Topic 1: An Introduction

Give course unit structure and an introduction of security

- **Main textbook: Cryptography and Network Security,** 7th Edition by William Stallings.
- Supplementary textbook: Computer Security: Art and Science, 2nd Edition, Matt Bishop.
- Some of the slides/data here are from Cyber *Security Threats* slides by Dr Paul Twomey, the Lowy Institute for International Policy, Argo Pacific Pty Ltd.

Overview

- Part 1
  - Introduction to the Course Unit
- Part 2
  - What is Security
  - Security Problems and Challenges
- Part 3
  - Achieving Security
  - Security Models
  - Course Roadmap
  - Conclusion

Home Reading: Stalling's book, Chapter 1

Introduction to the Course Unit

- This course is about Cryptography and System Security
  - Cryptography:
    - important and commonly used cryptographic methods and techniques
  - Applied cryptography:
    - apply cryptography methods and techniques to solve security problems in a networked/distributed system setting
  - Access control
- Who should take this module
  - This is a technical module, so if you are interested in security and willing to learn some mathematical stuff, …
  - No prerequisite

Introduction to the Course Unit

- Course Unit Leader
  - Ning Zhang
  - ning.zhang@manchester.ac.uk

Introduction to the Course Unit

- Reading materials
  - **Main textbook: Cryptography and Network Security,** 7ed by William Stallings.
  - Computer Security: Art and Science, 2nd Edition, Matt Bishop.
  - Many other useful books; you can use the topics covered in the lecture handouts to scope your reading.
  - Lot of useful resource on the Internet, e.g. www.cert.org and www.nist.gov.
- All the teaching docs (except for the videos) are hosted in the Blackboard.

Course Unit Structure

- New materials are structured into topics
  - One topic per week; multiple parts per topic
  - One lecture handout per topic
- 5 sets of Quiz questions for formative assessment purpose
- Exercise questions for enhancing understanding of lecture contents and problem-solving skills
- Two-hour synchronous sessions per week via Blackboard Collaborate
  - For addressing any questions you may have on the lecture contents
  - For working on the exercise questions

Course Unit Structure

- Download and install CrypTool available at: http://www.cryptool.org/ (I recommend CrypTool 1.4.30 for Windows: https://www.cryptool.org/ct1download/SetupCrypTool_1_4_30_en.exe. There are also versions for MacOS and Linux). This is an e-learning tool for cryptographic algorithms. Other tools are fine, as long as they do the job.
-
- Assessment
  - 70% exam and 30% coursework

Part 2 Overview

- What is Security
- Security Problems and Challenges

What is Security?

History and Present

- Before the large-scale applications of the Internet
  - Interests in security were largely confined to the military domain
  - Other communities did not care much: the Internet was only a research network in its early stage
- Some milestones
  - Morris worm – 1988; Brought down a large fraction of the Internet
  - E-commerce, ATM/financial transactions – late 80s
  - Mosaic and Netscape – early 90s
  - Mobile Internet - Internet anywhere, anytime and by any devices
  - Cloud Computing - on-demand provisioning of computational and storage resources.
  - IoT (Internet of Things) – embedded devices, connected world, smart environment, …
  - Crypto currencies, smart contract signing, getting rid of third parties - blockchain technologies

Security Threats and Challenges

- Threats in a generic context (Confidentiality, Integrity and Availability, or CIA)
  - Disclosure (threats to confidentiality):
    - Snooping, sniffing (data in transit)
    - Unauthorised access (systems, data at rest)
  - Deception (fraud and forgeries):
    - Spoofing (identity theft)
    - Unauthorised data modification
    - Replay (intercept and retransmit)
    - Repudiation (false denial) of origin, repudiation of receipt
  - Disruption (threat to availability): modification, delay, Denial of Services (DoS)

Threats to integrity

Security Threats and Challenges

- Specific Threats (e.g. Top 5 Cyber Threats/Attacks)
  1. Social Engineering Attacks
     - What is our social media threat profile?
     - Who is monitoring it?
     - What tools are available for such monitoring?
     - What are our social media use policies? How do we implement them?
  2. Supply Chain Attacks
     - What sensitive information am I sharing with my vendors?
     - How do I assess the risk of each vendor?
     - What tools and services can I use to effectively control the threats posed by such a risk?

https://www.softwareone.com/en-gb/blog/articles/2019/01/24/biggest-cyber-security-challenges-in-2019

Security Threats and Challenges

3. IoT and Infrastructure Attacks
   - How are IoT and infrastructure devices impacting my risk?
   - Who is managing and controlling the risk?
   - What are the remediation protocols and policies that will help me to mitigate the risk?
4. Identity and Mobile Authentication
   - How should I control authentication and access across multiple devices, almost all connected to the Internet and with a varying degree of trust?
   - What kind of biometric and MFA (multi-factor-authentication) solutions are appropriate for my environment?
   - What cloud-based solutions should I use to access sensitive information?

Security Threats and Challenges

5. Rise of Zero-Day Threats and Polymorphic Attacks
   - What should I do if zero-day vulnerabilities are discovered for a mission-critical system?
   - Which security vendors and products should I trust for effective countermeasure to polymorphic attacks?
   - What is the status of my systems for known vulnerabilities? Who manages this?
   - Should I take a cyber-insurance?

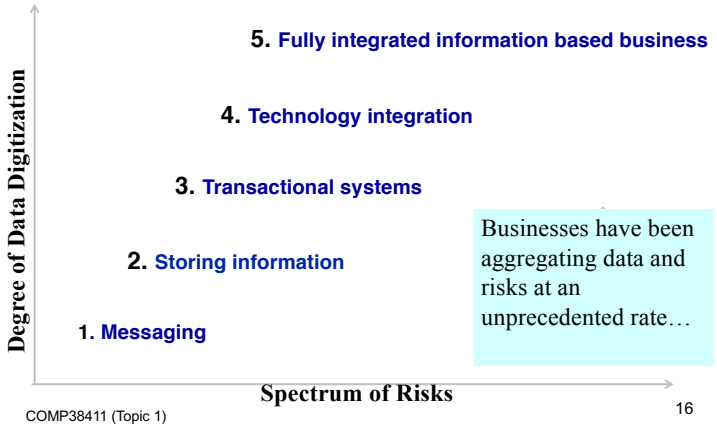
- Most attacks are done by using Malware (worms, viruses, Trojan, …)
- Hacking-as-a-Service

Security Threats and Challenges: Hacking as a Service

- 444,259 ransomware attacks took place worldwide in 2018.
- Hackers create 300,000 new pieces of malware daily.
- There is a hacker attack every 39 seconds.
- Hackers steal 75 records every second.
- You can purchase a consumer account for $1 on the dark market.
- More than 6,000 online criminal marketplaces sell ransomware products and services.
- Consulting services such as botnet setup ($350-$400).
- Infection/spreading services (~$100 per 1K installs).
- 73% of black hat hackers said traditional firewall and antivirus security is irrelevant or obsolete.

Source: Hostingtribunal.com

Security Threats and Challenges



Security Threats and Challenges

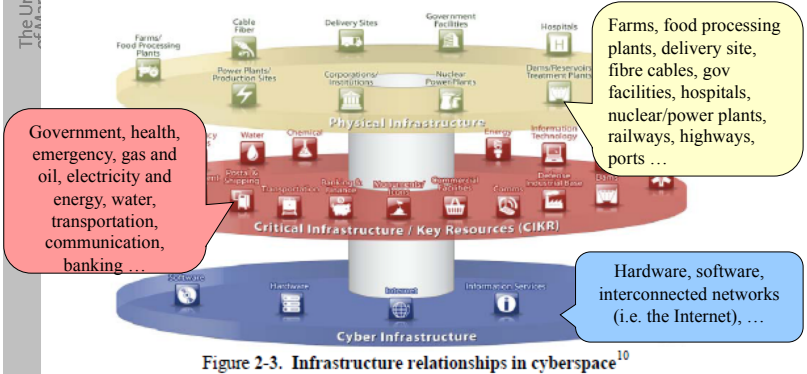


Figure 2-3. Infrastructure relationships in cyberspace[10]

Source: DHS, "Securing the Nation's Critical Cyber Infrastructure

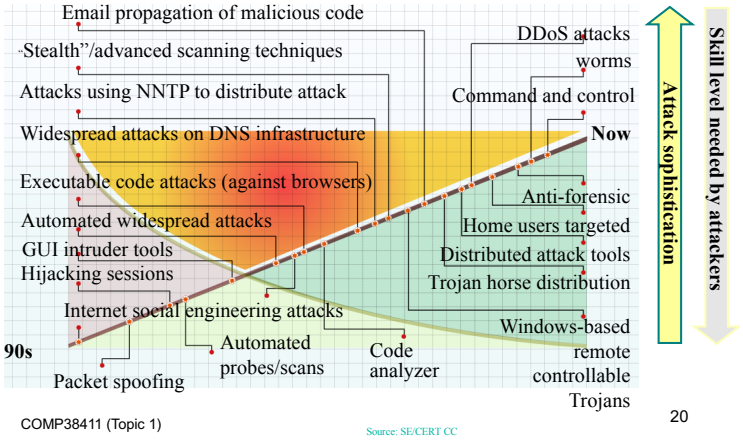
Threat Types	Motivation	Targets	Methods
Information Warfare	Military or political dominance	Critical infrastructure, political and military assets	Attack, corrupt, exploit, deny, conjoint with physical attack
Cyber Espionage	Gain of intellectual Property and Secrets	Governments, companies, individuals	Advanced Persistent Threats
Cyber Crime	Economic gain	Individuals, companies, governments	Fraud, ID theft, Extortion, Exploit
Cracking	Ego, personal enmity	Individuals, companies, governments	Attack, Exploit
Hacktivism	Political change	Governments, Companeis	Attack, defacing
Cyber Terror	Political change	Innocent victims, recruiting	Marketing, command and control, computer based violence

Source: analysis, Dr Irv Lachov

Security Threats and Challenges

- Naïve users - Lack of security awareness
- Inadequate management procedures
  - Insecure system set-up and configuration
  - Lack of proper policy making, implementation and enforcement procedures
- Global networks without national boundaries
- Heterogeneous devices, e.g. laptops, iPhones and PDAs, with universal connections
- Wireless and open channels
- Anonymous nature of many Internet-based services

Security Threats and Challenges



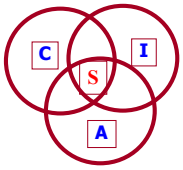
Source: SE/CERT CC

Part 3 Overview

- Achieving Security
- Security Models
- Course Roadmap
- Conclusion

Achieving Security – Basic security properties

- Securing information: CIA
  - Confidentiality
    - Keeping data and resources hidden
  - Integrity/Authenticity/Authentication (making sure data is authentic)
    - Content integrity (any unauthorised modification and replay of data can be detected)
    - Origin integrity (data is indeed from a claimed source)
  - Availability
    - Ensuring data/service is available to authorised users



Achieving Security – Advanced security properties

- Freshness
  - Ensuring data is not a replay/retransmission of 'old' data
- Non-repudiation
  - Protecting against repudiation (false denial)
- Fairness
  - Either all the parties have received what they expect to receive or none of them receives anything useful

Achieving Security – Life-cycle

- Define your security goal
  - Threats analysis and identification
    - Decide what to protect against
  - Policy/Requirement specification
    - Define what is, and is not, allowed
- Design and implementation: enforce policies (achieve security goal)
  - Decide how to protect in order to satisfy the specification
    - Technical measures
    - Procedural measures
- Operation and maintenance: security assurance
  - assess how well the implementation has achieved its security goal

Achieving Security – Threats analysis

- Identify assets, threats and vulnerabilities
- Assess the levels of risks on the assets based upon
  - Values of assets
  - Threats to assets and their importance
    - vulnerabilities and likelihood of exploitation
  - Not all threats are worth defeating (cost vs benefit)
- This may be carried out by using an Attack Tree

- Cost-benefit analysis
  - Is it cheaper to prevent (using security mechanisms) or recover (e.g. using restoration from backup) or just ignore?

Achieving Security – Threat analysis

- What is an **Attack Tree (Threat Tree)**
  - is a "conceptual diagrams showing how an asset, or target, might be attacked".
  - is consisted of one root node, children and leaf nodes.
- The root node representing the Attack Goal.
- Child nodes are conditions which must be satisfied to make the direct parent node true.
- Conditions may be 'OR' or 'AND': 'OR' represents alternative attack methods or avenues to succeed in the attack, whereas 'AND' represents multiple steps in launching an attack.

