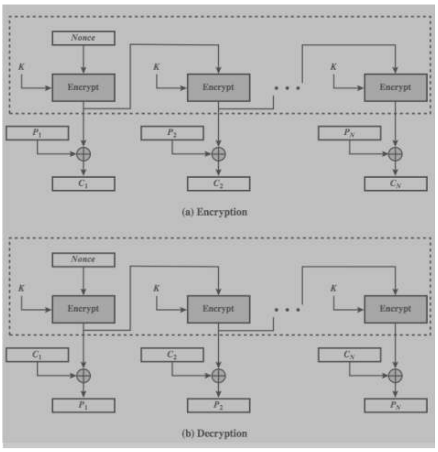
g.C stands for cipher text

Advantages and Limitations of CFB

- Appropriate when data arrives in bits/bytes
- Most common stream mode
- Encryption cannot be parallelizable
- Secure if IV is chosen randomly
- Errors propagate for several blocks after the error

Output feedback (OFB)

- Stream
- A private key block cipher such DES acts as a pseudo-random number generation
- Output of block cipher is added to the message
- Block cipher output is then fed back
- Feedback is independent of the message



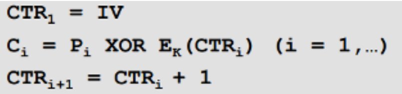
(a) Encryption

(b) Decryption

Advantages and limitations of OFB

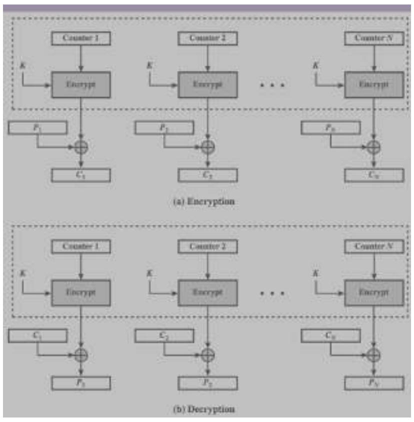
- Bit errors do not propagate
- More vulnerable to message stream modification
- Must never use name sequence of key and IV
- Sender and receiver must remain in sync

Counter (CTR)

- A more recent standard
- Similar to OFB but encypts counter value rather than any feedback value
- Must have a different counter value for every plaintext block (never reused)
- 

  $CTR_1 = IV$

  $C_i = P_i \text{ XOR } E_K(CTR_i) \quad (i = 1, \ldots)$

  $CTR_{i+1} = CTR_i + 1$
- 

  (a) Encryption

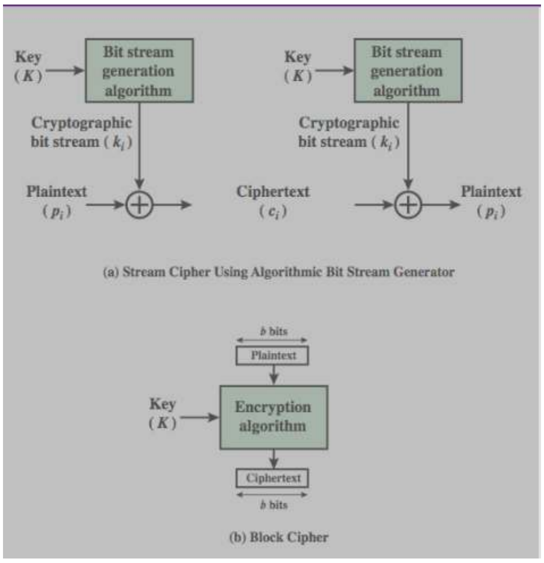
  (b) Decryption

Advantages/Limitations of CTR

- Efficiency
  - Can do parallel encryptions
  - Can pre-process in advance if needed
- Random access to encrypted data blocks
- Same IV/counter values should never be used again

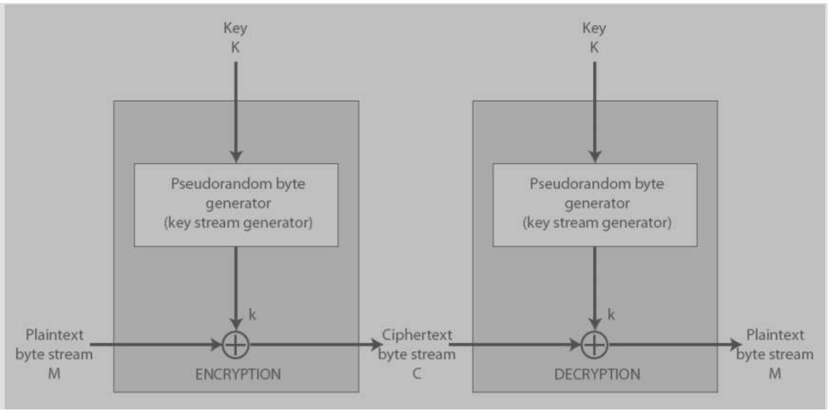
Stream ciphers encrypt bit by bit

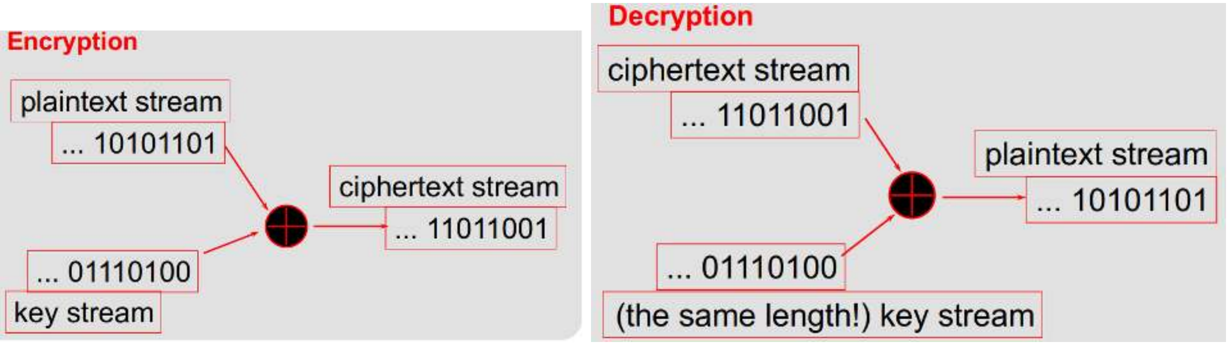
Block vs Stream Ciphers



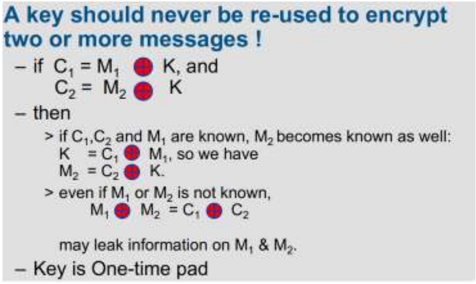
Stream Ciphers

- Process a message one byte or bit at a time
- Have a pseudo random keystream
- Combined XOR with plaintext bit by bit
- Randomness of stream key destroys statistical properties
- Must never reuse stream key





Never reuse a key!



A key should never be re-used to encrypt two or more messages !

- if $C_1 = M_1 \oplus K$, and
  $C_2 = M_2 \oplus K$
- then
  > if C_1,C_2 and M_1 are known, M_2 becomes known as well:
  $K = C_1 \oplus M_1$, so we have
  $M_2 = C_2 \oplus K$.
  > even if M_1 or M_2 is not known,
  $M_1 \oplus M_2 = C_1 \oplus C_2$
  may leak information on M_1 & M_2.
- Key is One-time pad

The problem with a stream cipher is that a key must be as long as the plaintext message. This is solved using a pseudo-random number generator

Pseudo random number generators

Algorithm that expands

- A random by relatively short key into a longer key but preserves the randomness

Stream cipher properties

- Some design considerations are
  - Long period with no repetition s
  - Statistically random
  - But depends on large enough key
- If designed properly can be as secure as block cipher but simpler and faster

Week 4 – Introduction to number theory

Number theory is used for public key cryptography. Public key encryption algorithms all make use of mathematical properties of whole numbers

Factor – an integer x is a factor of n if n can be divided by n without a remainder

Integer factorisation

Factorisation is the process of breaking down a composite number into two smaller, non trivial divisors, which when multiplied equal the integer

Prime factorisation

- Factoring n as a product of prime power
- 15 = 3 x 3
- 24 = 2^3 x 3

Every integer can be factored into a product of prime powers in a unique way.

Integer factorisation problem: Given an integer n, compute the prime factorisation of n

- Given the prime factors of n, there is a fast algorithm for multiplying them
- However going the other way is much hard, especially with large numbers
- There are algorithms but none are efficient

How we assess efficiency of algorithms

- Look at run time T as a function of length len(n) of algorithm input n
- Efficient: T = polynomial function of len(n) (quadratic ect)
- Inefficient: super polynomial (roots)

Relative prime/co-prime

Two numbers are defined as relatively prime (or coprime) if their greatest common factor is 1

However, each of the number need not necessarily be a prime number . e.g. 5,6

Modular addition: (c + d) mod n = (c mod n + d mod n) mod n

Modular subtraction: (c - d) mod n = (c mod n - d mod n) mod n

Modular multiplication = (c x d) mod n = (c mod n x d mon n) mod n

Modular division:

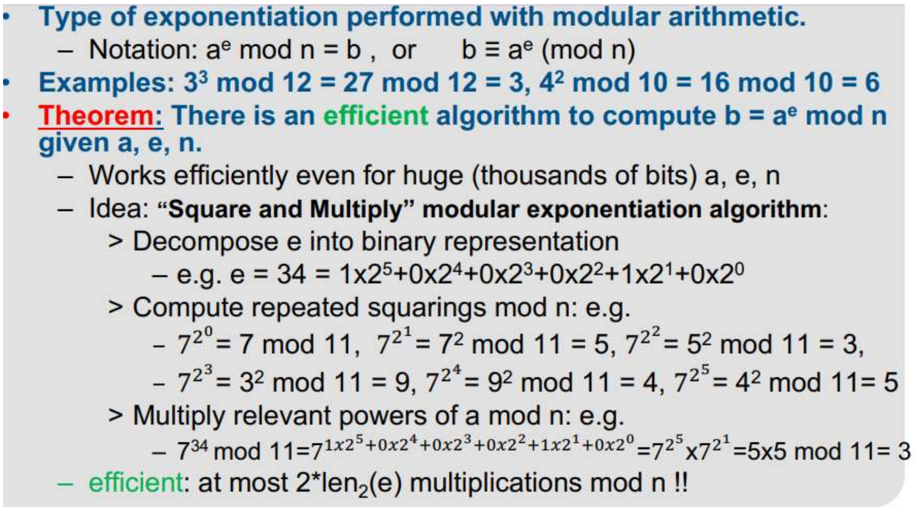
- Let c and d be integers, what is the value of c/d modulo n?
- Usually solved using the expression
  - D x = c mod n (*)
  - Need to find x
- It is possible that
  - There is no number satisfying x
  - X is not a unique value mod n
  - X is a unique value mod n
- Theorem: when d and n are relatively prime, there is a unique solution for x mod n satisfying(*)
  - When d and n are relatively prime, d has a multiplicative invers mod n, 1/d
  - 1/d x d = 1 mod n

Multiplicative inverse: Theorem, when d and n are relatively prime, the multiplicative inverse 1/d mod n exists, and there is an efficient algorithm to compute it given d and n

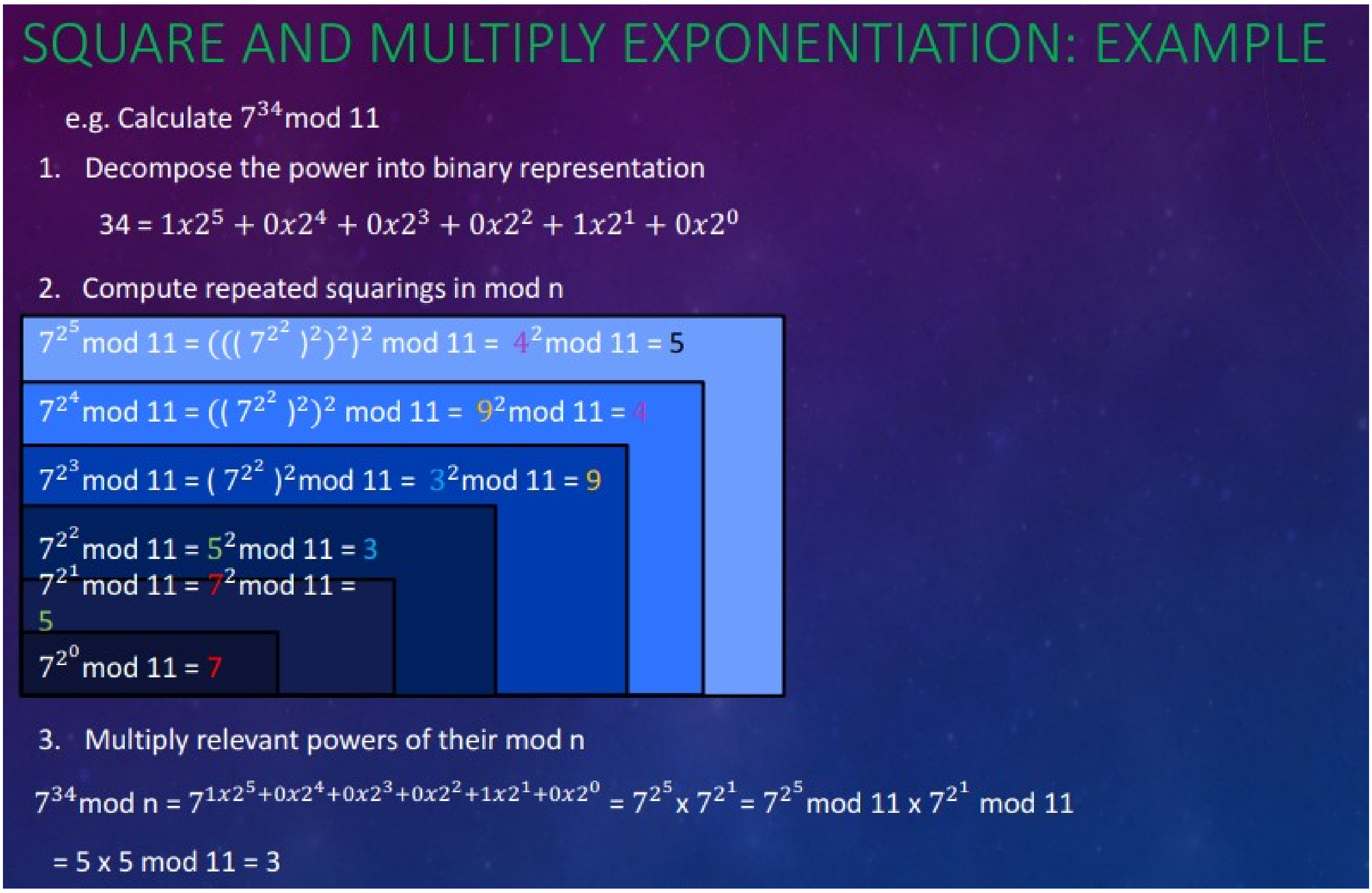
d x = c mod n.