  - Reference: https://en.wikipedia.org/wiki/Attack_tree

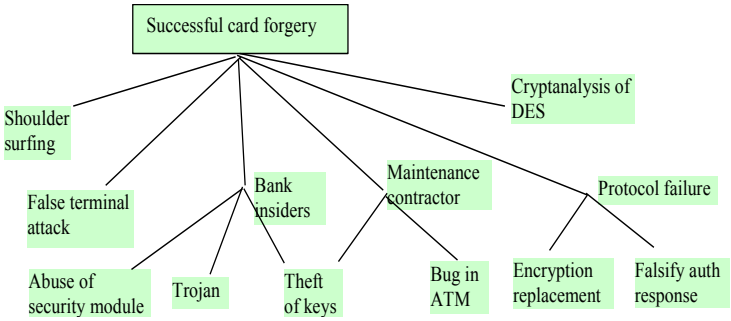
Achieving Security – Threat analysis

- Each node may be given a value to indicate, e.g.
  - **likelihood** that an attacker will mount the attack, or **probability** of succeeding the attack
  - **cost** in succeeding the attack, in terms of monetary cost, or time taken to accomplish the attack, etc.

- How to produce an Attack Tree
  - Identify an attack goal
  - Identify all the possible attack methods or avenues to achieve the goal

Achieving Security – Threat analysis

- An example of threat analysis using a **Threat Tree:**



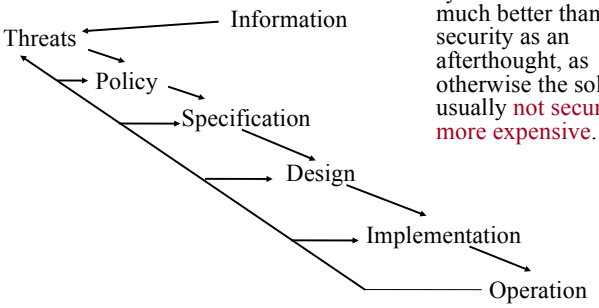
Achieving Security – Defining & achieving security goal

- Security measures: a method, protocol, tool, or procedure used to address the risks identified (or to enforce a security policy)
  - Prevention
    - Block attacks by closing vulnerabilities
    - Reduce the level of risks by making attack harder
    - Make another target more attractive than this target
    - E.g. access control (firewalls), encryption, digital signatures, honey pots …
  - Detection
    - Measures taken during or after the attacks
    - E.g. logging, auditing and intrusion detection
  - Recovery
    - Assess and repair damage
    - Continue to function correctly even if attack succeeds
  - Accept it and do nothing

Achieving Security - Operation and maintenance

- Assurance
  - Testing to check the correct implementation of policies.
  - Formal evaluation of the implementation.
  - Standards
    - US Security Evaluation Criteria (the Orange Book).
    - European ITSEC (Information Technology Security Evaluation Criteria).
- Human Issues
  - Organizational issues
    - Power and responsibility
    - Financial benefits
  - People problems
    - Outsiders and insiders
    - Social engineering

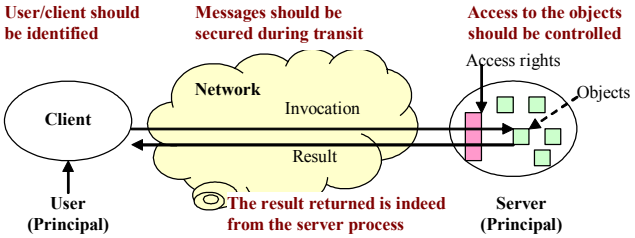
Tying it all together



Designing security into a system from the start is much better than adding security as an afterthought, as otherwise the solution is usually not secure and more expensive.

Security Models: A distributed system security model

- What are the security threats in this model?
- What are the security properties/services that are necessary to counter the threats?

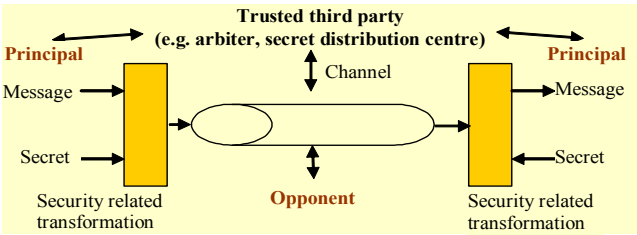


Security Models: a distributed system security model

- In this model, following issues arise:
  - Could the server be certain about the identity of the principal behind the invocation?
  - Could the client be certain about the invocation response message
    - Is it from the intended server?
    - Has it been altered during transit?
  - The channel should be secured
    - A perpetrator on the network could read, copy, alter, or inject messages as they travel across the network and gateways.
    - A perpetrator may attempt to save copies of messages and to replay them at a later time.
  - etc …

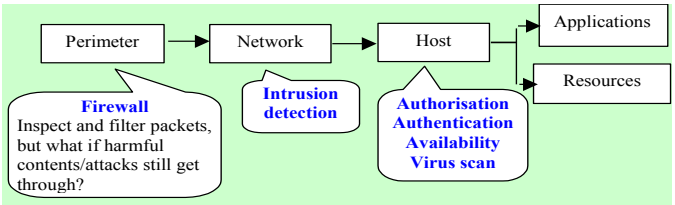
Security Models: Communication security model

- Here the emphasis is on protecting **data over the channel**.
- Security questions: authenticity (*prove the origin of a message + its integrity*) and confidentiality.



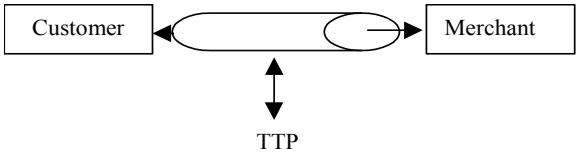
Security Models: Network security model

- Here the focus is on protecting data and services on a network against external attacks or unauthorised usage.
- Multi-level security measures.
- However, the use of mobile devices will make the boundary hard to define.



Security Models: E-commerce security model

- The opponent now is a misbehaving insider.
- The third party is now a trusted third party (TTP), e.g. an arbitrator, that offers some services.
- Non-repudiation service generates evidence for dispute resolution.



Course Roadmap

- **Security basics and fundamentals**
  - T2 – Classical Ciphers
  - T3 – Symmetric-key Ciphers
  - T4 - Asymmetric-key Ciphers
  - T5 – Cryptographic Checksum
  - T6 - Digital Signatures
  - T7 - Public Key Infrastructure
  - T8 – Key Management Issues
- **Security systems**
  - T9 – Entity Authentication
  - T10 – More Security Solutions
  - T11 – Access Control

Exercise Question – E1.1

Comment on the implications to risks (i.e. whether risks are increased or decreased) in terms of Confidentiality, Integrity and Availability in each of the following cases:

i. Disconnect a computer from the Internet;
ii. Have extensive data checks by different people/systems.

Exercise Question – E1.2

i) In this exercise, you are asked to identify, via literature research, potential cyber attack threats to *mobile* banking (i.e. perform banking transactions using your mobile phone). You are expected to be able to explain the attacking mechanism of each of your identified threats (i.e. how the attack is performed) and try to name any countermeasures to your identified threats.

ii) Draw a threat tree for 'Read your mate's email'.

Conclusions

- Networks and distributed systems are part of our daily lives.
- Most systems that surround us are networked via the Internet which is open to many attacks and threats.
- Security provisioning in such an environment is a complex task.
  - It encompasses issues of computer security, software security, wired network security, wireless network security, and processes/procedures (people)!
- People are often the weakest link in security.
- This course can only give you a flavour of these many interesting and exciting problems – security issues, threats and mechanisms (services and protocols) in a distributed environment.

Topic 2: Introduction to Cryptography

Introduce the basic concepts of cryptography, some classical techniques and cryptoanalysis attacks

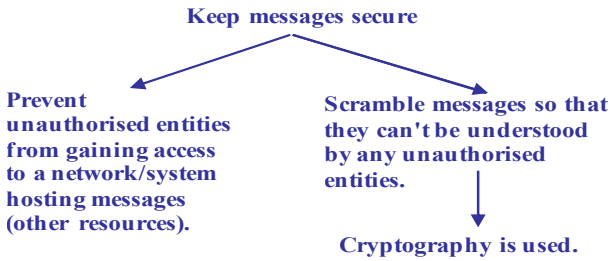
Source: Stalling's book, chapter 3

Overview

- Part 1
  - What is cryptography and its applications
  - Terminology and definitions
- Part 2
  - Caesar cipher
- Part 3
  - More classical encryption techniques
  - More on cryptographic attacks
  - Conclusion

What is Cryptography?

- **Cryptography** is "the art of keeping messages secure" by Schneier.



Application of Cryptography

- Confidentiality (secrecy, privacy) of data in transmission & in storage
- Integrity of data (data authentication/authenticity) in transit & in storage
- Authentication of an identity (entity authentication)
- Credential systems (a proof of qualification or competence of a person)
- Digital signatures
- Electronic money (e.g. cryptocurrency, bit coins)

Application of Cryptography

- Threshold cryptosystems (a decryption key, or a signature signing key, is shared among a group of entities and a subset of these entities (more than some threshold number) have to collaborate to perform the decryption or signature signing).
- Secure multi-party computations (e.g. multiple parties compute a function jointly, the input is from the multiple parties, but no party should learn anything rather than its own input and the final result of the computation)
- Digital right management (e.g. activation of a software license by authorized users)
- Electronic voting
- …

Achieving Confidentiality using Encryption

- Using encryption to achieve secure communication over an insecure channel
- Ciphers (cryptosystems)
  - Symmetric-key based (conventional ciphers)
    - Same key is used for encryption and decryption
    - Historical ciphers
    - Modern ciphers
  - Asymmetric-key based (public-key ciphers)
    - Different keys are used for encryption and decryption
    - Modern ciphers

Terminology

- Cryptography: practice and theory of concealing text.
- Plaintext or cleartext: a message in its original form.
- Ciphertext: a message in an encrypted form.
- Encryption: code a message to hide its meaning using an encryption key.
- Decryption: convert an encrypted message back to its original form using a decryption key.
- Other terms: encode and encipher for encryption, and decode and decipher for decryption.
- Cipher/Cryptosystem: the system that performs encryption and decryption.
- Cryptanalysis: attempts to discover plaintext or key.

Security Definitions (in the context of cryptographic based solutions)

- Unconditionally secure
  - The system cannot be defeated, no matter how much power is available by the adversary.
- Conditionally secure based on computational complexity
  - The perceived level of computation required to defeat the system, which exceeds the computational resources of the hypothesized adversary.
  - e.g. given the computing power, it takes very long time to break a ciphertext.

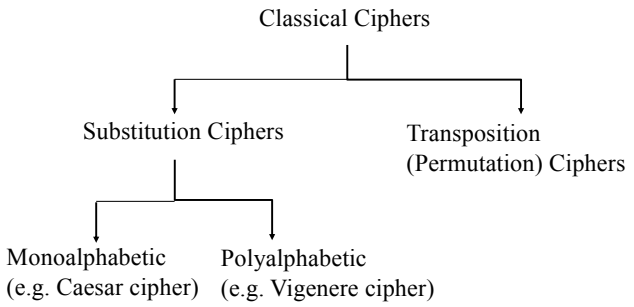
Security Definitions (in the context of cryptographic based solutions)

- Provably secure
  - Some solutions can be verified to be secure, e.g. in the case of security protocol designs.
- Ad hoc security
  - Claims of security generally remain questionable.
  - Unforeseen attacks remain a threat.

Part 2 Overview

- Caesar cipher

Classical Encryption Techniques



Classical Encryption Techniques

- Classical (historical) algorithms are based on substitution & permutation.
- Substitution -> Confusion
  - E.g. 'a' becomes 'b'
- Transposition/Permutation -> Diffusion
  - E.g. 'abcd' becomes 'dacb'
- XOR operator
- Simple/non-secure ciphers
  - Shift Cipher – Caesar Cipher,
  - Vigenere Cipher, …
- Secure cipher
  - One-Time Pad

•Modern ciphers use substitution technique: take in N bits and output a different set of N bits using a lookup table, called S-Boxes.

•Modern ciphers use transposition technique: they permute N bits using a lookup table, called P-Boxes.

- substitution: one letter is exchanged for another
- transposition: the order of the letters is rearranged

Classical Encryption Techniques - Caesar cipher (Shift cipher)

- It uses simple substitution
- Encryption operation: each plaintext letter is translated to the letter a fixed number of letters further down the alphabet table (circular right shift). 将每个明文字母翻译成字母表上再往下固定数量的字母（循环右移）。
- Decryption operation is the reverse of the encryption operation.
- The operation can be expressed using addition modulo 26.
  - The message must be a sequence of letters, each letter is identified with a number.
  - The key k is a number in the range 1, …, 25.
  - Encryption/decryption involve $\pm$ k to each letter (mod 26).

mod n

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

 ci = E(pi) = (pi+3) mod 26 Caesar used a shift of 3

Classical Encryption Techniques – Attacks on Caesar cipher

- Brute-force attack (or exhaustive key search) is by trying all possible keys.
- The three characteristics which make brute-force attack practical:
  - The encryption and decryption algorithms are in public domain.
  - There are only 25 keys to try.
  - The language of the plaintext is easily recognisable (compressed text not).
- Given a small number of plaintext-ciphertext pairs encrypted under a key K, K can be recovered by exhaustive key search with 2^{n-1} processing complexity (where n is the bit-length of the key).
- If the plaintexts are known to contain redundancy, then ciphertext-only exhaustive key search is possible with a relatively small number of ciphertexts.
- With today's computing power, (symmetric) key length should be at least 128 bits.
- Also vulnerable to another form of attack - frequency analysis (also known as counting letters) attack.

Classical Encryption Techniques – Attacks on Caesar cipher

- Also vulnerable to frequency analysis (also known as counting letter) attack.
- Frequency analysis attack is a known-ciphertext attack based on the study of the frequency of letters or groups of letters in a ciphertext.

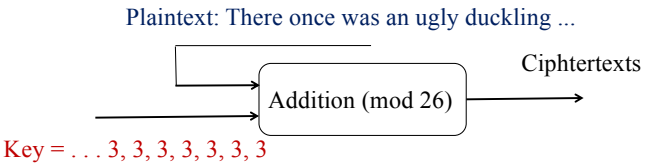
Part 3 Overview

- More classical encryption techniques
  - Vigenere cipher
  - One-time pad and stream ciphers
  - Transposition cipher
- More on cryptographic attacks
- Conclusion

Let us start with a quick question on Caesar cipher

(a) This is a diagram illustrating Caesar Cipher encryption operation. Could you propose a (simple) solution to hide letter frequency distributions in plaintexts, so that, from ciphertexts, the frequency distributions in plaintexts are not so obvious.

(b) How to choose the key stream to make the ciphertext the hardest to break?



take two input: plaintext + key
one-to-one mapping, every plaintext letter is always replaced by a fixed cipher text

Letter Frequency Distribution in English (Literature)

- **This is in percentage term** (this may vary depending on the content/size of the text)

a	b	c	d	e	f	g	h	i
8.2	1.5	2.8	4.2	12.7	2.2	2.0	6.1	7.0

j	k	l	m	n	o	p	q	r
0.1	0.8	4.0	2.4	6.7	7.5	1.9	0.1	6.0

s	t	u	v	w	x	y	z
6.3	9.0	2.8	1.0	2.4	2.0	0.1	0.1

a, e, i, t are used more often than other letters
this will make a successful attack easier more efficient than using the brute force attack
so we should hide it, flattern the frequency

vigenere has a 26*26 martix
plaintext letter represents the column竖着的
and the key represents the row横着的, so when look up the matrix, we can get the corresponding letter

Vigenere Cipher

还是前面那个a-0，b-1的表得到的bed等于143

ob3: the longer the keyword, the better in terms of hiding the plaintext the letter distribution

- This cipher uses a keyword. For example, let's say the keyword (K) is 'bed', i.e. 143 (as b → 1, e → 4, d → 3). The plaintext (P) (*There once was an ugly duckling ...*), and the corresponding ciphertext (C) are given below:

ob2: help to flattern the frequency of letters

P:	T	h	e	r	e	o	n	c	e	w	a	s	a	n	u	g	l	y	d	u	c	k	l	i	n	g
K:	1	4	3	1	4	3	1	4	3	1	4	3	1	4	3	1	4	3	1	4	3	1	4	3	1	4
C	u	l	h	s	i	r	o	i	h	x	e	v	b	r	x	h	p	b	e	y	f	l	p	l	0	k

repeat itself bcz the keyword is shorter than the plaintext

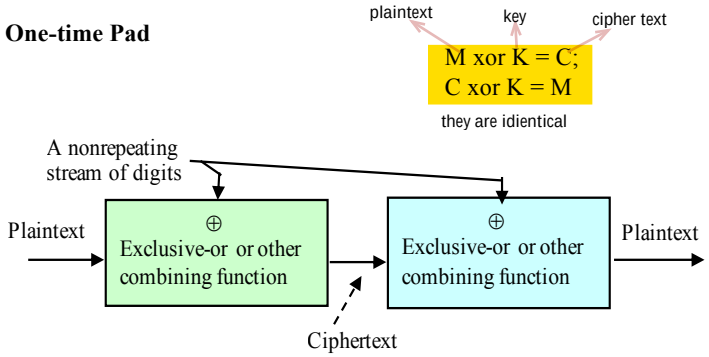
- Here, each plaintext letter has been shifted by a different amount that is determined by the key.
- This cipher is significantly more secure than a regular Caesar Cipher. Its security level is dependent on the keyword length.

One-time Pad

- If the keyword length is as long as the plaintext and is random, then we have a cipher with perfect secrecy.
- **One-time pad** is such a cipher.
  - **it uses an one-time random key that is as long as the plaintext with no repetitions (only used once).** operate on bit
- If used properly, it is provably unbreakable. (Shannon, 1949)
- It was proposed by Gilbert Vernam during World War I.
- It is a special variant of the stream cipher.
  - Typically a stream cipher uses (mod 2) (exclusive-or, i.e. XOR) function.

secure but not pritical

One-time Pad



Stream Ciphers

- Stream ciphers encrypt individual bit or character streams.
- When encrypting individual bit streams, XOR is used, i.e.
  ciphertext (C) = plaintext (M) xor keystream (KS)

  $M = m_1\ m_2\ m_3\ \ldots\ m_i\ \ldots$

  $KS = k_1\ k_2\ k_3\ \ldots\ k_i\ \ldots$

  $C = c_1\ c_2\ c_3\ \ldots\ c_i\ \ldots$

  where $c_i = m_i$ xor k_i, and m_i is typically a byte (8 bits) or 1 bit.
- Replace the random key in One-time Pad by a pseudo-random sequence, generated by a cryptographic pseudo-random generator that is 'seeded' with the key.

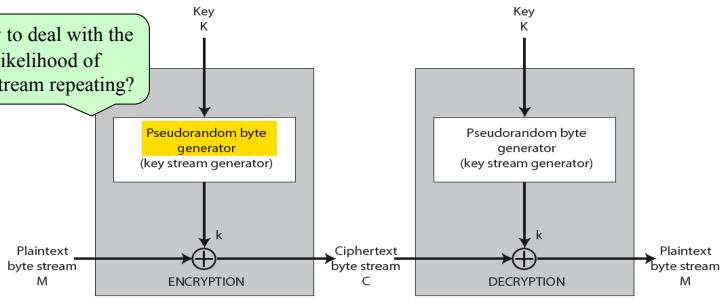
Stream Ciphers

使用初始化生成器的密钥生成一个密钥流。

Generate a keystream using a key that initializes the generator.

rather use random

How to deal with the likelihood of keystream repeating?



Transposition Cipher

- The ciphertext is generated by performing permutation on the plaintext (i.e. changing the order of the alphabets in the plaintext).
- An example:

key	4	3	1	2	5	6	7
plaintext	a	t	t	a	c	k	p
	o	s	t	p	o	n	e
	d	u	n	t	i	l	t
	w	o	a	m	x	y	z

ciphertext ttnaaptmtsuoaodwcoixknlypetz

  - Write the plaintext in a rectangle, row by row, and read the message off, column by column, but permuting the order of the columns, where Key = order of the columns to read.

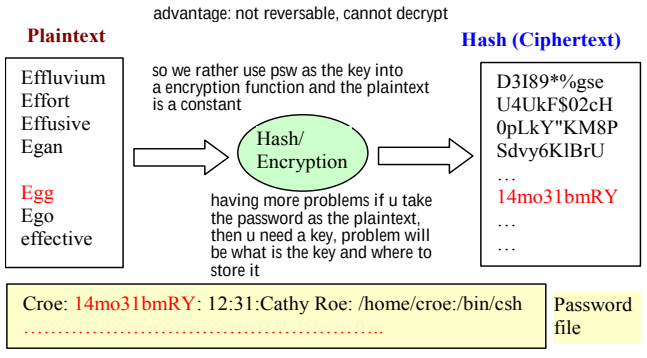
Cryptographic Attacks

- The security of any (modern) cipher is based ***not*** on the secrecy of an algorithm, but on the security of the cryptographic keys!
- Common types of attacks
  - Try to break or 'crack' the algorithm by exploring any flaws (缺陷) in the algorithm (frequency analysis).
  - Assume attackers can recognize a plaintext, try to decrypt a ciphertext with every possible key until a recognised plaintext is obtained (brute force attack or exhaustive key search attack).
  - Run the algorithm on massive amount of (probable) plaintexts until a plaintext that encrypts to the ciphertext he is analysing is found (dictionary attack).

盗力攻击
穷尽密钥搜索攻击
have to make the key long enough

在大量的（可能的）明文上运行算法，直到找到一个与他所分析的密文加密的明文 - 字典攻击

What is a dictionary attack? - an example



More on Cryptographic Attacks

- Ciphertext-only attack (e.g. frequency analysis)
  - Attacker knows ciphertexts of several messages encrypted with same key, plaintext is recognizable;
  - Goal: to find plaintext, possibly key
- Known-plaintext attack (e.g. dictionary attack)
  - Attacker observes <plaintext, ciphertext> pairs encrypted with same key;
  - goal: to find key

More on Cryptographic Attacks

- Chosen-plaintext attack
  - Attacker can choose the plaintext and look at the paired ciphertext;
  - goal: to find the key
- Cryptographic attacks often exploit the redundancy of natural language
  - Lossless compression before encryption removes redundancy

Exercise Question – E2.1

- Given the Letter Frequency Distribution in English, as shown in the next slide, and the following ciphertext which has been generated using the Caesar cipher (but a different key), use the frequency analysis method to work out the encryption key and the corresponding plaintext.

Ciphertext:
bpmzm wvkm eia iv cotg lckstqvo eqbp nmibpmza itt abcjjg ivl jzwev …

Key: ??
Plaintext: ??

Exercise Question – E2.1 continue

- **Letter Frequency Distribution in English (in percentage)**
  (this may vary depending on the content/size of the text)

E	T	A	O	I	N	S	H	R	D	L	U	C	M	W	F	Y	G	P	B	V	K	X	J	Q	Z

a	b	c	d	e	f	g	h	i
8.2	1.5	2.8	4.2	12.7	2.2	2.0	6.1	7.0

j	k	l	m	n	o	p	q	r
0.1	0.8	4.0	2.4	6.7	7.5	1.9	0.1	6.0

s	t	u	v	w	x	y	z
6.3	9.0	2.8	1.0	2.4	2.0	0.1	0.1

Exercise Question – E2.2

The stream cipher diagram given earlier shows that a key stream used for encryption/decryption is generated by using a pseudo-random generator that is 'seeded' with a (shorter) key, K. This key, K, is usually called encryption/decryption key.

(i) Comment on the benefit(s) of this approach, i.e. why is the key stream generated from K?
(ii) How to ensure (or to minimize the chances) that the output of the pseudo-random generator (i.e. the key stream) is non-repeating?

This approach addresses the following issue:
• The key stream must be unique for each encryption, i.e. a key stream must not be used twice, as, otherwise, the encryption will not be secure:
• K = M xor C => M ' = K xor C ' = (M xor C) xor C '

• This is a dangerous property and we must never ever reuse the same keystream to encrypt two different messages.
• To ensure a key stream non-repeating can be challenging:
(a) their distributions are expensive – a key stream should be as long as the message to be protected and this is too expensive for long messages;
(b) managing and storing a large number of key streams may also be problematic;
(c) there is an synchronisation issue too – the key stream used by a sender/receiver pair for a particular message must be the same.

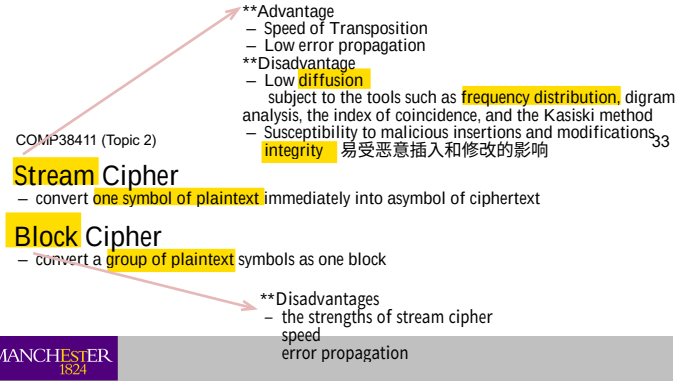
(a) Use a strong mixing function as the pseudo random generator.
(b) add another input into the function, a counter, which changes (e.g. increment by 1) for each iteration.
(c) If the counter value reaches its maximum, then change the key, K.

Familiarise with CrypTool

- Download and install CrypTool 1.4.30 from http://www.cryptool.org/index.php/en/download-topmenu-63.html (or use another on-line cryptotool).
- This is a cryptographic e-learning software; it has a number of features which can make your learning interesting:
  - It is a freeware program with graphical user interface.
  - It visualises a number of algorithms.
  - It contains nearly all state-of-the-art cryptography functions.
  - It can be used to analyse cryptographic methods …
- Play with the CryptTool and learn its capabilities.