Multiply both sides by 1/d mod n to give x

Modular Exponentiation



- **Type of exponentiation performed with modular arithmetic.**
  - Notation: a^e mod n = b , or b ≡ a^e (mod n)
- **Examples: 3^3 mod 12 = 27 mod 12 = 3, 4^2 mod 10 = 16 mod 10 = 6**
- **Theorem: There is an efficient algorithm to compute b = a^e mod n given a, e, n.**
  - Works efficiently even for huge (thousands of bits) a, e, n
  - Idea: **"Square and Multiply" modular exponentiation algorithm**:
    - > Decompose e into binary representation
      - e.g. e = 34 = $1x2^5+0x2^4+0x2^3+0x2^2+1x2^1+0x2^0$
    - > Compute repeated squarings mod n: e.g.
      - 7^{2^0}= 7 mod 11, 7^{2^1}= 7^2 mod 11 = 5, 7^{2^2}= 5^2 mod 11 = 3,
      - 7^{2^3}= 3^2 mod 11 = 9, 7^{2^4}= 9^2 mod 11 = 4, 7^{2^5}= 4^2 mod 11= 5
    - > Multiply relevant powers of a mod n: e.g.
      - 7^{34} mod 11=$7^{1x2^5+0x2^4+0x2^3+0x2^2+1x2^1+0x2^0}$=$7^{2^5}x7^{2^1}$=5x5 mod 11= 3
  - efficient: at most 2*len_2(e) multiplications mod n !!



SQUARE AND MULTIPLY EXPONENTIATION: EXAMPLE

e.g. Calculate 7^{34}mod 11

1. Decompose the power into binary representation

   $34 = 1x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 0x2^0$

2. Compute repeated squarings in mod n

7^{2^5}mod 11 = $(((7^{2^2})^2)^2)^2$ mod 11 = 4^2mod 11 = 5

7^{2^4}mod 11 = $((7^{2^2})^2)^2$ mod 11 = 9^2mod 11 = 4

7^{2^3}mod 11 = $(7^{2^2})^2$mod 11 = 3^2mod 11 = 9

7^{2^2}mod 11 = 5^2mod 11 = 3

7^{2^1}mod 11 = 7^2mod 11 = 5

7^{2^0}mod 11 = 7

3. Multiply relevant powers of their mod n

7^{34}mod n = $7^{1x2^5+0x2^4+0x2^3+0x2^2+1x2^1+0x2^0}$ = 7^{2^5} x 7^{2^1} = 7^{2^5}mod 11 x 7^{2^1} mod 11

= 5 x 5 mod 11 = 3

Discrete Logarithm Problem

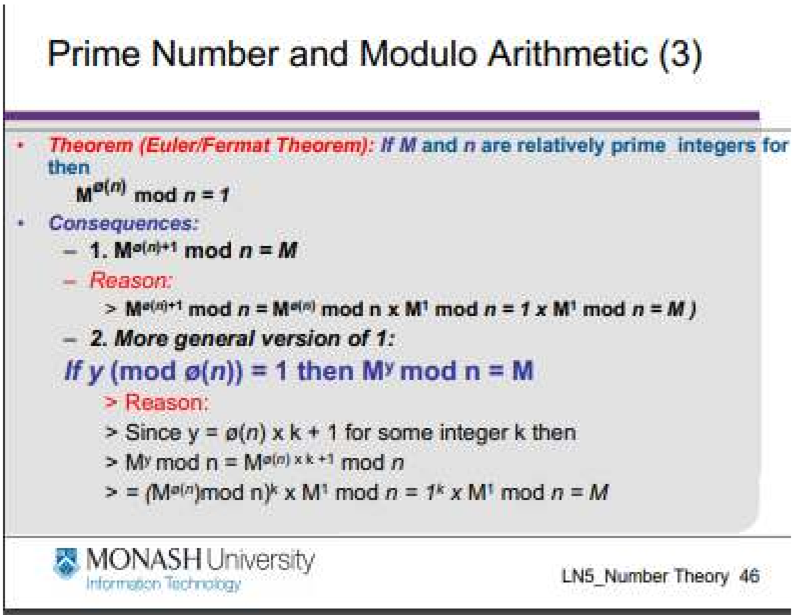
- In contrast to modular exponentiation the discrete logarithm problem asks to go back wards
- Given b = a^e mod n, a and n, compute e
- There is no efficient algorithms to solve this

Euler's Totient function ϕ(n)

- The Eulers totient ϕ(n) of a positive integer n is defined to be the number of positive integers less than n, that are coprime to n
- For prime numbers, this is P -1
- E.g. 33. Counting all the numbers that are not multiples of 3 or 11

If p, q are prime numbers and n = p x q

Ø(n) = Ø(p*q) = (p-1)*(q-1) = p*q – (p + q -1)



Prime Number and Modulo Arithmetic (3)

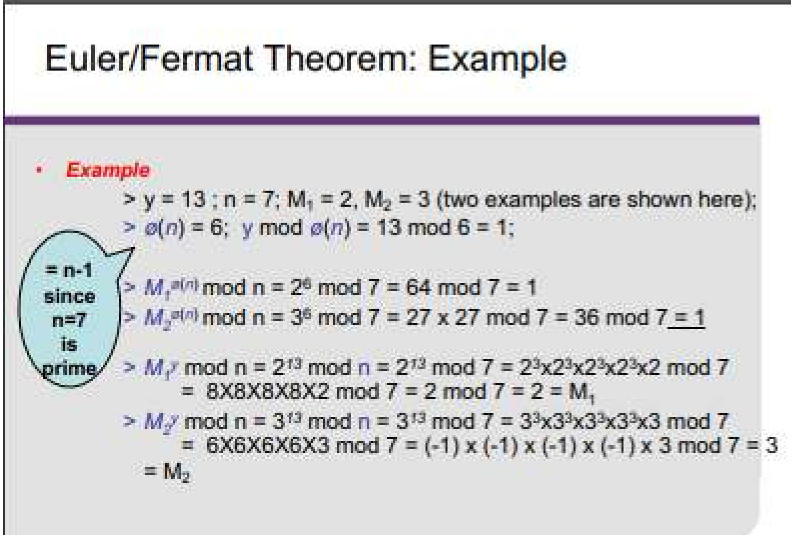
- *Theorem (Euler/Fermat Theorem): If M* and *n* are relatively prime integers for then

  $M^{\varnothing(n)}$ mod $n = 1$
- *Consequences:*
  - 1. $M^{\varnothing(n)+1}$ mod $n = M$
  - *Reason:*
    - > $M^{\varnothing(n)+1}$ mod $n = M^{\varnothing(n)}$ mod n x M^1 mod $n = 1$ x M^1 mod $n = M$)
  - *2. More general version of 1:*

  ***If y* (mod *ø*(*n*)) = 1 then M^y mod n = M**
    - > Reason:
    - > Since y = $\varnothing(n)$ x k + 1 for some integer k then
    - > M^y mod n = $M^{\varnothing(n) \times k + 1}$ mod *n*
    - > = $(M^{\varnothing(n)}$mod n$)^k$ x M^1 mod $n = 1^k$ x M^1 mod $n = M$

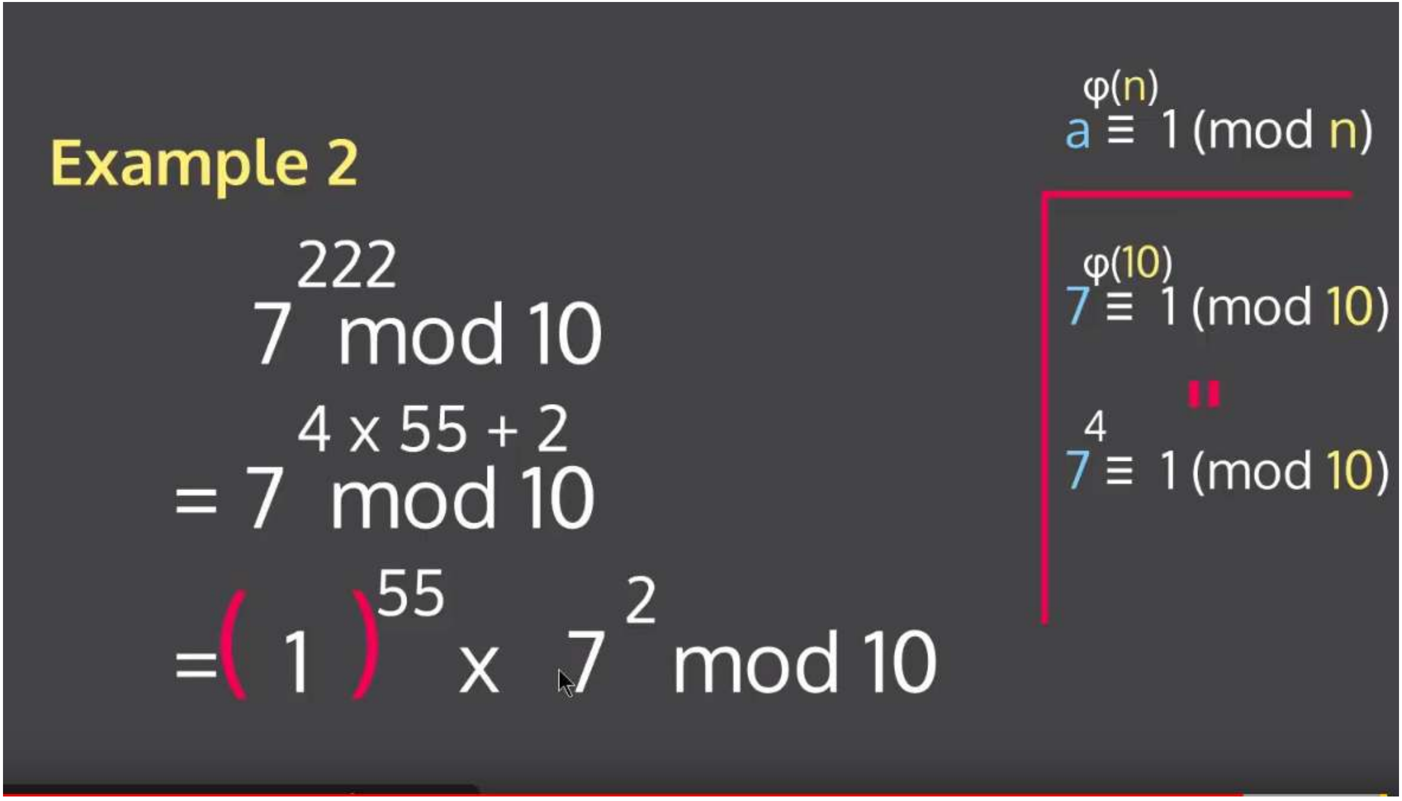
MONASH University
Information Technology

LN5_Number Theory 46



Euler/Fermat Theorem: Example

- *Example*
  - > y = 13 ; n = 7; $M_1 = 2$, $M_2 = 3$ (two examples are shown here);
  - > $\varnothing(n) = 6$; y mod $\varnothing(n)$ = 13 mod 6 = 1;
  - > $M_1^{\varnothing(n)}$ mod n = 2^6 mod 7 = 64 mod 7 = 1
  - > $M_2^{\varnothing(n)}$ mod n = 3^6 mod 7 = 27 x 27 mod 7 = 36 mod 7 = 1
  - > M_1^y mod n = 2^{13} mod n = 2^{13} mod 7 = 2^3x2^3x2^3x2^3x2 mod 7
    = 8X8X8X8X2 mod 7 = 2 mod 7 = 2 = M_1
  - > M_2^y mod n = 3^{13} mod n = 3^{13} mod 7 = 3^3x3^3x3^3x3^3x3 mod 7
    = 6X6X6X6X3 mod 7 = (-1) x (-1) x (-1) x (-1) x 3 mod 7 = 3
    = M_2