Conclusions

- Explained a number of historical ciphers.
- Explained how some of the ciphers may be attacked.
- Introduced the concepts of substitution and transposition (permutation) as basic cipher operations for classical cryptosystems.
- Introduced some cryptographic attacks
- A secure cryptosystem must withstand all sorts of attacks.



Topic 3: Symmetrical Cryptography

Understand the principles of modern symmetric (conventional) cryptography

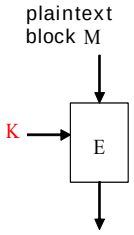
Source: chapters 4, 6 and 7 of Cryptography and Network Security

Overview

- Part 1
  - Block Ciphers Overview
- Part 2
  - Data Encryption Standard (DES) and 3DES
- Part 3
  - Advanced Encryption Standard (AES)
- Part 4
  - Use of Block Ciphers in Real World – Modes of Encryptions
  - Block Ciphers vs Stream Ciphers
  - Conclusion

Some of the slides are based on Lawrie Brown's slides supplied with William Stalling's book "Cryptography and Network Security: Principles and Practice," 7th Ed, 2017.

Block Cipher



- Plaintext is divided into blocks of fixed length and blocks are encrypted one at a time.
- In addition to **a key generation function**, a block cipher has two functions, **an encryption function**, $E_K(.)$, and **a decryption function**, $D_K(\cdot)$, such that

$$C = E_K(M) \text{ (or } C = E(K, M))$$
$$M = D_K(C) \text{ (or } M = D(K, C))$$

where
  - M is a plaintext block and C is a ciphertext block
  - K is a secret (a symmetric or a private key)

Block Cipher Design Criteria

- Completeness
  - Each bit of the output should depend on every bit of the input and every bit of the key.
    输出的每一个位子都应该取决于输入的每一个位子
- Avalanche effect (Diffusion) 雪崩效应（扩散）
  - Changing one bit in the message input should change many bits in the output. 改变信息输入中的一个位应该改变输出中的许多位
  - Also, changing one bit in the key should result in the change of many bits in the output.
- Statistical independence (Confusion)
  - Input and output should appear to be statistically independent.
    统计独立性（混淆）
    输入和输出在统计上应该是独立的。

Block Cipher Design

- Claude Shannon identified that confusion and diffusion are two properties of the operation of a secure cipher and these properties can be achieve by using substitution and permutation.
- Horst Feistel provided an implementation of this idea - Feistel block cipher structure (Feistel block cipher, Feistel network).
  - A complex encryption function can be built out of some simple operations (round function) by repeatedly using them.
- Examples of simple operations
  - substitutions 替换
  - permutations 个排列组合
  - XOR XOR
  - modular multiplication 模块化乘法
- Ciphers that use substitution and permutation are called substitution-permutation (S-P) networks.

Feistel Block Cipher

- Structure overview
  - Initial permutation (IP) of bits.
  - Split into two halves: a left half and a right half.
  - 16 rounds of identical operations, but each round uses a different subkey (round key).
  - Inverse initial permutation (IP-inverse).

num of round每次可能是不同的
相同操作，不同的子键subkey/round key

Use a simple example to illustrate:
IP: 2-6-3-1-4-8-5-7

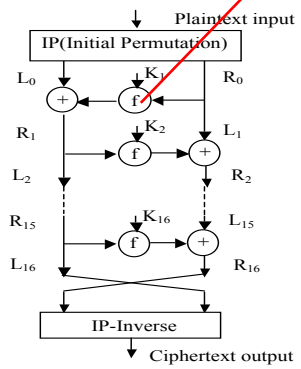
IP-inverse: 4-1-3-5-7-2-8-6

IP Input:
X1, X2, X3, X4, X5, X6, X7, X8
IP Output (IP-inverse Input):
X2, X6, X3, X1, X4, X8, X5, X7
IP-inverse Output:
X1, X2, X3, X4, X5, X6, X7, X8

xor：如果a、b两个值不相同，则异或结果为1。
如果a、b两个值相同，异或结果为0。

Feistel Block Cipher

complex function, don't worry about what is inside



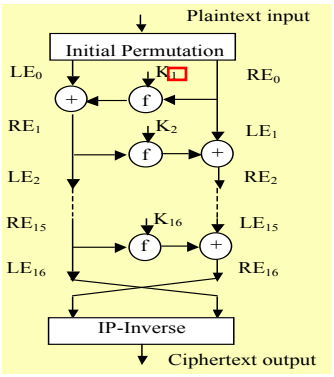
Encryption:
r rounds (for DES, r=16)
Plaintext = (L_0, R_0)

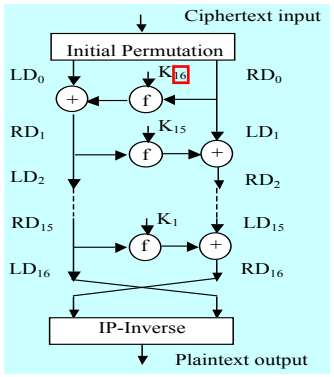
For $1 <= i <= r$
$L_i = R_{i-1}$ k is the round key
$R_i = L_{i-1}$ **xor** $f(R_{i-1}, K_i)$
Subkeys K_i is derived from key K
Ciphertext = (R_r, L_r)

Decryption:
Is the same as the encryption process except that the subkeys are applied in a reverse order.

Feistel Block Cipher

the only difference is the K16 uses the first at decryption





Block Cipher Design - Feistel Block Cipher

- Round function *f*:
  - Typically use permutations, substitutions, modular arithmetic.
  - Takes a *n*-bit block and outputs a *n*-bit block.
  - Each use of the round function employs a different subkey derived from *K*.
- Block size, *n*
  - larger block sizes mean greater security but make encryption/decryption slower; typically *n* is 128-bit or 256-bits.
- Key size, *s*
  - larger key size means greater security but reduced speed; a 128-bit size has become a norm. semmetric
- Number of rounds, *r* (typically 10+ rounds).

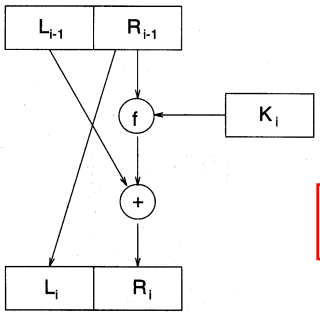
Part 2 Overview

- Data Encryption Standard (DES) and 3DES

DES (Data Encryption Standard)

- First published in 1977 as a US Federal standard.
- DES is a de facto international standard for banking security.
- DES is **a Feistel block cipher**
  - Block length is 64 bits,
  - key *K* is 56 bits; actually 8 bytes, but the 8th bit in each byte is a parity-check bit. 奇偶校验位
- The subkeys $k_1, k_2 ..., k_{16}$ are each 48-bits, generated from key *K*.
- The DES decryption algorithm is the same as the encryption one; the only difference is that the keys for each round must be used in the reverse order, i.e. k_{16} first and k_1 last.

DES – Architecture of Each Round

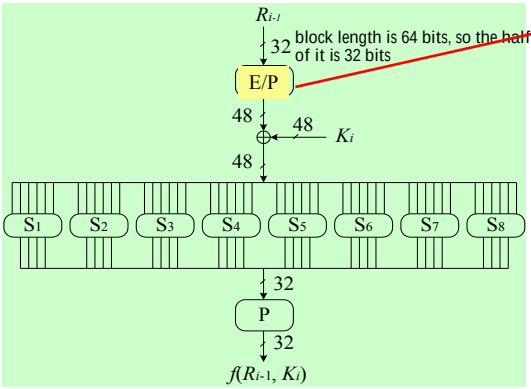


$L_i = R_{i-1}$
$R_i = L_{i-1}$ **xor** $f(R_{i-1}, K_i)$

DES – Round Function, f

- Step 1 - Expansion Permutation: 扩容换位
  - Right half (32 bits) is expanded (and permuted) to 48 bits.
  - Diffusing relationship of input bits to output bits.
- Step 2 - Use of Round Key:
  - 48 bits are XOR-ed with the round key (48 bits).
- Step 3 - Splitting:
  - Result is split into **eight** lots of six bits each. 结果分成八批，每批六位。
- Step 4 - S-Box: (S = Substitution)
  - Each six bit lot is used as an index to an S-box to produce a four-bit result. 每一个六位的批次被用作S箱的索引，以产生一个四位的结果。
- Step 5 - P-Box: (P = Permutation)
  - 32 bits output from 8 S-Boxes are permuted = the output of *f*.

DES – Round Function, f the f function only influences the right half



DES - A simple example of E/P

E/P (Expansion Permutation)							
4	1	2	3	2	3	4	1

all the bits appear twice

- E/P input = 1101
- E/P output = 11101011

DES

- S-Box operation
  - Each of the 8 different S-boxes is a table of 4 rows and 16 columns.
  - The 6 input bits specify which row and column, i.e. a cell, to use.
    - Bits 1 and 6 select the row.
    - Bits 2-5 select the column.
  - The decimal value in the cell is then converted into a 4-bit result, which is the output from the S-box.
- Efficient to encrypt/decrypt, so mainly used for encryption of **message contents - confidentiality**.

- The algorithm public, but the design principles are kept secret. Built-in trapdoors might be placed in secret boxes.

DES - S-Box Operation

- An example using S-Box 1 (S1)

Input to S1 = b1-b2-b3-b4-b5-b6 = 100011

b1-b6 = 11 = row 3
b2-b3-b4-b5 = 0001 = column 1

it has 4 rows, and each row can be indexed by two bits, 2 by power 2 is 4,
each column can be indexed by 4 bits

Output = 12 = 1100

S1	x0000x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0yyyy0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
0000 1	00	15	07	04	14	02	13	10	10	06	12	11	09	05	03	08
0001 2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
0010 3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13
0011																

range from 0-15

DES Strength

the key size is too small

- Its weakness is 56-bit key - which is good enough to deter casual DES key browsing, but not for a dedicated adversary.
- Use of a 56-bit key - can be broken on average in 2^{55} (i.e. $3.6 * 10^{16}$) trials.

give us 2 power of 56 options key space

	trials/second	time required
how powerful ur computer is to perform the attack	1	10^9 years
	10^3	10^6 years
	10^6	10^3 years
	10^9	1 year
	10^{12}	10 hours

  - a DES chip does 1 million encryptions per second.
  - a million chips in parallel do 10^{12} trials per second.
- For today's computing power, key size should be at least 128 bits.
- Improvements: Triple DES (3DES), AES (Rijndael)

two options: 1. nested encryption
2. a brand new standard

Triple DES

is nested encryption
so it is slow in performance

- Involves use of two or three DES keys.
- EDE2 (triple DES using two keys)
  - EDE2 uses two DES keys (K_1, K_2), encryption algorithm E, and decryption algorithm D, *i.e.* $C=E_{K1}(D_{K2}(E_{K1}(M)))$

k1 is used twice

  - So the key length is 112-bits.
  - The use of D here does not have any security implication; it just makes triple-DES backward compatible with single DES if $K_1=K_2$.

使三重DES向后兼容单重DES

- EDE3 (triple DES using three keys)
  - Liked by some; EDE3 uses three keys, $C=E_{K3}(D_{K2}(E_{K1}(M)))$; the key length is 168 bits.

BUT due to the meet-in-the-middle attack, the effective key lengths for both cases are much shorter.

由于中间相遇攻击，这两种情况的有效密钥长度都要短得多。

Meet-in-the-Middle Attack

users will collect records of encryption and want to find a match, if he has a huge record it allow him to successfully mount an attack with a much shorter time

- Time-memory tradeoff
- Let us use double DES to explain this attack
- Principle
  - build a table of keys
  - Compute f(k, M) for every possible key
    - f is an encryption function, M is a known message
  - Eavesdrop a value f(k', M)
  - If f(k', M)=f(k, M), then there is a good chance k'=k.

窃听

Meet-in-the-Middle Attack

- An attack example
  - Assume:
    - a new encryption function: F(k1, k2, M) = f(k1, f(k2, M))
    - A pair (M, C) is known
  - Attacker:
    - Encrypt M, i.e., computing f(k2, M), for all possible values of k2; store the values in a table
    - Decrypt C, i.e., computing f^{-1}(k1, C), for all possible values of k1, and for each result check the table
    - A match reveals a possible combination of the two keys

Part 3 Overview

- Advanced Encryption Standard (AES)

AES - Background

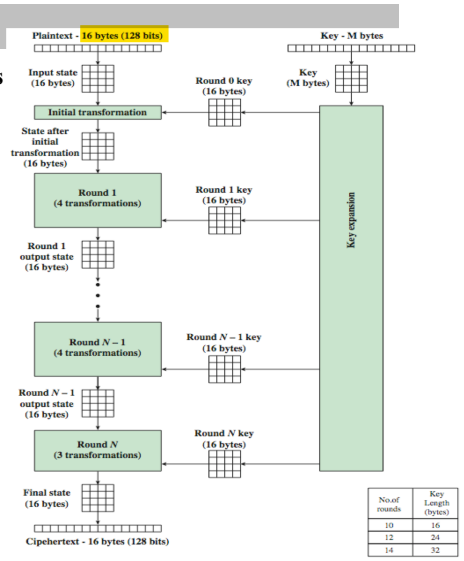
- US NIST issued call for algorithms to replace DES in 1997.
  - stronger & faster than 3DES
  - active life of 20-30 years (+ archival use)
  - provide full specification & design details
  - both C & Java implementations
- 15 candidates accepted in 98
- 5 were shortlisted in 99
- Rijndael was selected as the AES in 2000, and formally nominated as the Advanced Encryption Standard (AES) in 2001.
- Designers
  - Vincent Rijmen, Joan Daemen → Rijndael.
- Website: http://www.nist.gov/aes/

AES – Overview

- Like DES, AES is a symmetric block cipher.
  - The same key is used to encrypt and decrypt the message.
  - The plaintext and the ciphertext have the same size.
- Different from DES, it is an **iterative** rather than **feistel** cipher.
- Block size is 128 bits (others are allowed but not recognised by the standard).
- The key lengths are 128, 192, or 256 bits, i.e. the standard comprises **three** block ciphers**, AES-128, AES-192** and **AES-256.**
- It is a **substitution-permutation** cipher involving *r* rounds:
  - for key length=128 bits, r=10;
  - for key length =192 bits, r=12; and
  - for key length =256 bits, r=14.

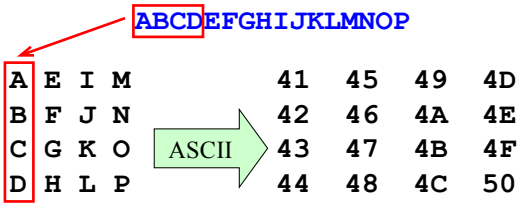
AES – Encryption Process

- Operate on bytes – efficient in software implementation



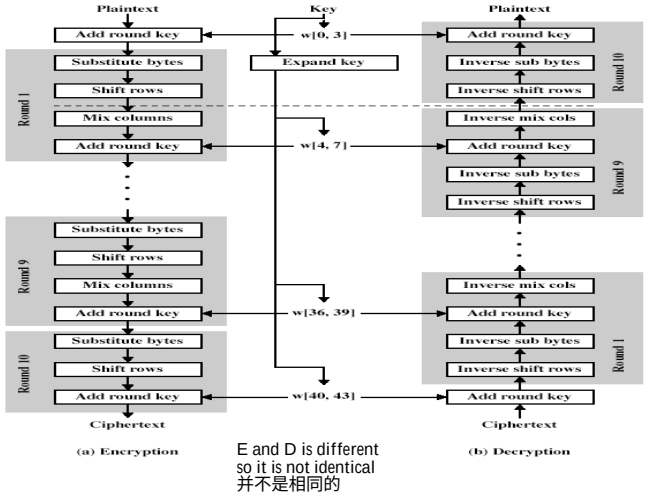
AES – State

- AES has a fixed block size of 128 bits (16 bytes) called a *state,*
- e.g.



tranformations used in each decryption round

AES



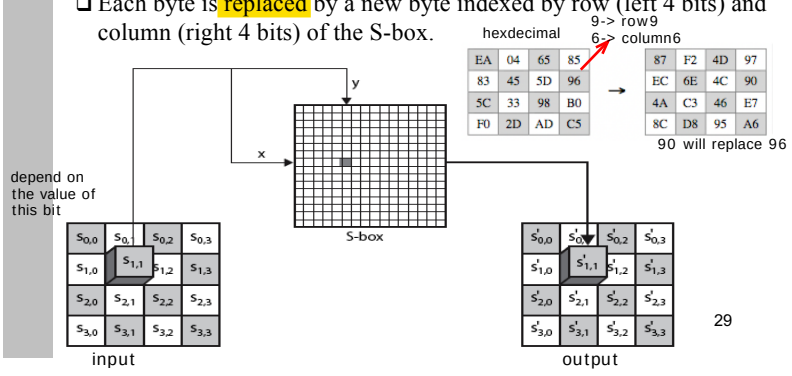
E and D is different so it is not identical
并不是相同的

AES – Structure

- Round transformation consists of:
  - Substitute bytes (SubBytes).
  - Shift rows (ShiftRows).
  - Mix columns (MixColumns).
  - Add round key (AddRoundKey).
- Sequential and light-weight key schedule.

AES – Substitute Bytes

- The SubBytes transformation is via a simple table/S-box lookup.
- One S-box for the whole cipher, a 16 × 16 matrix of byte values, that contains a permutation of all possible 256 8-bit values.
- Each byte is replaced by a new byte indexed by row (left 4 bits) and column (right 4 bits) of the S-box.



AES – Shift Rows

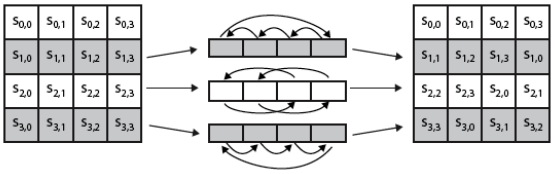
- The ShiftRows transformation is a simple permutation (circular byte shift):
  - 1st row: no change;
  - 2nd row: 1-byte circular left shift;
  - 3rd row: 2-byte circular left shift;
  - 4th row: 3-byte circular left shift.

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

- Decryption uses circular right shift.
- This step permutes bytes between the columns.



AES – Mix Columns

- A few notes about modular polynomial arithmetic, Galois Field, GF(P^n) (in AES, p=2, n=8)
- A bit-string (a_{n-1}, a_{n-2}, ..., a_1, a_0) is expressed in the form of a polynomial, i.e.

$$f(x)=a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\ldots+a_1x+a_0$$

- Arithmetic follows the ordinary rules of polynomial arithmetic using the basic rules of algebra, with the following two refinements:
  - Arithmetic on the coefficients is performed modulo p
    - when p=2, addition and subtraction are done by bitwise XOR.

AES – Mix Columns

to use primie polynomial

- If a multiplication result is a polynomial of degree greater than (n-1), then the polynomial is reduced by modulo some irreducible polynomial $m(x)$ of degree n, i.e., divide it by $m(x)$ and keep the remainder.
  - In AES, $m(x)=x^8+x^4+x^3+x+1$, i.e. 100011011 (or 11B). If the result is more than 8 bits, the extra bits are cancelled out by XORing the result with the 9-bit string (100011011).

- **mixColumn, along with shiftRows, provides diffusion.**

AES – Mix Columns

- Each byte of a column is mapped into a new value that is a function of all four bytes in the column; effectively a matrix multiplication in GF(2^8) using irreducible polynomial $m(x)$ $=x^8+x^4+x^3+x+1$ (or {11B})

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$$(\{02\}\bullet\{87\})\oplus(\{03\}\bullet\{6E\})\oplus\{46\}\oplus\{A6\}=\{47\}$$

$$\begin{bmatrix}02&03&01&01\\01&02&03&01\\01&01&02&03\\03&01&01&02\end{bmatrix}\begin{bmatrix}s_{0,0}&s_{0,1}&s_{0,2}&s_{0,3}\\s_{1,0}&s_{1,1}&s_{1,2}&s_{1,3}\\s_{2,0}&s_{2,1}&s_{2,2}&s_{2,3}\\s_{3,0}&s_{3,1}&s_{3,2}&s_{3,3}\end{bmatrix}=\begin{bmatrix}s'_{0,0}&s'_{0,1}&s'_{0,2}&s'_{0,3}\\s'_{1,0}&s'_{1,1}&s'_{1,2}&s'_{1,3}\\s'_{2,0}&s'_{2,1}&s'_{2,2}&s'_{2,3}\\s'_{3,0}&s'_{3,1}&s'_{3,2}&s'_{3,3}\end{bmatrix}$$

diffusion to achieve avlanche effect

AES – Mix Columns

- Arithmetic in the finite field GF(2^8) with irreducible polynomial $m(x)$ $= (x^8+x^4+x^3+x+1)$ ⇒ (1 0001 1011) or {11B}

For example:

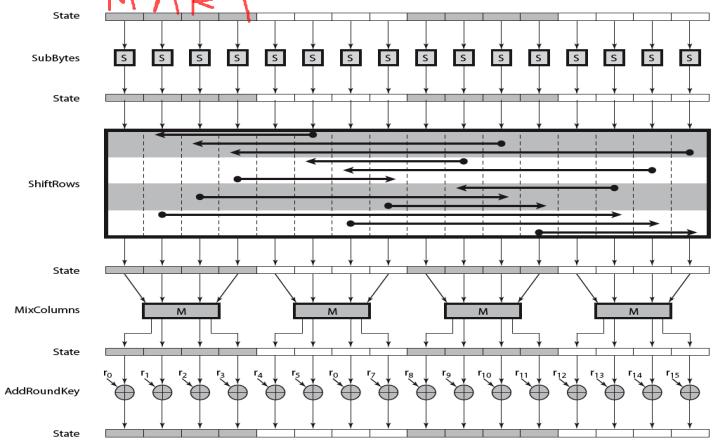
- {02} • {87} mod {11B}= (0000 0010)(1000 0111) = x (x^7+x^2+x+1) mod $m(x)$
  $= (x^8+x^3+x^2+x)$ mod $(x^8+x^4+x^3+x+1)$
  $= x^4+x^2+1$ = (0001 0101)
- {03} •{6E} = {11}{110 1110} = $(x+1)$ $(x^6+x^5+x^3+x^2+x)$ mod $m(x)$
  $= (x^7+x^6+x^4+x^3+x^2+x^6+x^5+x^3+x^2+x)$ mod $(x^8+x^4+x^3+x+1)$
  $= x^7+x^5+x^4+x$ ={1011 0010}

- 0001 0101⊕1011 0010⊕0100 0110⊕1010 0110=0100 0111=47

AES – Add Round Key

- In this AddRoundKey transformation, each byte of the state is combined with the round key using XOR, i.e. the 128 bits of state are bitwise XORed with the 128 bits of the round key.
- The round key is derived from the cipher key using a key schedule.

AES – One Round Operation



AES – Pseudo code

- **AES-128 (Encryption):**
  AddRoundKey(S,K[0]); K[0] is the cipher key, K, and other round keys are expanded from K.

  for (i = 1; i <= 9; i++)
  {
  SubBytes(S);
  ShiftRows(S);
  MixColumns(S);
  AddRoundKey(S,K[i]);
  }

  SubBytes(S);
  ShiftRows(S);
  AddRoundKey(S,K[10]).

AES-128 (Decryption) (first apply InvMixColumns to the round key)
AddRoundKey(S,K[10]);
for (i = 9; i >= 1; i--)
{
InvSubBytes(S);
InvShiftRows(S);
InvMixColumns(S);
AddRoundKey(S,K[i]);
}

InvSubBytes(S);
InvShiftRows(S);
AddRoundKey(S,K[0]).

DES versus AES

- DES:
  - Substitution-Permutation, iterated cipher, Feistel structure.
  - 64-bit block size, 56-bit key size.
  - 8 different S-boxes.
  - design optimised for hardware implementations.
  - closed (secret) design process.
- AES:
  - Substitution-Permutation, iterated cipher.
  - 128-bit block size, 128/192/256-bit key sizes.
  - 1 S-box.
  - design optimised for byte-orientated implementations.
  - open design and evaluation process.

Other Symmetrical Ciphers

Ciphers/Algos	Mode (block length in bits)	Key length (bits)
DES	Block cipher (64)	56
Triple DES	Block cipher (64)	168 (=3*56) (112 effective)
Rijndael	Block cipher (128, 192, or 256)	128, 192, or 256
Blowfish	Block cipher (64)	Variable up to 448
IDEA	Block cipher (64)	128
RC5	Block cipher (32, 64, 128)	Variable up to 2040

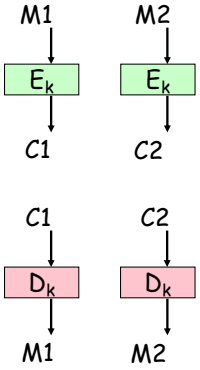
Part 4 Overview

- Use of Block Ciphers in Real World – Modes of Encryptions
- Block Ciphers vs Stream Ciphers
- Conclusion

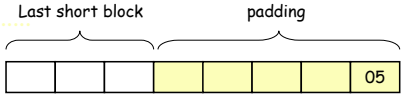
Modes of Encryption – encrypting large messages

- If a message is longer than a block size, block cipher can be used in a number of ways/modes to encrypt the message.
- Here we cover **three** modes of encryption/operations:
  - ECB – Electronic Code Book mode
  - CBC – Cipher Block Chaining mode
  - CTR – Counter mode
- These modes of encryption have been standardised internationally and are applicable to any block ciphers.