Example 2

$$7^{222} \text{ mod } 10$$

$$= 7^{4 \times 55 + 2} \text{ mod } 10$$

$$= (1)^{55} \times 7^{2} \text{ mod } 10$$

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

$$7^{\varphi(10)} \equiv 1 \pmod{10}$$

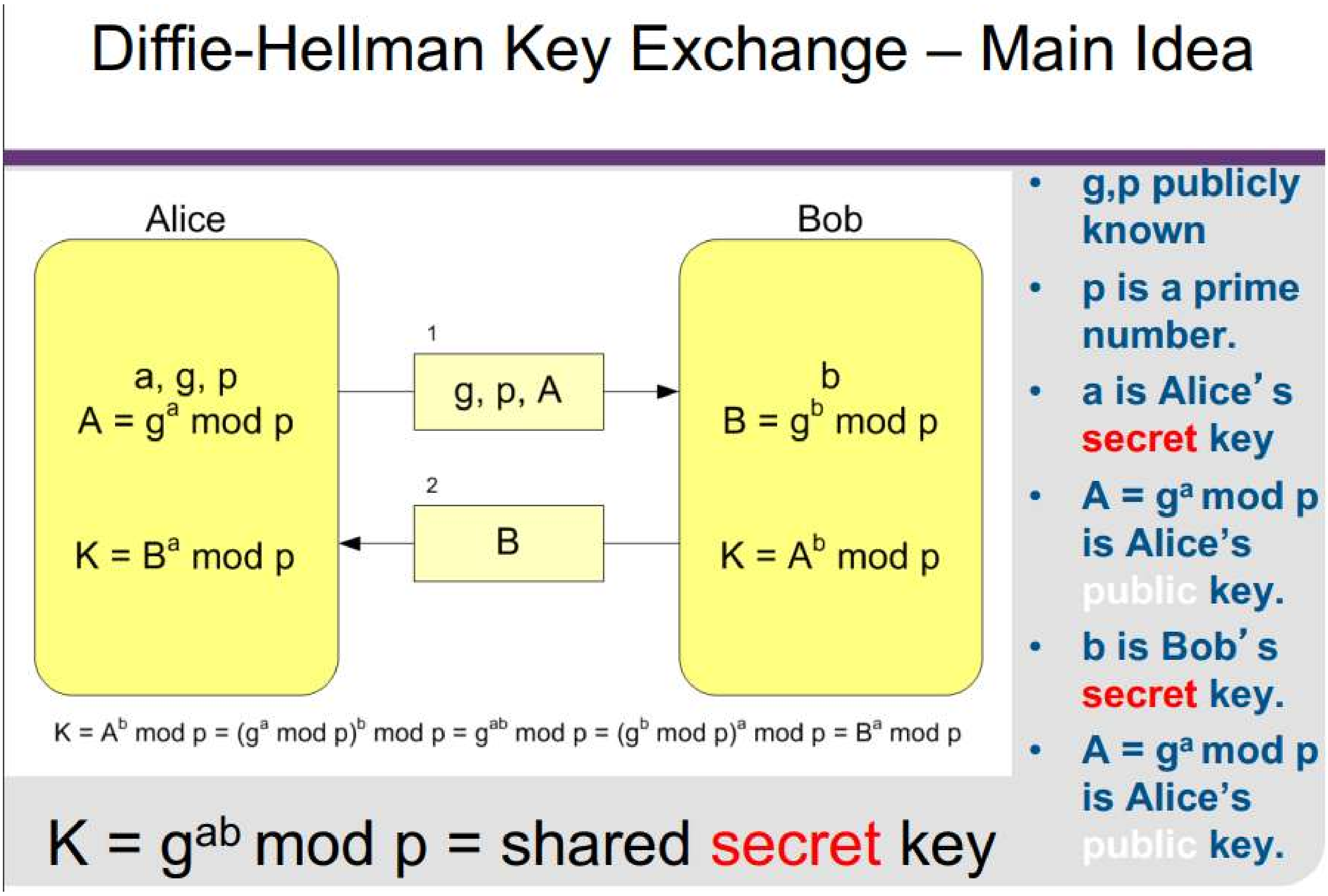
$$\|$$

$$7^{4} \equiv 1 \pmod{10}$$

Week 5 – Public Key Encryption

Public Key Algorithm Requirements

1. Efficiency for legitimate users
   - Computationally easy to crease key-pairs
   - Computationally easy for sender knowing public key to encrypt messages
   - Computationally easy for receiver knowing private key to decrypt cipher text
2. Security against attacks:
   - Computationally infeasible for attacker to determine private key from public key
   - Computationally infeasible for attacker to recover original message from ciphertext and public key



Diffie-Hellman Key Exchange – Main Idea

Alice

a, g, p

$A = g^a \bmod p$

$K = B^a \bmod p$

1

g, p, A

2

B

Bob

b

$B = g^b \bmod p$

$K = A^b \bmod p$

$K = A^b \bmod p = (g^a \bmod p)^b \bmod p = g^{ab} \bmod p = (g^b \bmod p)^a \bmod p = B^a \bmod p$

$K = g^{ab} \bmod p$ = shared secret key

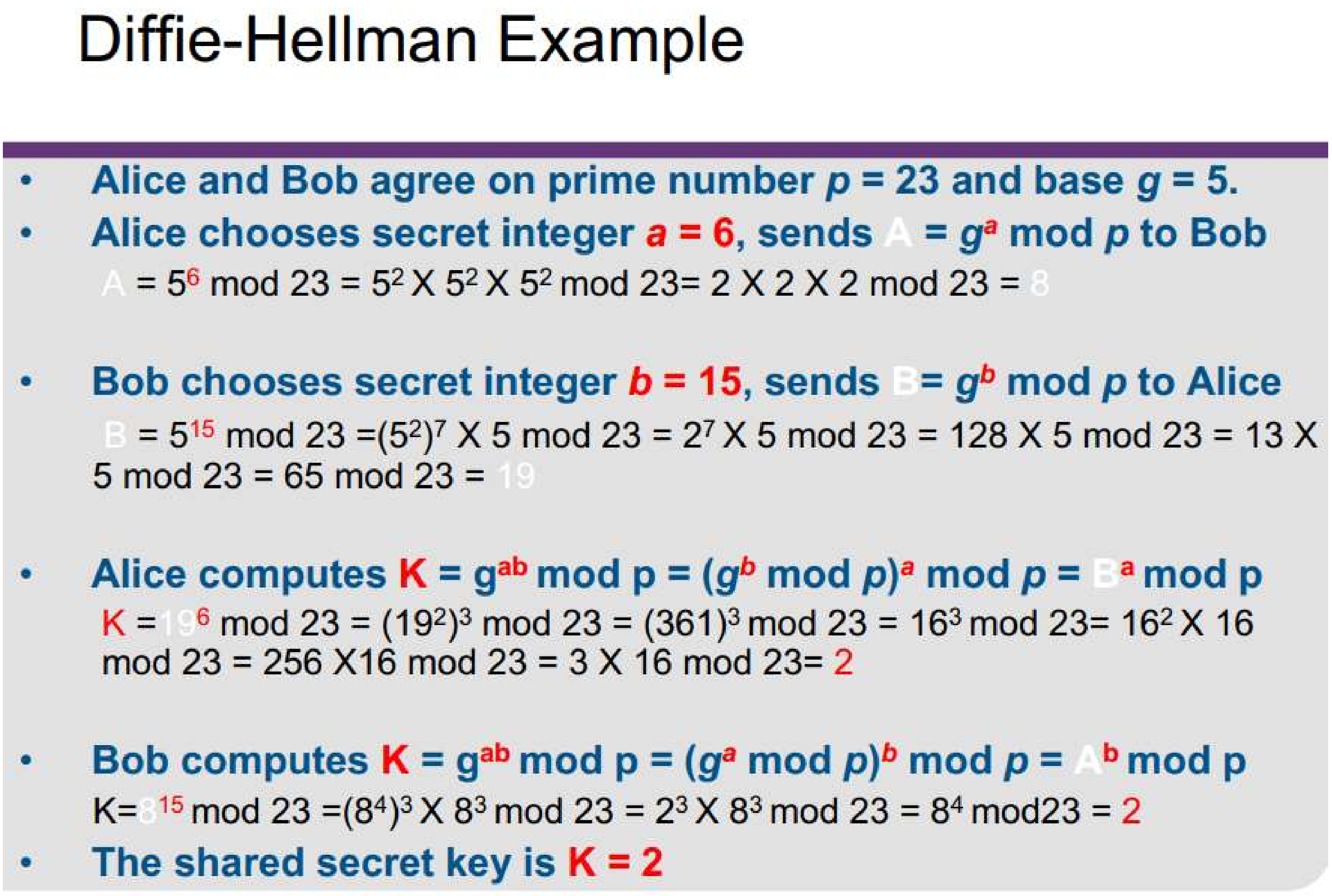
- g,p publicly known
- p is a prime number.
- a is Alice's secret key
- $A = g^a \bmod p$ is Alice's public key.
- b is Bob's secret key.
- $A = g^a \bmod p$ is Alice's public key.

Correctness

- Alice and Bob can easily compute the shared secret key $K = g^{ab}$ mod p using modular exponentiation and their secret keys
  - Alice uses her secret a with Bob's public B: $K = B^a$ mod p
  - Bob uses his secret b with Alice's public A: $K = A^b$ mod p

Security

- Attacker has to solve a computationally hard problem to compute the shared key K
  - Given public g, p, A, B, compute $K = g^{ab}$ mod p



Diffie-Hellman Example

- Alice and Bob agree on prime number p = 23 and base g = 5.
- Alice chooses secret integer a = 6, sends A = g^a mod p to Bob

  A = 5^6 mod 23 = 5^2 X 5^2 X 5^2 mod 23= 2 X 2 X 2 mod 23 = 8

- Bob chooses secret integer b = 15, sends B= g^b mod p to Alice

  B = 5^{15} mod 23 =$(5^2)^7$ X 5 mod 23 = 2^7 X 5 mod 23 = 128 X 5 mod 23 = 13 X 5 mod 23 = 65 mod 23 = 19

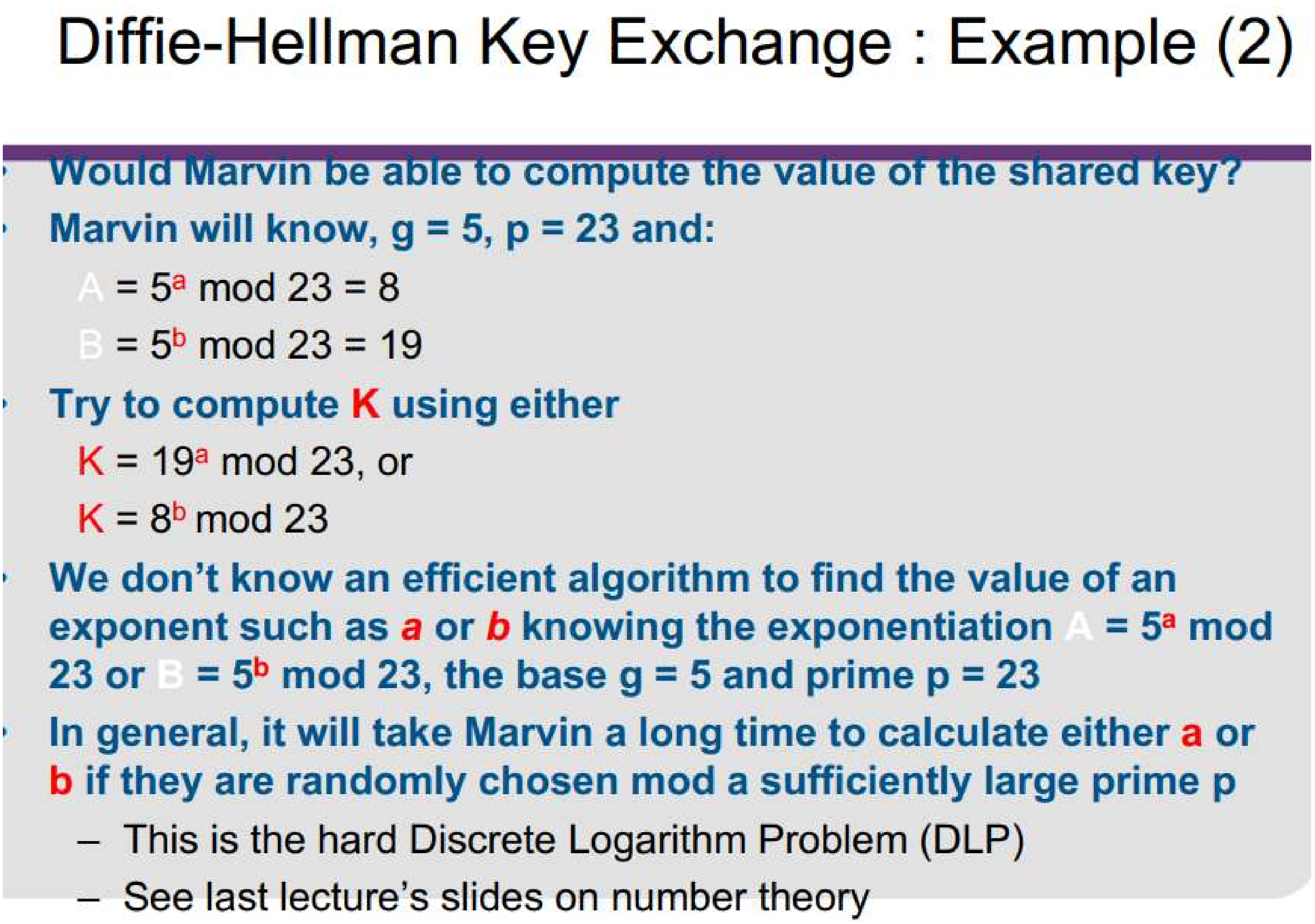
- Alice computes K = g^{ab} mod p = $(g^b$ mod $p)^a$ mod p = B^a mod p

  K = 19^6 mod 23 = $(19^2)^3$ mod 23 = $(361)^3$ mod 23 = 16^3 mod 23= 16^2 X 16 mod 23 = 256 X16 mod 23 = 3 X 16 mod 23= 2

- Bob computes K = g^{ab} mod p = $(g^a$ mod $p)^b$ mod p = A^b mod p

  K=8^{15} mod 23 =$(8^4)^3$ X 8^3 mod 23 = 2^3 X 8^3 mod 23 = 8^4 mod23 = 2

- The shared secret key is K = 2



Diffie-Hellman Key Exchange : Example (2)

- Would Marvin be able to compute the value of the shared key?
- Marvin will know, g = 5, p = 23 and:

  A = 5^a mod 23 = 8

  B = 5^b mod 23 = 19

- Try to compute K using either

  K = 19^a mod 23, or

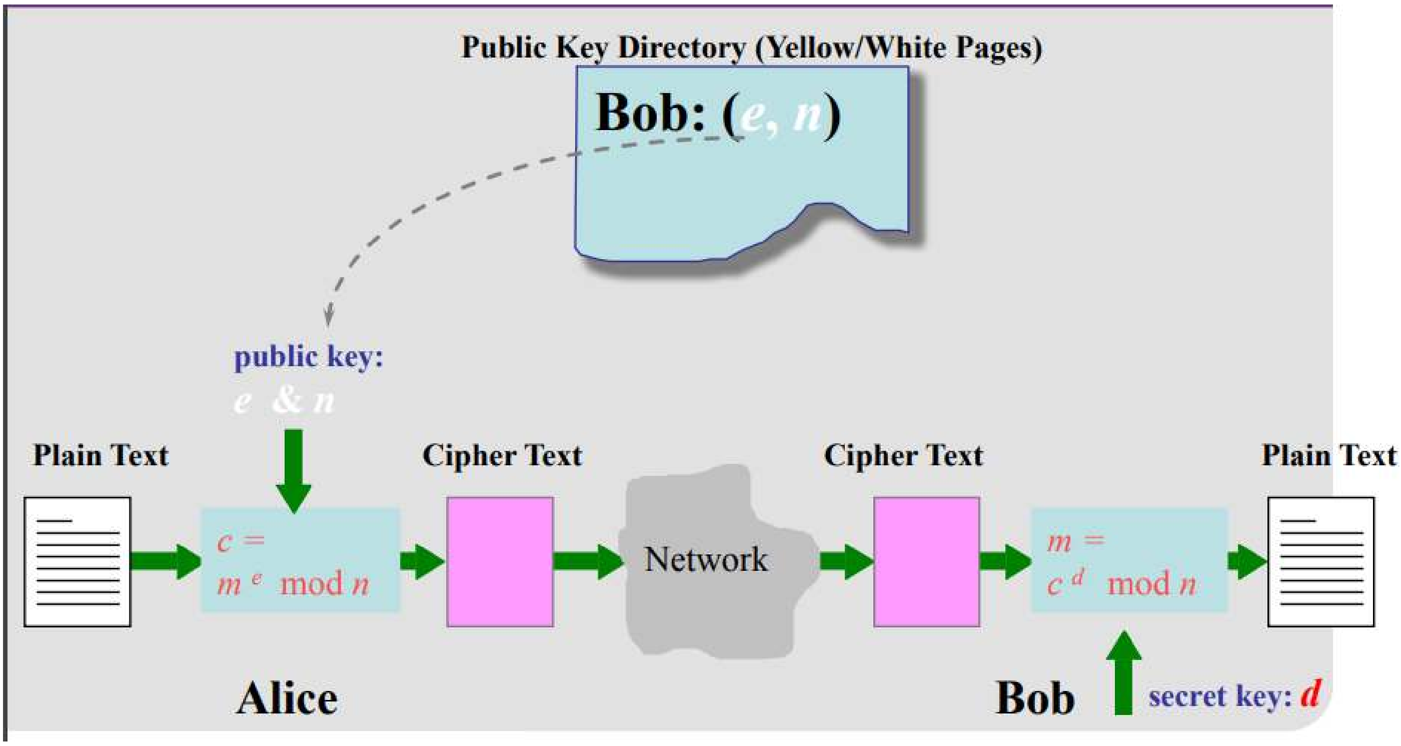
  K = 8^b mod 23

- We don't know an efficient algorithm to find the value of an exponent such as a or b knowing the exponentiation A = 5^a mod 23 or B = 5^b mod 23, the base g = 5 and prime p = 23
- In general, it will take Marvin a long time to calculate either a or b if they are randomly chosen mod a sufficiently large prime p
  - This is the hard Discrete Logarithm Problem (DLP)
  - See last lecture's slides on number theory

El-Gamal Public Key Encryption

- Public Key / Private key pair generated by Bob
  - Selected a large prime p
  - Selected a base g mod p
  - Selects a secret integer b in [0,1 …, p-1)
  - Computes B = g^b mod p
- Encryption of message M by Alice to Bob
  - Alice looks up Bob's public key = (g,p,B)
  - Alice then generated an ephemeral one time pblic key A = g^a mod p and a corresponding random private key a in {0,1,…, p-1}
  - Alice computes the ephemeral (one time) share secret key K= B^a mod p = g^{ab} mod p
  - Message is then encrypted using this key

RSA



Bob

- Chooses 2 large prime numbers, p and q and multiplies them. N = p x q
- Finds out two numbers, e and d such that
  - (e x d) mod ϕ(n) = 1
  - (e x d) mod [(p-1)x(q-1)] = 1
- Public key (e, n)
- Private key (d,n)

Alice: Encryption

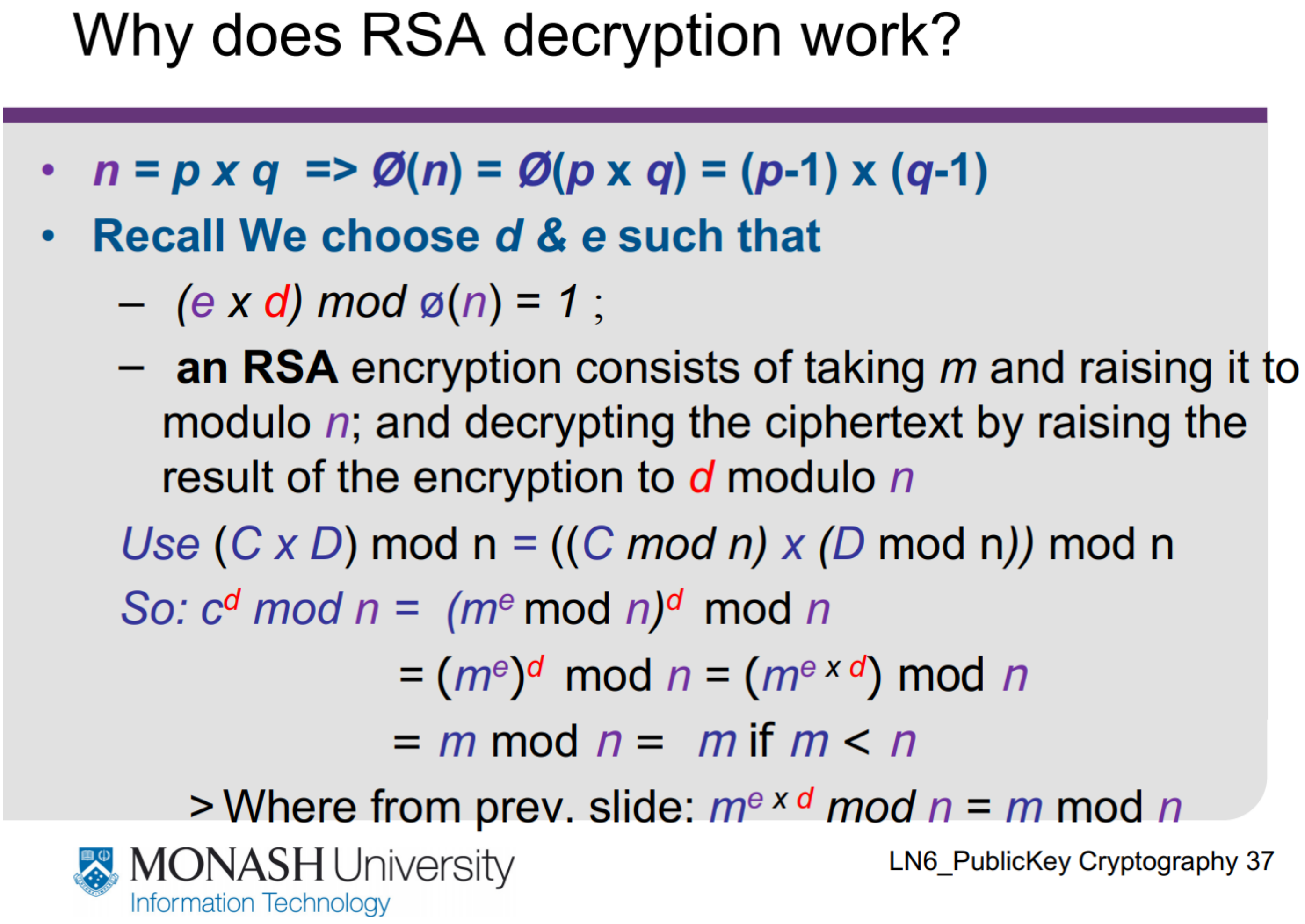
- Has message m to send to Bob
- Finds out Bob's public key (e,n)
- Calculates m^e (mod n) – c
- Sends ciphertext c to Bob

Bob: Decryption

- Uses his matching secret decryption key (d, n) to calculate c^d (mod n) = m

Modular Arithmetic in Cryptography

- Consequence of Euler/Fermat Theorem
  - If m and n are relatively prime intergers and y (mod $\phi(n)$ = 1
  - Then m^y mod n = m
- In RSA Key generation let y = e x d then
  - M^y mod n
  - $M^{e \times d}$ mod n
  - M mod n, since y = mod $\phi(n)$ = e x d mod $\phi(n)$ = 1



Why does RSA decryption work?

- $n = p \times q$ => $Ø(n) = Ø(p \times q) = (p-1) \times (q-1)$
- **Recall We choose *d* & *e* such that**
  - $(e \times d) \bmod ø(n) = 1$;
  - **an RSA** encryption consists of taking *m* and raising it to modulo *n*; and decrypting the ciphertext by raising the result of the encryption to *d* modulo *n*

Use $(C \times D)$ mod n = ((C mod n) x (D mod n)) mod n

So: c^d mod n = $(m^e \bmod n)^d \bmod n$

$= (m^e)^d \bmod n = (m^{e \times d}) \bmod n$

$= m \bmod n =$ m if $m < n$

> Where from prev. slide: $m^{e \times d} \bmod n = m \bmod n$

MONASH University
Information Technology

LN6_PublicKey Cryptography 37

Why is RSA Secure

- An attacker only knows that $C = m^e$ (mod n). He knows that m is an umber between 0 and n – 1. She he would use brute force search to go through all the combinations
- Marvin can try to compute d, the secret key, however he needs to factor n = p x q first, which is the Hard Integer Factorisation Problem. While is is easy to multiply large primes together, it is computationally infeasible to factor or split a large composite into its primes.

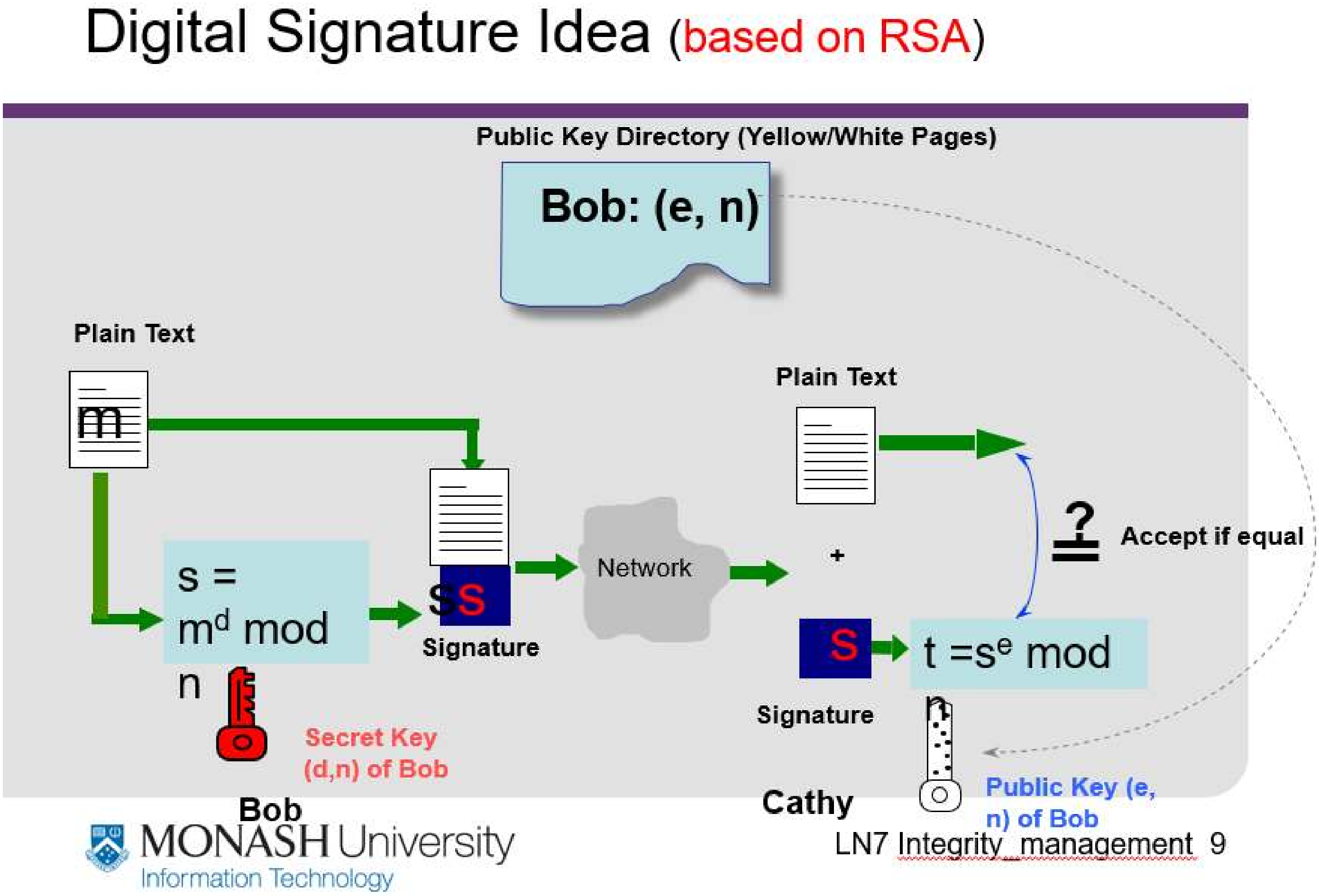
What about active attacks?