Modes of Encryption - ECB mode



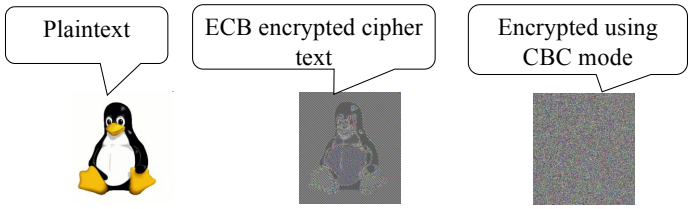
- $C_n = E_k(M_n)$ (or $E(K, M_n)$);
- $M_n = D_k(C_n)$; n = {1, 2, …}.
- Each block is encrypted independently using the same key. The last block should be padded if necessary.
- Usually the last byte indicates the number of padding bytes added; this allows the receiver to remove the padding.



Modes of Encryption - ECB mode

- Blocks are encrypted independently of other blocks
  - Reordering ciphertext blocks results in correspondingly reordered plaintext blocks.
- The same block of plaintext always produces the same ciphertext (with the same key)
  - patterns in plaintext show up in ciphertext.
- Error propagation: errors in one ciphertext block only affects the same plaintext block; they do not propagate to other blocks.
- Not recommended for messages longer than one block of data.

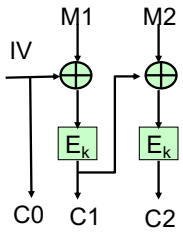
Modes of Encryption - ECB mode



- Source: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

Modes of Encryption - CBC mode

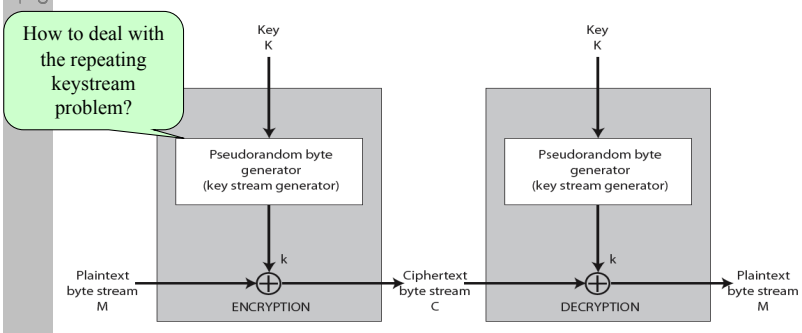
CBC encryption



- Equation for encryption: $C_i = E_K(M_i \textbf{ XOR } C_{i-1})$, where C_0 = IV (Initialization Vector).
- **In this example, the plaintext is *M1M2,* and the ciphertext is *C0C1C2.***
- Ciphertext block Cj depends on Mj and all the preceding plaintext blocks.
  - Reordering ciphertext blocks affects decryption.
  - Repeated patterns in the plaintext are 掩盖 concealed by the feedback.
  - There is error propagation.
- Using different *IV*s in different encryption operations will make the same plaintext encrypted to different ciphertexts.

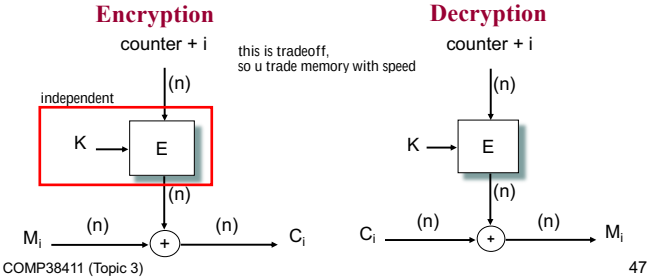
Recall this slide from Topic 2: stream cipher

Generate a keystream from a short key that initializes the generator.



Modes of Encryption – CTR mode

- The idea is to use a block cipher encryption function as the pseudorandom number generator to generate the key stream. 其思想是使用一个块密码加密函数作为伪随机数生成器来生成密钥流。
- A counter value, equal to the block size, is used. The value must be different for each encryption operation.
- Typically the counter is initialised to some value, and then incremented by 1 for each subsequent block (modulo 2^n, where n is the block length).



Modes of Encryption – CTR mode

voice communication in 4g, delay -> steam cipher

- The CTR mode actually converts a block cipher into a stream cipher.
- Each block can be decrypted independently of the others
  - Parallelizable.
  - Support random access.
  - The values to be XORed with the plaintext can be pre-computed.
- The counter needs to be synchronised
  - If a block is inserted into or deleted from the ciphertext stream then synchronization is lost and the plaintext cannot be recovered.
- No error propagation
  - a ciphertext block that is modified during transmission affects only the decryption of that block.

Why in CTR mode, only the encryption function of a block cipher is used (decryption is not needed)?

Block Ciphers vs Stream Ciphers

- While block ciphers encrypt blocks of characters, stream ciphers encrypt individual characters or bit streams.
- Stream ciphers
  - are usually faster than block ciphers in hardware; mostly used for continuous communications and/or real-time applications.
  - requires less memory space, so cheaper for resource restrained devices such as embedded sensors.
  - have limited or no error propagation, so advantageous when transmission errors are probable.
  - can be built out of block ciphers, e.g. by using CTR modes.

Exercise Question – E3.1

- By applying DES twice using two different 56-bit keys, K1 and K2, to encrypt a message, M, i.e. $C=E_{K2}[E_{K1}[M]]$, where C is the ciphertext, we have a double DES encryption. Would this double DES encryption double the security level of a single DES encryption? Justify your answer

Assuming n1 and n2 are, respectively, the lengths of K1 and K2 and n1 = n2 = n.

• Case 1: without using meet-in-the-middle method, i.e. by using brute-force attack
The anticipated number of attempts before compromising the encryption is: 2^2n/2=2^2n-1

• Case 2: using meet-in-the-middle method
With the Meet-in-the-Middle attack, the attacker first computes EK1(M) for all values of K1 and DK2(C) for all possible values of K2. He then compares the results from the two sets. If the result from any of the EK1(M) set matches with a result from the DK2(C) set, the pair of K1 and K2 is probably the correct keys. In this case, the anticipated number of attempts before compromising the encryption is: 2^n+ 2^n=2^n+1.
• As 2^n+1 <2^2n-1 (when n>2), so: an attacker can use the Meet-in-the-Middle attack to attack the double DES scheme more efficiently than the brute-force attack.

Exercise Question – E3.2

The diagram on the next page illustrates an early version of the ATM (Automatic Teller Machine) solution. From the diagram, it can be seen that:

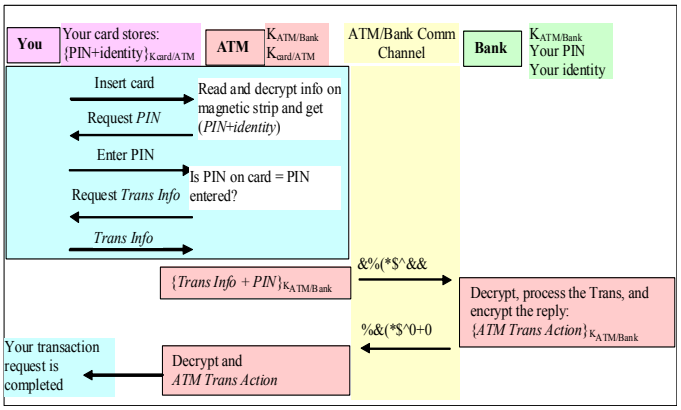
- Cash card stores the ciphertext of the user's Identity (ID) and PIN that are encrypted using a symmetric key, $K_{card/ATM}$.
- The communication between ATM and bank backend office is secured using another symmetric key, $K_{ATM/Bank}$.

Answer the following questions:

(i) Identify any vulnerability in this solution, and propose a solution to address any vulnerability that you have identified.

(ii) Are there any other issues that you could identify from this application of symmetric ciphers?

Exercise Question – E3.2 continue



Conclusions

- Modern symmetric ciphers come in two variants: block ciphers and stream ciphers.
- The mostly used block ciphers are DES/3DES/AES; and the most recent block cipher standard is the AES - Rijndael.
- Both DES and AES obtain their security by repeated applications of a simple round function consisted of substitution, permutation, shift and key addition.
- To use a block cipher, one needs to specify a mode of encryption/operation:
  - the simplest mode is ECB mode, but it is not secure for long message encryptions.
  - CBC mode is the default mode in most commercial applications that encrypt more than one data block.
  - CTR modes can help you to convert a block cipher into a stream cipher.
- Symmetrical ciphers have a key exchange problem and do not support non-repudiation.

Topic 4: Public-key Cryptography (PKC)

Understand the principles of Public-Key Cryptography (PKC)

Source: Stallings' book, chapter 9

Overview

- Part 1
  - Introduction
- Part 2
  - Mathematical Basics for RSA
- Part 3
  - RSA Algorithm
  - Hybrid Cryptosystems
  - Conclusion

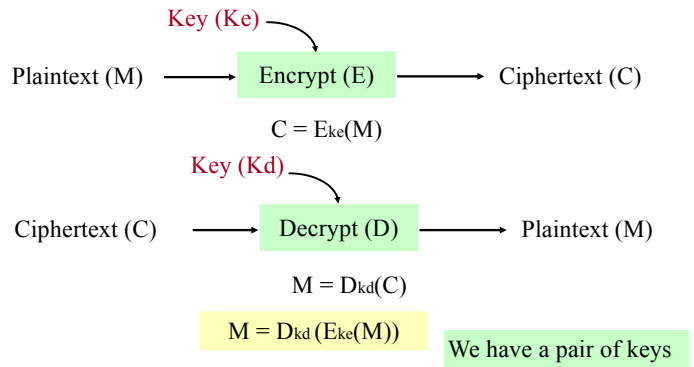
Introduction - Motivation

- Up to this point, all cryptographic schemes are based on shared secret keys - symmetric (or conventional) cryptography.
- The problems with symmetric cryptography - motivations
  - As two/more parties share the same key, non-repudiation can not be achieved without the involvement of a trusted third party.
  - A separate key is needed for each pair of users (or even for each ciphertext encryption – session key).
    - So an n-user system requires $n*(n-1)/2$ keys - the n^2 problem.
    - Generating and distributing these keys are a challenging problem.
    - Maintaining security for the keys already distributed is also challenging - can one remember so many keys?

Introduction - PKC Features

- In 1976, Diffie and Hellman (DH) first presented the notion of public key cryptography.
  - Keys could come in pairs - one public and one private; and it is infeasible to generate one key from the other; encryption produced by using one of the keys could only be reversed by the other key in the pair.

Introduction - PKC Features



We have a pair of keys here: {Ke, Kd}

Introduction - PKC Features

- PKC is based on the idea of a trapdoor function, or mathematically "hard" problems.
- The pair of private and public keys are related mathematically.
- Easy to generate keys (public and private).
- Hard to compute private key from public key.
- Easy to encrypt and decrypt if the right key is known.
- Hard to recover plaintext from ciphertext without the right key.

One-way function, f

C = f (M) "Easy"

M = f^{-1} (C) "Infeasible"

Trap-door one-way function, f

C = f (K, M) "Easy" if K & M known

M = f^{-1} (K, C) "Easy" if K & C known

M = f^{-1} (K, C) "Infeasible" if K not known, C known

Introduction - Commonly used One-Way Function

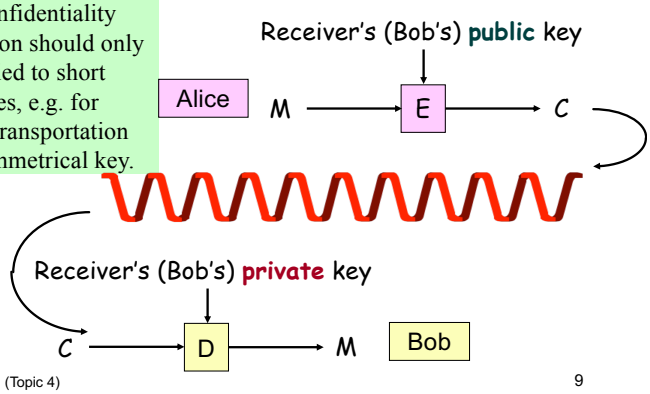
- Integer factorization (used in RSA)
  - Finding prime factors of a large integer: $n=p*q$
  - n is known
  - find p and q
- Discrete logarithm (used in DSS and DH)
  - $a^x=b \bmod p$
  - a, b and p are known
  - finding an integer, x, satisfying this equation

Public-key Cryptographic Algorithms

- Since 1976, numerous public-key cryptographic algorithms have been proposed. Among those secure and practical public-key algorithms
  - some are suitable for **encryption** (+ **key distribution**);
  - some are only useful for **digital signatures, e.g. DSA** (Digital Signature Algorithm; or DSS - Digital Signature Standard);
  - some are for **key agreement,** e.g. **DH** algorithm;
  - only three algorithms, **RSA,** ElGamal and Rabin, works well for both encryption and digital signatures.

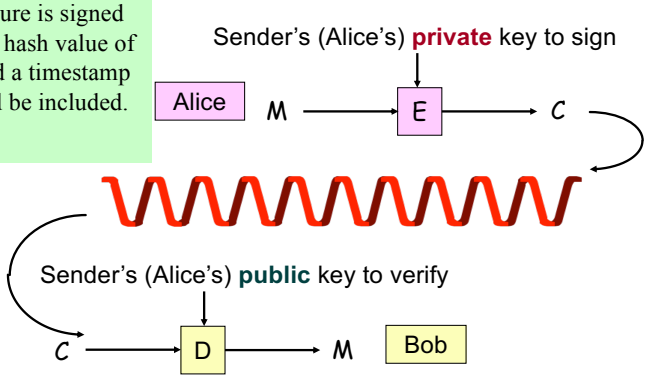
Introduction - Achieve Confidentiality (Secrecy)

- This confidentiality protection should only be applied to short messages, e.g. for secure transportation of a symmetrical key.



Introduction - Achieve Authenticity

- Signature is signed on the hash value of M, and a timestamp should be included.



Introduction - RSA Applications

- RSA is commonly used for
  - Confidentiality
    - ➢Encrypt the plaintext M using **recipient's public key;**
    - ➢As only the recipient has the corresponding private key, so M can only be read by the recipient.
  - Digital signature - message authenticity (message authentication or integrity) and non-repudiation of message origin
    - ➢Sign M (actually the hash of M) using **sender's private key;**
    - ➢As only the sender has this private key, so the message must have been signed by the sender.

Part 2 Overview

- Mathematical Basics for RSA

Mathematical Basics - Modular arithmetic

- Given some integer, n, the set of integers $[0, 1, \ldots, n-1]$ is the set of possible remainders when one divides any integer by n.
- This set is called the set of residues/remainders modulo n.

- **Mathematical definition**

$$a = b \bmod n$$

means there exists an integer number k such that a can be represented as

$$a = k \cdot n + b$$

with the condition that: $0 \le b \le n-1$

Here we are not interested in the value of k; the important thing is its existence.

Mathematical Basics - Modular arithmetic

- Given integers, a, b, and $n \neq 0$, *a is congruent to b modulo n if and only if*

$$a-b = k \cdot n$$

for some integer k, *i.e. n divides $(a-b)$.*
- We call n the modulus, and b is remainder or residue of a modulo n.
- Examples:
  - $9 \bmod 5 = 4$
  - $20 \bmod 9 = 2$
  - $17 = 2 \ mod\ 5$ *since* $17-2 = 3 \cdot 5$
- $x = y$ if and only if

$$(x \bmod n) = (y \bmod n)$$

An example
The modulo operator is commutative with the basic arithmetic operations. For example, it does not matter whether you **first multiply**

$$18 \cdot 13 = 234 = 4 \ mod\ 10$$

or first calculate the modulus **and then multiply:**

$$18 \cdot 13 = 8 \cdot 3 \ mod\ 10$$
$$= 24 \ mod\ 10 = 4 \ mod\ 10$$

Mathematical Basics - Modular arithmetic

- Properties
  - $a = a \ mod\ n$
  - $a = b \ mod\ n \Leftrightarrow b = a \ mod\ n$
  - $a = b \bmod n \ \&\ b = c \bmod n \Rightarrow a = c \bmod n$
  - $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
  - $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$
  - $a \cdot (b + c) \bmod n = (a \cdot b + a \cdot c) \bmod n$
  - $a \cdot x \bmod n = 1$ where x is an integer and called the multiplicative inverse of a; in this case, x can be written as a^{-1}, *i.e.* $a \cdot a^{-1} \bmod n = 1$.

Mathematical Basics - Modular arithmetic

- Existence of **multiplicative inverse**
  - Given $a \in [0, n-1]$, find $x \in [0, n-1]$ such that $a \cdot x \bmod n = 1$;
  - E.g. as $3 \cdot 4 \bmod 11 = 12 \bmod 11 = 1$, so we say, 3 and 4 are each other's multiplicative inverse mod 11.

- iff n is a prime, a and n are relative prime, i.e. $\gcd(a, n)=1$, then $a \in (0, n-1]$ has a unique inverse modulo n.

- An integer $p > 1$ is a **prime number** if it is divisible only by itself and 1, e.g. 7.
- a and b are said to be **relatively prime** if only 1 can divide each of them, e.g. are 8 and 15 relatively prime?

Mathematical Basics - Multiplication table *Mod 4*

x \ a	1	2	3
1	1	2	3
2	2	0	2
3	3	2	1

The inverse of 2 (mod 4) does not exist, because there isn't another number x in the finite field that can satisfy

$a.x = 1 \ mod\ 4.$

Not surprising!

There are two common divisors between a (2) and n (4), as 2 and 4 are not relatively prime.

Mathematical Basics - Multiplication table *Mod 11*

x \ a	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
4	4	8	1	5	9	2	6	10	3	7
5	5	10	4	9	3	8	2	7	1	6
6	6	1	7	2	8	3	9	4	10	5
7	7	3	10	6	2	9	5	1	8	4
8	8	5	2	10	7	4	1	9	6	3
9	9	7	5	3	1	10	8	6	4	2
10	10	9	8	7	6	5	4	3	2	1

Mathematical Basics - Multiplication Table Mod 11

- The *Table* gives the multiplication results for mod 11 (11 is a prime), the following can be noted:
  - In each row/column,
    - we can find *ALL* the integers in this set, and
    - each integer only appears **ONCE.**
  - For every integer, *a*, we can find another integer, *x*, that satisfies this equation:
    $a.x=1 \bmod 11$, $(a.x=1 \bmod n)$
    That is, *every integer* in this set has its multiplicative inverse.
  - For example, 1 and 1; 2 and 6; 3 and 4; etc.

Mathematical Basics - Multiplication Table Mod 11

- Here $n=11$ is a prime,
  - Given *a* and *b*, there is a unique solution, *x*, that satisfies:
    $a.x = b \bmod n$
  - All the integers in the set, $\{1, 2, \ldots, (n-1)\}$, are relative prime to *n,* i.e. there are $(n-1)=10$ such integers.
- **Euler's Totient/Phi Function (Euler's Theorem):**
  - If *n* is a prime, all of the positive integers, from *1* through to $(n-1)$, are relative prime to *n* and this is written as $\varphi(n)=(n-1)$.
  - If we have two different prime numbers, *p* and *q,* then for $n=p.q$,
    - $\varphi(n)=\varphi(p.q)=(p-1).(q-1)$
    - $a^{\varphi(n)}=1 \bmod n$

Part 3 Overview

- RSA Algorithm
- Hybrid Cryptosystems
- Conclusion

RSA Algorithm

- The algorithm was invented by Ron **R**ivest, Ali **S**hamir, and Leonard **A**dleman.
- It is by far the easiest to understand and implement.
- It has withstood years of cryptanalysis - remains by far most popular and well trusted scheme.
- The algorithm actually consists of two numbers, the modulus (represented by the letter ***n***) and the public exponent (represented by the letter ***e***).
- The modulus is the product of two very large prime numbers (100 to 400 digits), represented by the letters ***p*** and ***q***. ***p*** and ***q*** must be kept secret.

RSA Algorithm

- It is a block cipher. The block length is limited by some integer *n;* the plaintext and ciphertext, when being converted to integers, are between 0 and *n*-1.
- The algorithm has three functions:
  - **Key generation**
  - **Encryption**
  - **Decryption**

  } **Use the same mathematical function, but different keys.**

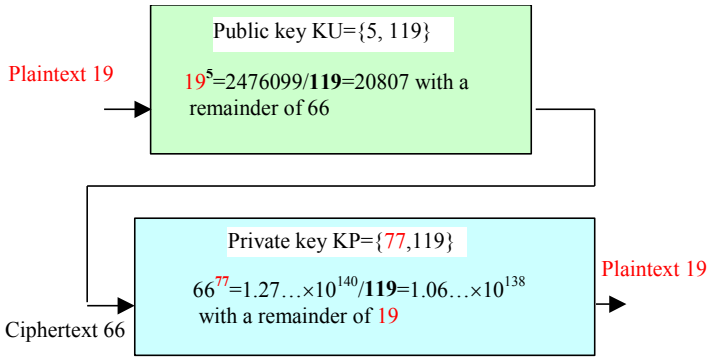
RSA Algorithm

- Key generation:
  - select two large primes (e.g. 200 digits) *p* and *q*
  - calculate $n = p.q$ and $\varphi(n) = (p - 1).(q - 1)$
  - select integer *e* relatively prime to $\varphi(n)$ & $1<e<\varphi(n)$
  - calculate $d = e^{-1} \bmod \varphi(n)$ (or $d.e = 1 \bmod \varphi(n)$)
  - **public key = {*e*, *n* }**
  - **private key ={*d*, *n*}**
  - *To summarise*:
    - *p, q* are private & chosen;
    - $n = p.q$ is public & calculated (but keep *p, q* secret);
    - *e* is public & chosen, and *d* is private & calculated.

RSA Algorithm

- Encryption:
  - represent the plaintext as an integer M in [0, n-1], i.e. $M < n$;
  - ciphertext: $C = M^e \bmod n$
- Decryption:
  - ciphertext: C
  - plaintext: $M = C^d \bmod n$
- An example of using RSA to encrypt a message
  - select p=7 and q=17
  - calculate $n = p.q = 119$ and $\varphi(n) = (p-1).(q-1) = 96$
  - select $e = 5$, relatively prime to $\varphi(n)$=96 and less than $\varphi(n)$
  - calculate d=77, such that $de = 1 \bmod \varphi(n)$ (= 96) and $d < 96$
  - let $M = 19$, then ciphertext $C = 19^5 \bmod 119 = 66$.

RSA Algorithm



Why RSA Works

- Euler's Theorem:
  - $a^{\varphi(n)} \bmod n = 1$ where $\gcd(a,n)=1$
- In RSA, we have:
  - $n=p.q$
  - $\varphi(n)=(p-1)(q-1)$
  - chose e & d to be inverses $\bmod \varphi(n)$
  - hence $e.d=1+k.\varphi(n)$ for some k
- hence:

$$C^d = M^{e.d} = M^{1+k.\varphi(n)} = M^1.(M^{\varphi(n)})^k$$
$$= M^1.(1)^k = M^1 = M \bmod n$$

RSA Algorithm - Standard

- PKCS#1 standard defines the use of RSA algorithm. It defines the key generation, encryption, decryption, digital signatures, verification, public key format, padding, and several other issues with RSA. It is probably the most widely used RSA standard, and most of the security protocols using RSA are also compatible with the PKCS#1 standard.
  - PKCS#1 standard - http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/index.html

RSA Algorithm - Some facts for the RSA

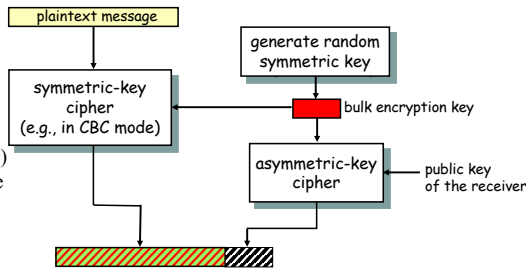
- Security of RSA relies on difficulty of finding d given $\{e, n\}$.
  - the problem of computing d from $\{e, n\}$ is computationally equivalent to the problem of factoring n
    - ➢If one can factorise n, then he can find p and q, and hence calculated d;
- p and q should differ in length by only a few digits, and both should be on the order of 100 - 200 digits or even larger.
  - n with 150 digits could be factored in about 1 year.
  - factoring n with 200 digits could take about 1000 years (assuming about 10^{12} operations per second).

Hybrid Cryptosystems

- Public key ciphers are much slower than symmetric key ciphers.
  - E.g. 1000 times slower in hardware, and 100 times slower in software, than DES.
- Symmetric ciphers
  - have key management problem.
  - can not provide non-repudiation service without the involvement of a trusted third party.
- So, usually, we combine them to get the strengths of both – this leads to the hybrid cryptosystem
  - Public cipher for symmetric key establishment/transportation and/or for digital signature generation.
  - Symmetric cipher for bulk encryption.

Hybrid Cryptosystems

- To speed the things up, hybrid encryption is used.
  - A symmetrical algorithm with a random session key (bulk encryption key) is used to encrypt the message;
  - A public-key algorithm is used to encrypt the random session (symmetric) key - digital enveloping.



What is the strength, and what is the remaining problem of this system?

Exercise Question – E4.1

- Name three application scenarios or cases where using RSA is preferable than using AES and name one application scenario where the use of AES is necessary.

Exercise Question – E4.2

- You are a recipient of p = 5, q = 7. You make the modulus n = 35 public. You also choose an exponent e = 5 and make that public too.
  Messages are sent to you, one letter at a time. Letters are coded into numbers as: A -> 0, B -> 1, and so on.

  Now, the following message has arrived for you:
  17 19 7 9 0 12 24

  Decrypt this message.

Conclusions

- Different from a symmetric-key cipher, in a public-key cipher, a pair of mathematically related keys are used, one for encryption and the other for decryption.
- Public key cryptography provides capabilities that can not be attained with symmetric cryptography, but it is too inefficient to be used alone - for large text encryption.
- PCK ciphers: RSA, DSS (Digital Signature Standard), DH (Diffie-Hellman), ...