- Attacker fakes a public key pretending to be Bob, and generates their own secret key
- To prevent the attack Alice must confirm that she has Bobs correct public key using certificates

Week 6 – Integrity and Authentication

Need for digital signatures

- Social and business activities and their associated documents are becoming digital
- Hand written signatures are not applicable to digital data
- Signatures are used for non-repudiation (Where an entity cannot deceive another by falsely denying responsibility for an act
- It must verify the author and the date and time of the signature
- It must to authenticate the contents at the time of the signature
- It must be verifiable by third parties, to resolve disputes

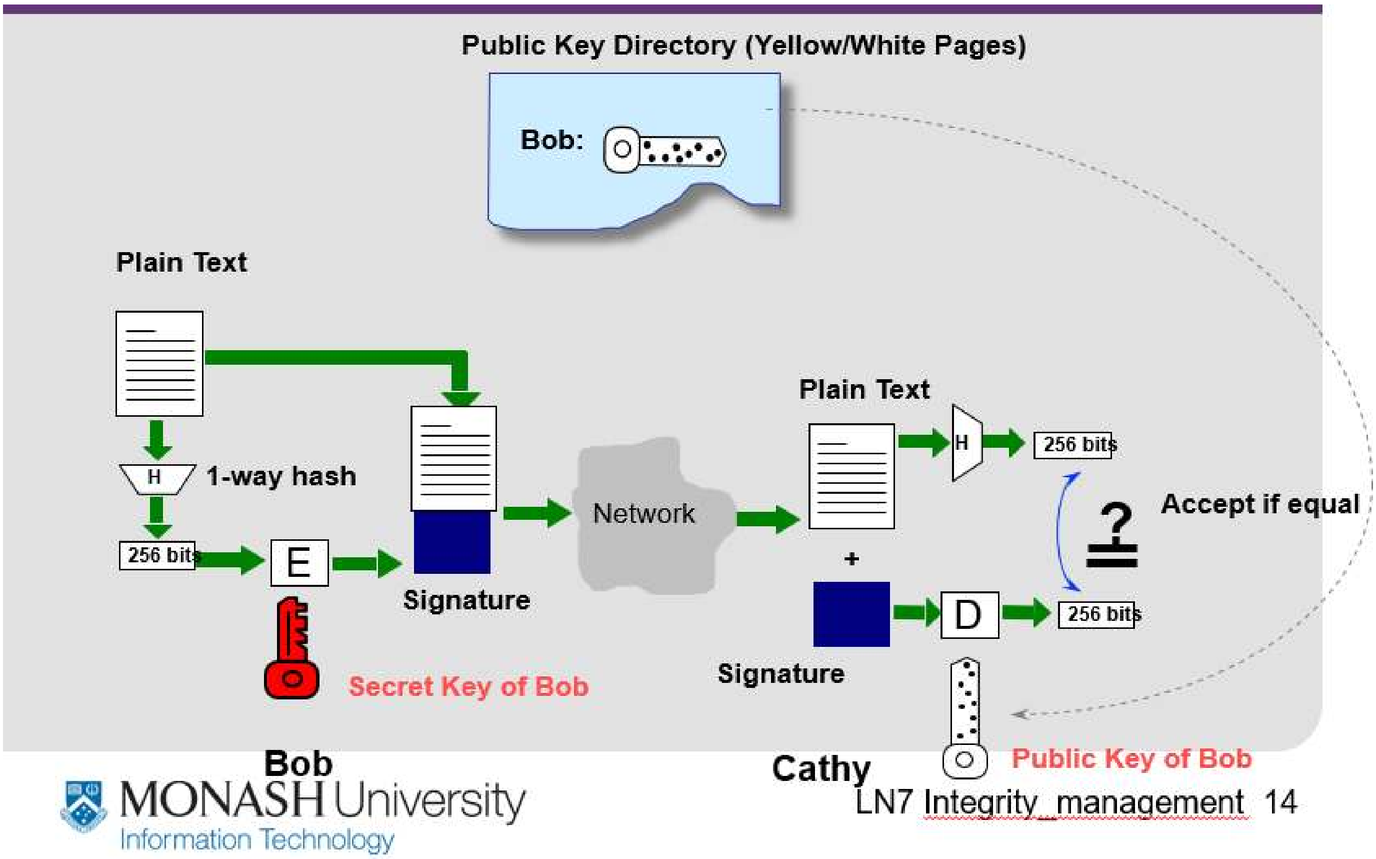


Digital Signature Idea (based on RSA)

LN7 Integrity_management 9

The need for one way hash algorithms

- In the previous example, a document has to be an interger
- To sign a very long document, we need a one-way hash algorithm
- The message gets hashed into a shorter value and signed – benefit is efficiency



A one way hash algorithm hashes an arbitrary length input document into a condensed (short) fixed length output (typically 256 bits)

Properties

1. Handles input data of any size
2. Produces a fixed length output
3. Easy to evaluate
4. Hard to reverse
5. Hard to find collisions

- There is no feasible algorithm to find two or more input documents which are hashed into the same condensed output. This **prevents forgery** when an encrypted hash code is used as in digital signatures. A strong hash function protects **against an attack in which one party generates a message for another party to sign.**

Why digital signatures?

- Unforgeable
- Undeniable by the signatory
- Universally verifiable
- Differs from doc to doc
- Easily implementable

Digital signature properties

- Must be analogous to a handwritten signature
  - Verify author, date and time
  - Authenticate the contents at time of signature
  - Must be verifiable by third parties

Digital Signature Requirements

- Must depend on the message signed
- Must use information to the originator (sender)
- Must be relatively easy to produce
- Must be relatively easy to recognize and verify
- Must be computationally infeasible to forge
- Must be practical to save digital signature in storage

Digital Signature Unforgeability Goal

- Aim for strong unforgeability security goal
  - Allow Choses Message Attacks
    - Signature forging attacker can observe many messages and their corresponding signatures by the target signer
  - Rule out even weakest possible type of forgery

Message Authentication Codes (MACs) Symmetric Key Authentication

Message authentication

Message authentication is generally concerned with:

- Protecting the integrity of the message
- Validating the identity of the originator

Digital signatures provide a general solution, but is often overkill

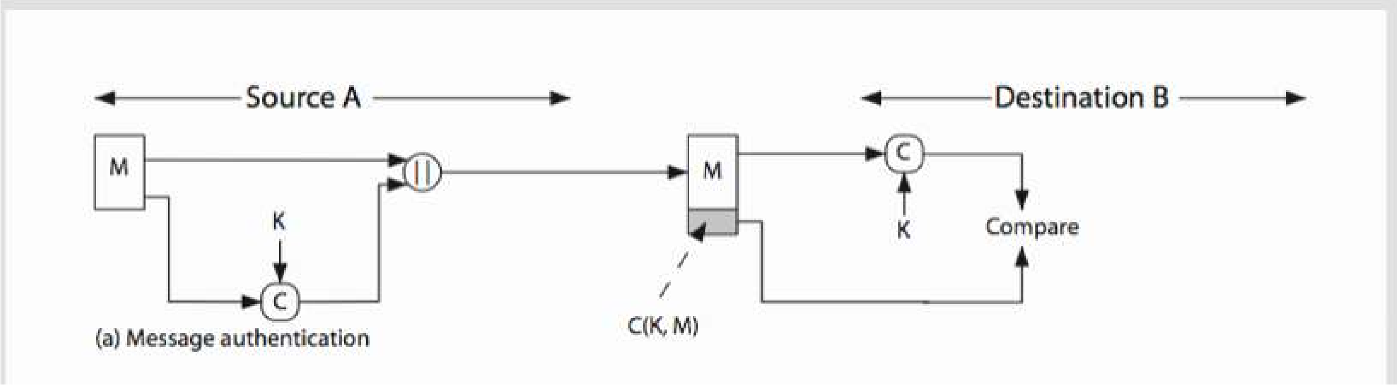
- Non-repudiation of the origin and public verifiability is often not needed

In these cases we use MAC

Message Authentication Code: Symmetric Key version of Digital signature

- Generated by an algorithm that creates a small fixed-sized block: known as cryptographic checksum
  - Depending on both message and some shared secret key
  - Like encryption though need not be reversible
  - Appended to message as an authenticator
- Receiver performs same computation on message and checks if it matches with the received MAC
- Like symmetric key encryption, it is usually much faster and uses shorter keys than signatures
- Provides assurance that message is unaltered and comes from sender
- Does not provide non-repudiation or public verifiability.

An alternative authentication technique involves the use of a **secret key to generate a small fixed-size block** of data, known as a **cryptographic checksum or MAC** that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K. A MAC function is similar to encryption, except that the MAC algorithm need not be reversible, as it must for decryption.



(a) Message authentication

MAC Properties

MAC can be viewed as a cryptographic checksum

- Condenses a variable-length message M
- Using a secret key K
- To a fixed size authentication

Is a many to one function

- Potentially many messages can generate the same MAC
- But finding those messages should be difficult

Requirements for MACs

Need the MAC to satisfy the following:

1. knowing a message and MAC, is infeasible to find another message with same MAC
   - deals with message replacement attacks
2. MACs should be uniformly distributed across the messages
   - deals with the need to thwart a brute-force attack based on chosen plaintext
3. MAC should depend equally on all bits of the message
   - dictates that the authentication algorithm should not be weaker with respect to certain parts or bits of the message than others.

How to build a MAC

Two common ways

- using block ciphers (CBC mode for authentication)
- using one-way cryptographic hash functions (HMAC)

MACs Using Symmetric Ciphers

- Can use any block cipher chaining mode and use the final block as a MAC
- However, this needs to be done carefully to deal securely with arbitrary message lengths
- Data Authentication Algorithm (DAA) was a widely used MAC based on DES-CBC
  - But this MAC is no longer recommended for use as it is too small for sufficient security, and has other problems
- Modern replacement: CMAC (NIST, 2005)
  - Uses two keys K1, K2 derived from a single key K
  - Works with any secure block cipher (e.g. AES-128)

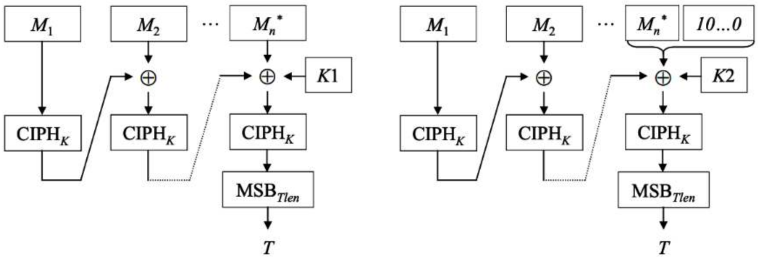


Figure 1: Illustration of the two cases of MAC Generation.

The two cases of MAC Generation are illustrated in Figure 1 above. On the left is the case where the message length is a positive multiple of the block size; on the right is the case where the message length is not a positive multiple of the block length.

MACs using one way hash functions

- Like MAC, hash function is a one way function that accepts a message M and produces a fixed-size message digest H(M)
- Unlike MAC, hash function takes only the message (not the key) as input to generate the message digest (hash value)
- Hash is used to detect changes to message
- Can be used with both symmetric and public key cryptography
    - Hashing the message in digital signatures (earlier today)
    - Can also be used to design a MAC!

Hash Functions: Attacks

Two attack approaches

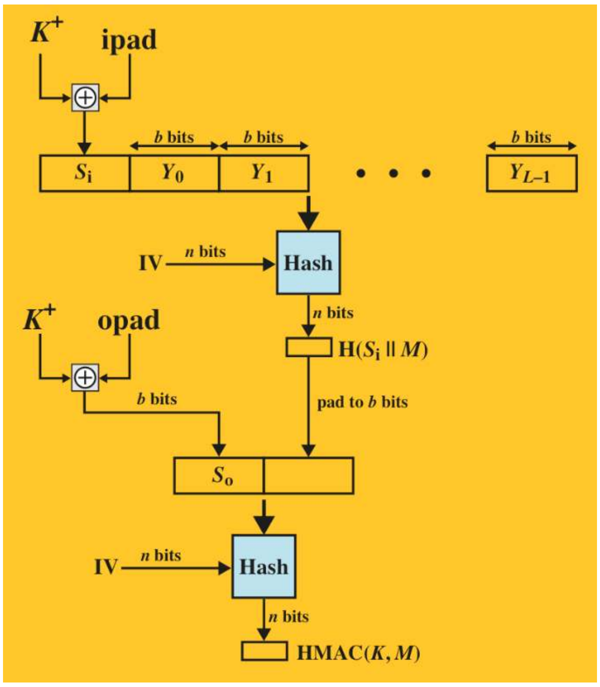
- Cryptanalysis exploiting weakness in algorithm
- Brute force attack: trial many inputs, strength proportional to size of hash code (2^n/2)

HMAC Design Objectives

- use, without modifications, hash functions
- allow for easy replaceability of embedded hash function
- preserve original performance of hash function without significant degradation
- use and handle keys in a simple way.
- have well understood cryptographic analysis of authentication mechanism strength

HMAC

- specified as Internet standard RFC2104
- uses hash function on the message:
    - HMAC(K,M)= Hash[(K+ XOR opad) ||
- Hash[(K+ XOR ipad) || M)]]
    - where K+ is the key padded out to size
    - opad, ipad are specified padding constants
- overhead is just hash calculations on 3 more blocks than hashing the message alone
- any hash function can be used
    - eg. MD5, SHA-1, RIPEMD-160 (not recommended)
    - SHA-2, SHA-3 (recommended)



HMAC Security

- proved security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
    - brute force attack on key used (2n)
    - birthday attack (but since keyed would need to observe a very large number of messages)
    - Collision resistant attacks, an adversary wishes to find 2 messages that yield the same hash (2n/2)
- choose hash function used based on speed verses security constraints

Achieving Integrity + Confidentiality: How to Combine Authentication + Encryption

- In secure communication (e.g. secure client to web server), need both:
    - Confidentiality of the messages
    - Integrity/authentication of the messages
- Can achieve it by combining separate authentication and encryption
    - E.g. encrypt-then-MAC
    - use separate keys for encrypt and MAC to avoid security vulnerabilities
    - Needs to be done carefully to preserve security of both mechanisms
- Efficient and secure authenticated encryption modes of operation of block ciphers exist for this purpose (e.g. GCM mode, NIST 2007, will not cover in detail)

Integrity + Confidentiality: public key

- if public-key encryption / signature is used:
- Pub key encryption provides no authentication of sender
    - since anyone potentially knows public-key
- however if
    - senders sign the message using their private-key
    - then encrypt with recipients public key
    - have both secrecy and authentication
- cost of two public-key mechanisms
-

(d) Public-key encryption: confidentiality, authentication, and signature

Week 7 Entity Authentication

- Process of verifying an identity claimed by or for a system
- Has two steps
  - Identification – specify identifier
  - Verification – bind entity (person) and identifier

Means of User Authentication

- four means of authenticating user's identity
- based one something you
  - know - e.g. password, PIN (SYK)
  - possess - e.g. key, token, smartcard (SYH)
  - are (static biometrics) - e.g. fingerprint, retina (SYA)
  - do (dynamic biometrics) - e.g. voice, sign (SYA)
- can use alone or combined
- all can provide user authentication
- all have issues

Password authentication

- widely used user authentication method
  - user provides name/login and password
  - system compares password with that saved for specified login
- The user ID:
  - determines that the user is authorized to access the system
  - determines the user’s privileges
  - is used in discretionary access control

Simple attack countermeasures

- Online password guessing countermeasures:
  - account lockout mechanisms
  - policies against using common passwords but instead hard to guess passwords
  - training & enforcement of password policies

- Device hijacking:
    - *automatic logout*
- electronic monitoring
    - encrypted network links (e.g. SSL/TLS: later)

Hard to guess password: counting

- How many 8 character passwords are possible?
    - Assume each character is lower/upper case letter
    - Assume each character is lower/upper letter or digit
- How many 9 character passwords are possible?
- 8 char low/up case + digits: (26*2+10)^8 = 62^8 ~ 10^(14.3)
- 9 char low /up case: (26*2)^9 = 52^9 = 10^(15.4) (larger by more than 10 times than above)

User account management

- Never create an account or allow use of a device without setting up a well chosen user authentication password.
- Keep an eye on inactive accounts.

Hard to guess passwords: Password management

- Not allowing a password that is similar to the user name, derivative (copied) or the 'known' words in the system dictionary.
- Make sure that the rules in the "educating users" are applied.
- Implementing password aging.

How to store passwords in a system

- Storing passwords in clear form is dangerous:
- Passive exposure threat for password file
- Idea: Store a transformed password f(P) in file
- Goal: Infeasible for an attacker exposing stored transformed password f(P) and transform f(), to:
- recover the actual password P, or even
    - recover any other password P' such that f(P') = f(P) (a "collision" for f)
- Common Operating Systems use this idea using a cryptographic one-way (hash) function f (see last lecture!)

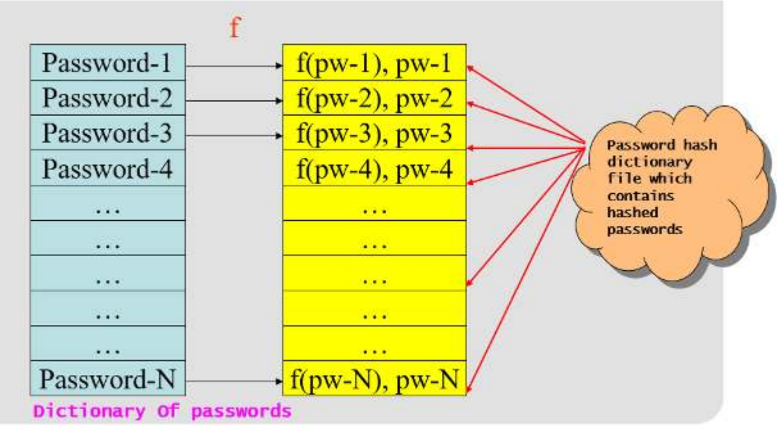
Which function to use for hashing passwords?

- For password hashing, should use for hash function f specially designed standard password based key derivation one-way function (PBKDF)
    - Examples: PBKDF2, scrypt, Argon2
    - A kind of mode of operation of hash functions
- Idea:
    - Slow down a cryptographic one-way hash function , T_f ~ 100 ms (typ., adjustable by user)
    - Function uses a lot of memory (increase cost of special-purpose hardware implementations)
    - Time still unnoticeable for single hash (verification), but significantly slow down brute-force attacks (slow-down factor ~ 1 million!)

Precomputation- based dictionary attack

- Password Crackers are known to use a precomputation-based dictionary attack
    - Precomputation (offline) attack phase (slow)

        - Create a dictionary of possible passwords P
        - For each P in dictionary, compute hash f(P)
        - Store (f(P), P) pairs in table addressable by f(P) ("reverse dictionary")
- Password lookup attack phase (very fast):
    - When system password hash file exposed, look up each file hash in reverse dictionary
    - Expose many system passwords very quickly.



How to prevent Precomputation Dictionary Attack: Salting a password hash function

- Precomputation phase of the attack will be infeasible if table is too big to precompute

- Countermeasure Idea: append randomness ("salt") R to password P before hashing P with PBKDF

    Instead of storing f(P), store (R, f(P,R))

    Salt R is randomly chosen L bit string (say L = 128 bit) chosen independently for each password

- Precomputation attack now may need to guess salt R in advance, or compute 2^L tables (one for every possible R value)

    → precomputation attack infeasible if L large enough!

Shadow Password File: Why do we need it?

- A system without a shadow password file keeps the information about the user details in a file that can be accessed by the normal user in the system.
- Normal users in the system may need to access the password file because it may contain information about the users such as full name.
- We can hash the password and keep the hashed password in the password file, however, it is still possible to attack for weak passwords by brute force search.
- To prevent the attack on weak passwords but allowing normal users to access the user information:
    - Remove the encrypted password replace it with an "x" or "*" in the password file and place the hashed password in a file accessible only by the superuser (called shadow password file).
    - Still, the password hashing protects strong passwords against any expsure of the shadow password file

Password Cracking

- If the system does not have shadow password, the /etc/passwd can be easily copied and the attack can be done offline.

- Main technique is to generate hashed version of potential password and compare it with the hashed password entry in /etc/passwd.

- Methods of attack:

Brute Force

> Try all possible combinations of alphanumeric/digits/symbols.

Dictionary Attack

Crackers need to have

- Where is the password or its derivative stored in the system?
- Hash function f used to hash the password (reverse engineering, Google)

Password aging

- Forces the users to change their password regularly.

  Prompt the user with change password screen during the logging process when the "age" of the password has reached the maximum allowable limit.

  If someone else knows the password, the damage is only for that period of time before it is changed.

- Additional security measure to password aging.

  Maintain password history and prevent re-use of earlier passwords.

One time password

- Challenge will vary every time and hence each challenge is used only once
  - Short time or One-off password
- Other examples:
- SMS, token for internet banking

Using better passwords

- clearly have problems with passwords
- goal to eliminate guessable passwords
- whilst still easy for user to remember
- techniques:
  - user education
  - computer-generated passwords
  - reactive password checking
  - proactive password checking

Token Authentication

- object user possesses to authenticate, e.g.

  memory card

  smartcard

Memory Card

- store but do not process data
- magnetic stripe card, e.g. bank card
- electronic memory card
- used alone for physical access
- with password/PIN for computer use
- drawbacks of memory cards include:
  - need special reader
  - loss of token issues

  - user dissatisfaction

Smart card

- credit-card like
- has own processor, memory, I/O ports
  - wired or wireless access by reader
  - may have crypto co-processor
  - ROM, EEPROM, RAM memory
- executes protocol to authenticate with reader/computer
- also have USB dongles

Smart phone

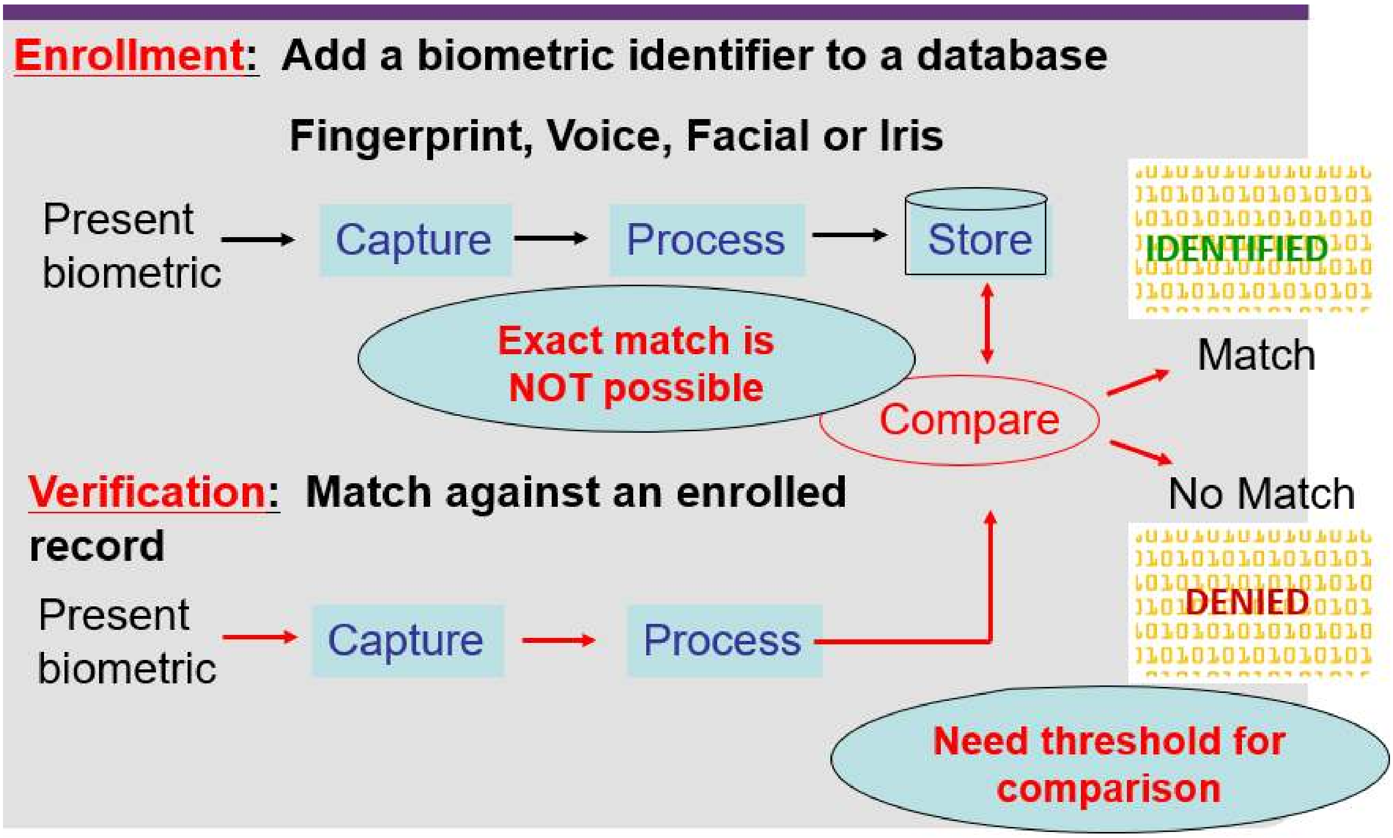
- Advantages:
  - usually already owned by users (no special-purpose hardware token needed)
  - Token authentication software easily installed via small app
- Drawback:
  - Higher likelihood of malware compromise compared to special-purpose hardware token
  - Many other apps running on same device!

Something you are

- Use of human Biometric(s)
- A characteristic of the body
- Presumed to be unique and invariant over time.
- Common biometrics:
  - Fingerprint
  - Iris Scan
  - Retinal Scan
  - Hand Geometry (palm)
  - Facial recognition
  - Voice, typing, key strokes, signature pressure

Stable vs Alterable Biometric

- Stable biometric
  - Does not change from one session to another.
  - Relatively static
  - Fingerprint, Face, Hand, Iris, Retina
- Alterable biometric
  - Can be altered easily if necessary
  - Voice, Keystrokes pattern



Enrollment: Add a biometric identifier to a database

Fingerprint, Voice, Facial or Iris

Present biometric → Capture → Process → Store

Exact match is NOT possible

Compare

Match

No Match

Verification: Match against an enrolled record

Present biometric → Capture → Process

Need threshold for comparison

Facial recognition

- The oldest method of biometric!
- There is widespread acceptance (and requirement !) for photo ID
- The issuing of *other* authentication devices (like passwords, key cards, digital signatures) usually depends on facial recognition by the agents of the issuing authority.
- Photo-ID may not be reliable, but has a very significant deterrent (restriction) effect.

Matching vs. non-matching

- These two types of error are known as false match and false non-match.
- A false match is when two pieces of biometric data from different people are judged to be from the same person, as in (a). This type of error is sometimes called a false accept (FALSE POSITIVE).
- A false non-match is when two pieces of biometric data from the same person are judged to be from different people, as in (b). This type of error is sometimes called a false reject (FALSE NEGATIVE).

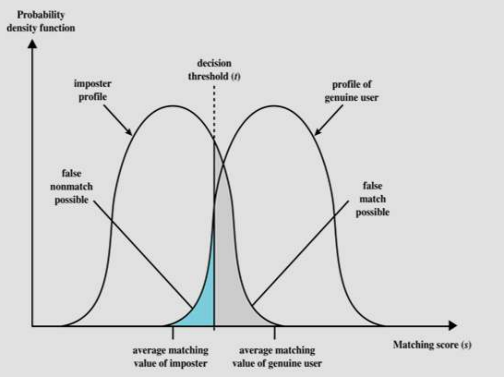
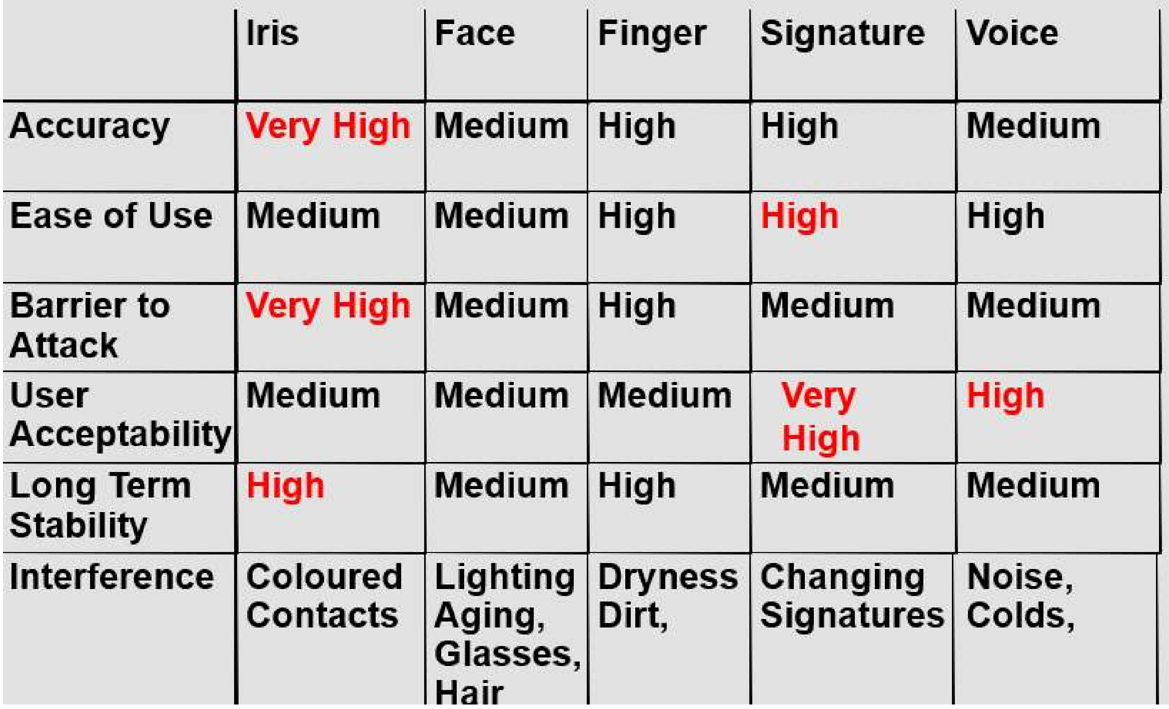


Figure 3.8 Profiles of a Biometric Characteristic of an Imposter and an Authorized Users In this depiction, the comparison between presented feature and a reference feature is reduced to a single numeric value. If the input value (s) is greater than a preassigned threshold (t), a match is declared.

How is accuracy measured?

- False Negative (rejection) rate (FRR)

    - Measures how often an authorized (genuine) user, who should BE recognized by the system, is not recognized.
    - I am not recognised as me!
- False Positive (acceptance) rate (FAR)
    - Measures how often an unauthorized (imposter) user, who should NOT be recognized by the system, is falsely recognized.
    - You are pretending to be me!



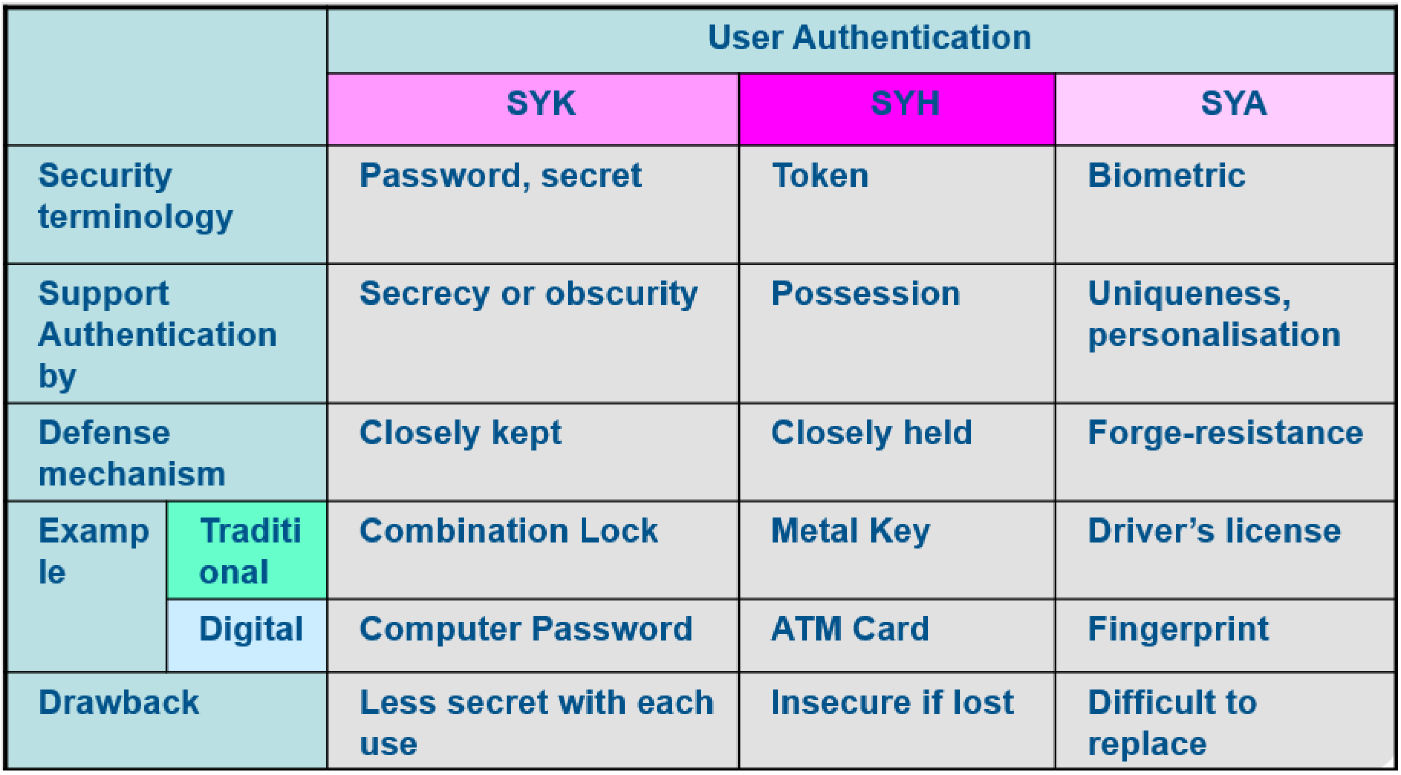
	Iris	Face	Finger	Signature	Voice
Accuracy	Very High	Medium	High	High	Medium
Ease of Use	Medium	Medium	High	High	High
Barrier to Attack	Very High	Medium	High	Medium	Medium
User Acceptability	Medium	Medium	Medium	Very High	High
Long Term Stability	High	Medium	High	Medium	Medium
Interference	Coloured Contacts	Lighting Aging, Glasses, Hair	Dryness Dirt,	Changing Signatures	Noise, Colds,

What happens if biometric info is stolen?

- Unlike static password that can be changed easily when a malicious activity is detected, it is not possible to change stable biometric information.
- It is possible to change the alterable biometric.

  e.g. change the phrases spoken for voice recognition
- Some biometric information is easier to duplicate or to forge compared to others, eg fingerprint is easier to obtain compared to iris.
- Multimodal can be used to overcome the issue.

  For example, use the combination of fingerprint and iris.
- “Behavior traits (genetic)” as biometric?



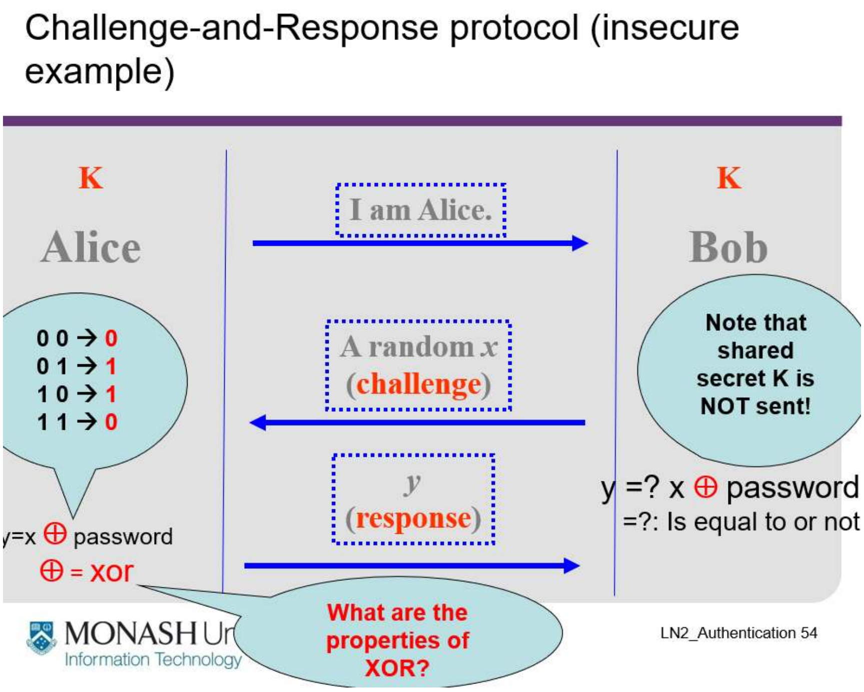
		User Authentication		
		SYK	SYH	SYA
Security terminology		Password, secret	Token	Biometric
Support Authentication by		Secrecy or obscurity	Possession	Uniqueness, personalisation
Defense mechanism		Closely kept	Closely held	Forge-resistance
Example	Traditional	Combination Lock	Metal Key	Driver's license
	Digital	Computer Password	ATM Card	Fingerprint
Drawback		Less secret with each use	Insecure if lost	Difficult to replace

Entity authentication security threats

- client attacks
  - adversary attempts to achieve user authentication without access to the remote host
- host attacks
  - directed at the user file at the host where passwords, token passcodes, or biometric templates are stored
- Eavesdropping
  - attempts to learn the password by physical proximity to user
- Replay
  - adversary repeats a previously captured user response
- trojan horse
  - an application or physical device masquerades as an authentic application
- denial-of-service
  - attempts to disable a user authentication service by flooding the service with numerous authentication attempts

Remote user authentication

- authentication over a network is more complex
  - threats of eavesdropping, capturing a password, replaying an authentication sequence
  - E.g. phishing threat for passwords!
- Common solution: use a token-based a challenge-response protocol
  - Setup: Token stores secret key K shared with verifiying host
  - Authentication Protocol:
    - User sends identity to host
    - host responds with challenge x (random / current time)
    - User's token computes and sends back response y = f(x,K)
    - host compares y from user with own computed f(x,K), if match user authenticated
- protects against eavesdropping and replay threats



Challenge-and-Response protocol (insecure example)

Note that shared secret K is NOT sent!

What are the properties of XOR?

Application of MAC to Challenge-Response user authentication

- Solution: To make a secure protocol against eavesdropping attacks, use a secure MAC
  - Verifier sends random challenge “message” x
  - Prover responds with R = MAC(K, x)
  - Verifier checks if R = MAC(K, x)
  - Protocol is secure against eavesdropping if MAC is existentially unforgeable under chosen message attack and x is suff. long to avoid x repeats (e.g. 256 bit).
- MAC challenge-response Variant 1:
  - Challenge x may be time of day instead of random
  - Advantage: Using synchronised clocks, no need for explicit challenge message to be transmitted

Application of digital signatures to Challenge-response user authentication

- **Problem with MAC-based challenge-response protocol:**
  - Verifier Server needs to store shared key K
  - What if hacker exposes Server’s stored key K?
- **Solution: Replace MAC with a digital signature**
  - Setup: prover generates sk, sends pk to verifier.
- Protocol:
  - Verifier sends random challenge “message” x
  - Prover responds with y = Sign(sk, x)
  - Verifier checks if y is a valid signature on x with respect to prover’s public key pk
- Protocol is secure against eavesdropping if signature is existentially unforgeable under chosen message attack and x is suff. long to avoid x repeats (e.g. 256 bit).

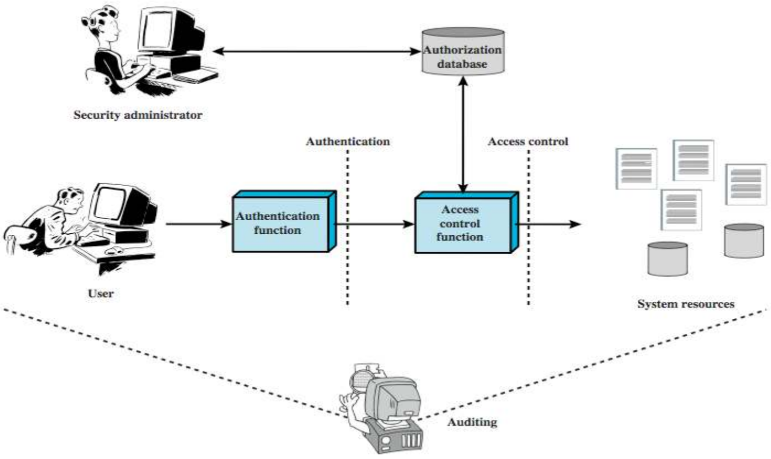
Multifactor User Authentication

- Multifactor authentication: combine more than one authentication method to improve security:
- Attacker needs to compromise all used factors to attack the system
- → reduce likelihood of successful attack
- E.g. common combination:
- Something you know (password) +
- Something you have (token)

Week 8 – Access Control

Access Control Principles

- Definition: "The prevention of unauthorised use of a resource, including the prevention of use of a resource in an unauthorised manner"
- central element of computer security
- assume have users and groups
    - authenticate to system
    - assigned access rights to certain resources on system

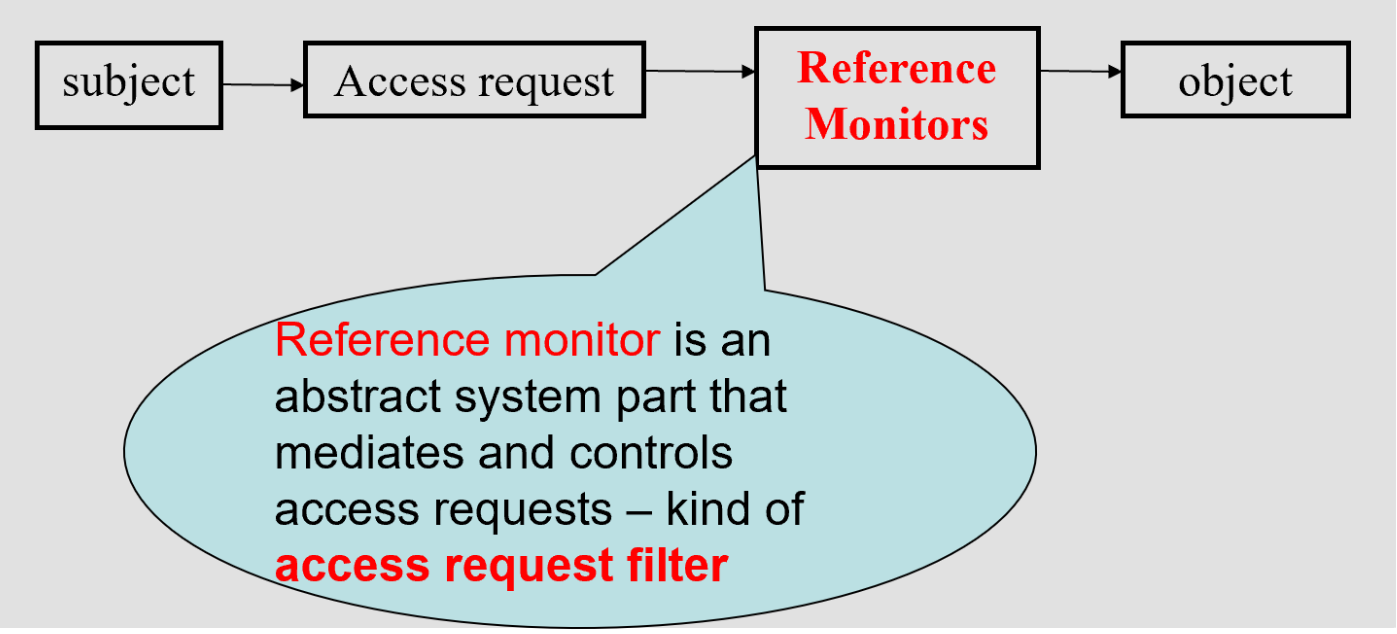


Access Control Elements

- subject - entity that can access objects
    - a process representing user/application
    - often have 3 classes: owner, group, world
- object - access controlled resource
    - e.g. files, directories, records, programs etc
    - number/type depend on environment
- access right - way in which subject accesses an object
    - e.g. read, write, execute, append, delete, create, search

What do we mean by access control?

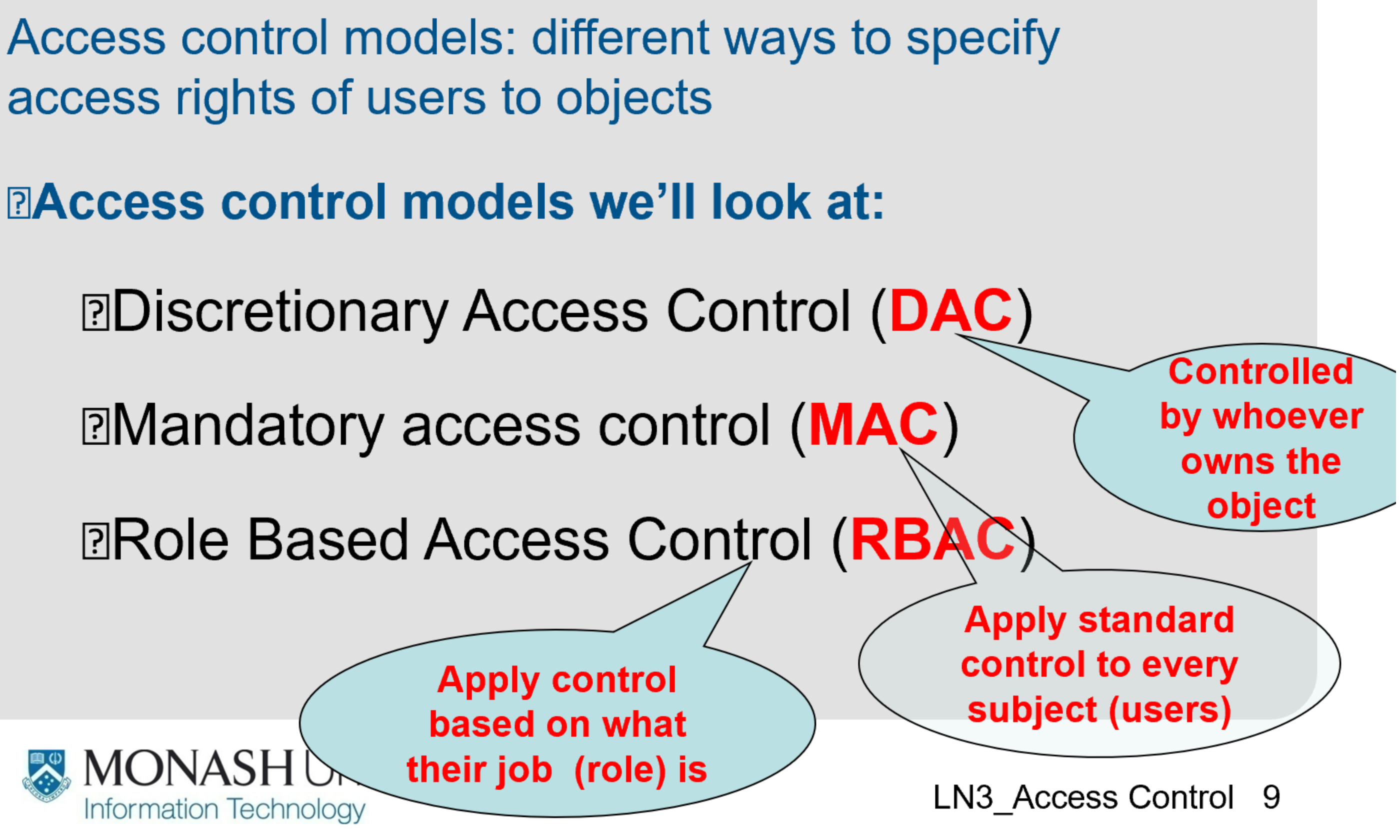
- Rules that define which subject can access what object
- Q: What does the term ACCESS mean with respect to information resources?
- normally read, write, execute
- e.g. Unix OS uses these operations
- The file owner can control the permissions to these operations.
- Windows OS has, in addition to these, permissions for *delete, change ownership and change permissions*
- *Some systems define write that includes reading rights. These systems often have one more operation à append (blind write)*



subject → Access request → Reference Monitors → object

Reference monitor is an abstract system part that mediates and controls access requests – kind of **access request filter**

Access Control Models



Access control models: different ways to specify access rights of users to objects

- **Access control models we'll look at:**
  - Discretionary Access Control (**DAC**)
  - Mandatory access control (**MAC**)
  - Role Based Access Control (**RBAC**)

Controlled by whoever owns the object

Apply standard control to every subject (users)

Apply control based on what their job (role) is

MONASH U
Information Technology

LN3_Access Control 9

Access Control Types:

- Discretionary Access Control (DAC)
  - Controls access based on the identity of the requestor and on access rules on what requestors are allowed/not allowed to do.
    - termed *discretionary because an* entity might have access rights that permit the entity, by its own volition, to enable another entity to access some resource.
- Mandatory access control (MAC)
  - Controls access based on comparing security labels (sensitivity of resources) with security clearances (eligibility of entities to access certain resources).
- Role-based access control (RBAC)
  - Controls access based on the roles that users have within the system and on rules stating what accesses are allowed to users in given roles.

Discretionary Access Control

- Access to information is controlled by the owner of the object
- It can also provide for centralised or distributed security management
  - Centralised security — an administrator provides and controls access
  - Distributed security — managers or team leaders control access
- Commonly used Operating Systems (UNIX, Windows,…) implement this method of access control

Mandatory Access Control

- Imposes universal security conditions for all users, IT systems and information
- Commonly used in military systems
- Information is classified based on attributes such as sensitivity, secrecy and confidentiality
- A subject is said to have a security clearance of a given level;
- an object is said to have a security classification of a given level
- MAC also known as a Multilevel model
- Categorizes information by **sensitivity** and user access is based on their **responsibility** level
- Security levels
- A hierarchy of sensitivity attributes (ordering of levels)
- Typical military-style hierarchy
- An object’s sensitivity attribute is called *classification*
- Subjects have *clearances* to access objects in the hierarchy
- *Dominates*-relation: we say that x dominates y iff level(x) ≥ level (y)
- *Allows*: subjects x access to objects y accesstype à binary value

Role-Based Access Control

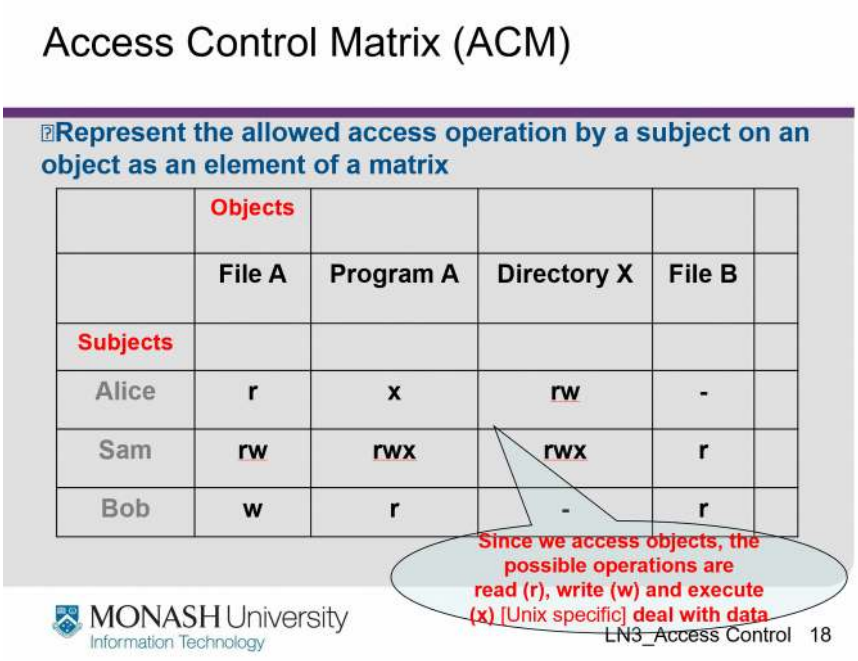
- Access to information and its resources is based on a user’s role (role = group of users, one policy for all users in group)
- It can cater for hierarchies and business constraints
  - Hierarchies define responsibilities and roles
  - Constraints provide the boundary of the job

Implementation of Access Control

- Who monitors that the access control requirements are not violated?
- Access control can also be implemented in different layers of technology.
- Application, services/middleware, operating systems or hardware.
- We will restrict ourself to the operating system level.

Implementation of Discretionary Access Control – Access Control Matrix

- Access control matrix (ACM)
- Defines the *subjects* (users), *objects* (information or resources) and *type of access*
- A combination of these three defines an authorisation rule (also known as access rule)



Access Control Matrix (ACM)

Represent the allowed access operation by a subject on an object as an element of a matrix

	Objects				
	File A	Program A	Directory X	File B	
Subjects					
Alice	r	x	rw	-	
Sam	rw	rwx	rwx	r	
Bob	w	r	-	r	

Since we access objects, the possible operations are read (r), write (w) and execute (x) [Unix specific] deal with data

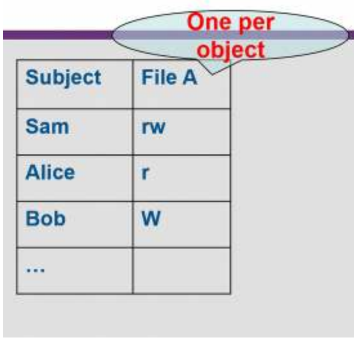
MONASH University Information Technology

LN3_Access Control 18

- In real systems, however, access control matrices are not very practical to implement because:
    - the matrix is usually sparse and there is a lot of wastage and redundancy
    - Although new subjects and/or objects can be added or removed easily, yet the centralized matrix could be a bottleneck for access enforcement
- Possible solutions:
    - Access Control List
    - Capabilities List

Access Control List

- Focus on the object.
- For each object specifies the subjects and their access operations.
- ACL is a column of the access control matrix
- The list is usually kept with the object.
- Simple to implement but difficult to manage.
- e.g. Removing a user.
- Each list can be of variable length

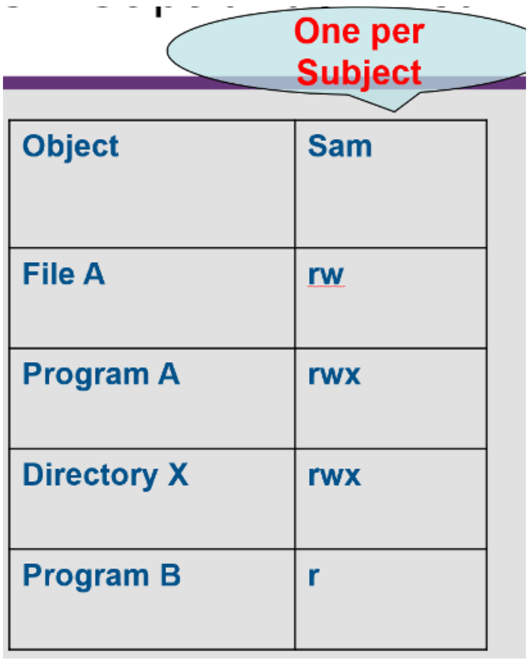


One per object

Subject	File A
Sam	rw
Alice	r
Bob	W
...	

Capabilities List

- Focus on the subject.
- For each subject specifies the objects and what kind of access operation that can be applied to the objects.
    - The rows of the access control matrix.
- Weakness:
    - It is difficult to determine the list of subjects with certain access right to an object.
- Often implemented with capability tickets
    - Ticket: a random number allowing access by subject to an object
    - Easy to delegate by passing ticket
- security risk if ticket is lost
    - Each list of variable length.



One per Subject

Object	Sam
File A	rw
Program A	rwx
Directory X	rwx
Program B	r

To simply the control structures

- Groups subjects into a role and assign access to a set of objects based on the role.
- A group is a collection of subjects with similar (or identical) access right.
- A role is a collection of access permissions that a user or a group of users have on a set of objects
    - defines the notion of context in which the access control is applied.
    - Easier to manage (one policy for all users with a given role, roles don't change as often as users)

Recap

- Access Control (access right) describes the way a subject may access an object.
- Access Control Matrix represents the allowed access operation by a subject on an object as an element of a matrix
- An Access Control List has one list per object: each list specified the users (subject) and their permitted access rights to the object.
- Capability List has one list per subject: each list specifies the objects the subject has access to
- DAC (Discretionary Access Control)
- The owner of an object decides on the access rights to the object.
- MAC (Mandatory Access Control)
- The administrator decides on the access rights to an object according to a certain policy.
- Most systems are based on DAC.
-