Topic 5: Cryptographic Checksums

Understand the need for cryptographic checksums, how to construct them, and their applications

Source: Stalling's book: chapters 11 and 12

Overview

- Part 1
  - Motivation and Definitions
- Part 2
  - Cryptographic Hash Functions
- Part 3
  - Block Cipher based MAC (Message Authentication Code)
  - HMAC (hash function based MAC)
- Part 4
  - Authenticated Encryption
  - Conclusion

Threats to Message Security

- Content disclosure (confidentiality)
- Traffic flow analysis (traffic flow confidentiality)
- Sender masquerading/impersonation (entity authentication)
- Content modification (message authentication)
- Sequence modification (message authentication)
- Timing modification (message authentication)
- Repudiation of transmission/origin (message authentication + …)
- Repudiation of receipt

These terms are used interchangeably:
Checksum=digest=fingerprint=compressed value=hash value
MAC (Message Authentication Code) = cryptographic checksum

Need for a Cryptographic Checksum

Why do I emphasise *'a certain degree'* here?

- Conventional (symmetric) encryption, A --> B: $E_K[M]$
  - Provides confidentiality, as ONLY A and B share K.
  - Provides ***a certain degree*** of origin authentication, as it could only come from A or B.
  - Does not provide non-repudiation, as
    - A could deny sending the message – repudiation of origin.
    - B could also generate the encryption.
- Another variant, A --> B: $E_K[M||h(M)]$
  - Assuming that h is a compression function.
  - What is the difference between this protocol and the one above?
  - What benefit does this protocol provide vs the one above?

Need for a Cryptographic Checksum

- Public-key encryption, A --> B: $E_{KUb}[M]$ (using B's public key)
  - Provide confidentiality as only B has the decryption key.
  - Provides no origin authentication, as anybody could get hold of B's public key.
- Digital signing, A --> B: $M||E_{KRa}[h(M)]$ (using A's private key)
  - Provides origin authentication and non-repudiation, as
    - Only A has KR_a, so signed item (called checksum) must have come from A.
    - Any party can use KU_a to verify the signed item.
    - Provided KU_a is trust-worthy, and the signature is dated.
  - Provide no confidentiality.

Need for a Cryptographic Checksum

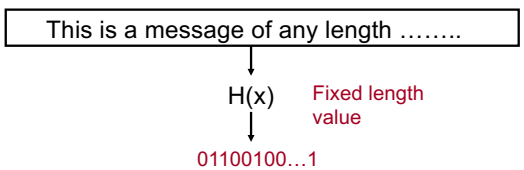
- Cryptographic checksum can be used to achieve
  - Content authentication (= origin + integrity) for any kind of messages including unstructured.
  - Non-repudiation of origin provided
    - the communication parties trust each other, or
    - the checksum is uniquely associated to the message originator, if the parties do not trust each other
  - Anti-replay (i.e. achieve freshness)
- It is attractive to applications that only require authentication
  - secure broadcast;
  - source code distribution.
- A cryptographic checksum should be
  - hard to forge and tamper-proof

Part 2 Overview

- Cryptographic Hash Functions

Hash Functions – Compression Property

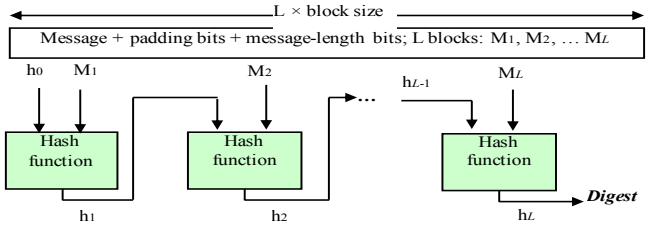
- Given a message M of arbitrary length, a hash function, H, produces a fixed-sized output, h (called a message digest, checksum, hash value, or fingerprint, of M), i.e. $h=H(M)$.
- H should be a function of all the bits of M.



This is a many-to-one mapping, so collisions are unavoidable, but we should make finding collisions as difficult as possible.

Hash Functions – Compression Property

- Each hash function execution processes a block of M; the output is the input for the next iteration; the output of the last block is the digest for M.



h0 is a constant initial value, and the output of the last block *is the digest* of the entire message.

Hash Functions - Security Requirements/Properties

Cryptographic hash functions are hash functions with additional security requirements/properties:

- Preimage resistant (one-way):
  Given $y \in Y$, it is computationally infeasible to find a value $x \in X$ s.t. $H(x) = y$.
- 2-nd preimage resistant (weak collision-resistant):
  Given $x \in X$, it is computationally infeasible to find another value $x' \in X$, s.t. $x' \neq x$ and $H(x') = H(x)$.
- Collision resistant (strong collision-resistant):
  It is computationally infeasible to find two distinct values, x' and $x \in X$, s.t. $H(x') = H(x)$.

PS: If H is strong collision-resistant, then it is also weak collision-resistant.

Hash Functions – Importance of the Security Properties

- **Signature forgery if weak collision resistance property is not met.**
  - Assuming that *A* has sent a signed message *M* to *B, i.e.* $M||s$ where $s=E_{KRa}[H(M)]$ and *KRa* is A's private key;
    - An attacker intercepts *A*'s signature and message;
    - The attacker finds another message *M'* with $H(M)=H(M')$;
    - The attacker now has your signature *s* on the message *M'*.
  - Think about the implication of this attack in real-life!

Hash Functions – Importance of the Security Properties

- **Repudiation if strong collision resistance property is not met.**
  - Assuming that *A* is to send a signed message *M* to *B*
    - *A* chooses two messages *M* and *M'* with $H(M)=H(M')$;
    - *A* signs *M* by generating signature $s=E_{KRa}[H(M)]$;
    - *A* sends *B* $M||s$;
    - Later *A* repudiates this signature, saying it was really a signature on the message *M'*.
  - Think about the implication of this attack, if
    - The communication is for *A* to make an e-payment; and
    - *M* is an electronic cheque for £10.
    - *M'* is an electronic cheque for £1000.

Hash Functions – Commonly used Hash Functions

	SHA-1	SHA-256	SHA-384	SHA-512
Hash value size	160	256	384	512
Block size	512	512	1024	1024
Word size	32	32	64	64
Security	80	128	192	256

* All sizes are measured in bits; *SHA = Secure Hash Algorithm

* Security refers to the fact that a birthday attack on a message digest of size *n* produces a collision with a work-factor of approx $2^{n/2}$.

Hash Function Applications

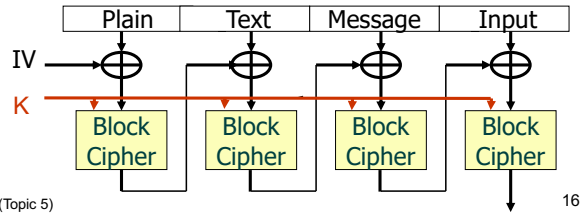
- Secure storage of passwords
- Digital signatures
- Pseudo-random number generations
- Bit commitment (or coin flipping) problem
- Bit coins
- Blockchains
- Digital payment systems
- Digital right management systems
- …

Part 3 Overview

- Block Cipher based MAC (Message Authentication Code)
- HMAC (hash function based MAC)

Message Authentication Code (MAC) using Block Ciphers

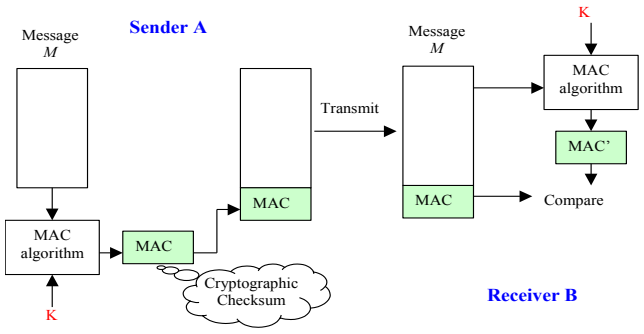
- A cryptographic checksum can also be generated using a symmetric block cipher, i.e. **MAC=$f_K(M)$,** where f_K is a block cipher based digest function.
- An example is CBC-MAC:
  - Can be any symmetric block cipher
  - the output of the last block is MAC
- The block cipher based digest function has an embedded key.



MAC in Operation

- **Sender**
  - uses K and a MACing function, f, to generate a checksum, $MAC=f_K(M)$.
  - then sends $M||MAC$, where || is concatenation of data items.
- **Receiver**
  - computes $MAC' = f_K(M')$, where M' is the message received, and K' is receiver's copy of the key.
  - If $MAC=MAC'$, then the message has not been tampered with.

MAC in Operation



MAC in Operation

- If only A and B know the secret key K, and if MAC=MAC', then the receiver can be assured
  - the message has not been altered - integrity protection;
  - the message is from the alleged sender - origin authentication;
  - the message is of the proper sequence if the message includes a sequence number;
  - the message is fresh - *i.e,* not a replay
    - if the message includes a timestamp; or
    - a random number contributed (fully or partially) by B (the recipient).
- The MAC operation described above is also applicable to MACs generated with another method, e.g. a hash function.

HMAC

- KeyedHash = Hash(K||Message)
- HMAC is a keyed hash function, specified as Internet standard

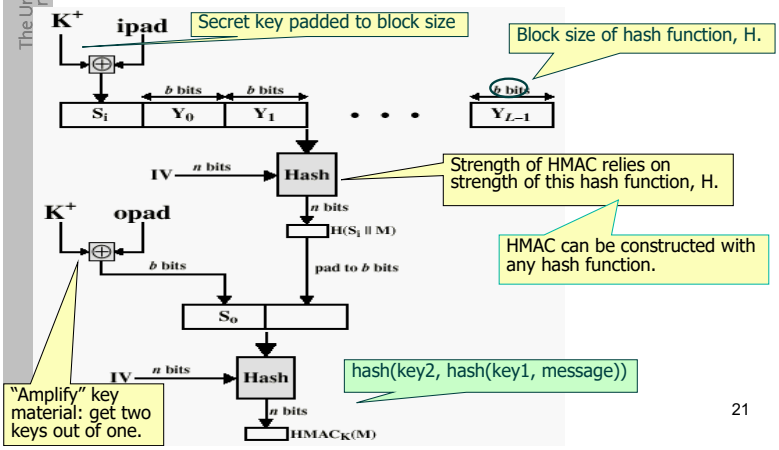
H(K2||H(K1||M))

HMAC(K, M)= H[(K^+ XOR opad) || H[(K^+ XOR ipad) || M)]];

where

H = any hash function such as SHA-256;
K^+ is the key padded out to block size;
ipad = a string by repeating the byte 0x36 (00110110) as often as necessary;
opad = a string by repeating the byte 0x5c (01011100) as often as necessary.

HMAC



Security of MACs

- Brute force attacks
  - Finding collisions cost $2^{n/2}$, where n is the bit-length of MAC value. The larger the n value, the more secure it is.
  - Attack key space or MAC by exploiting MACs with known message-MAC pairs.
- Worth mentioning: a MAC is not a digital signature; it does not provide non-repudiation.

Part 4 Overview

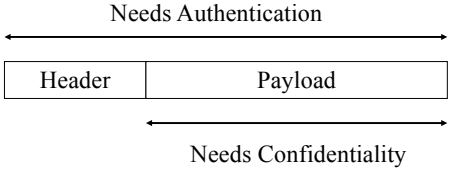
- Authenticated Encryption
  - Part 4.1 – Overview
  - Part 4.2 - CCM: Counter Mode with CBC
  - Part 4.3- GCM: Galois/Counter Mode
- Conclusion

Authenticated Encryption

- This is for achieving both message authentication and confidentiality.
- Possible approaches:
  - Hash-then-encrypt: E(K, (M||H(M))
  - MAC-then-encrypt (used in SSL): E(K2, (M||MAC(K1, M)); or Tag= MAC(K1, M), E(K2, (M||Tag)); Tag is for authentication
  - Encrypt-then-MAC (used in IPSec): C=E(K2, M), Tag=MAC(K1, C)
  - Encrypt-and-MAC (used in SSH): C=E(K2, M), Tag=MAC(K1, M)

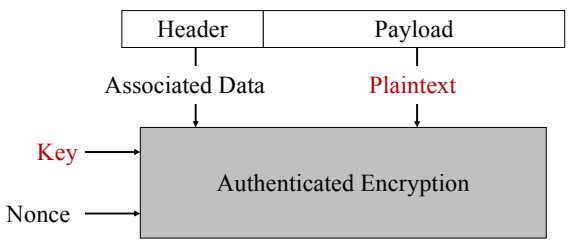
Authenticated Encryption with Associated Data (AEAD)

- In some contexts, such as networks, a message (packet) consists of two parts, a header field and a payload, and the two parts often have different security requirements.
- In addition, the protections should be provided efficiently (with as less overhead as possible).



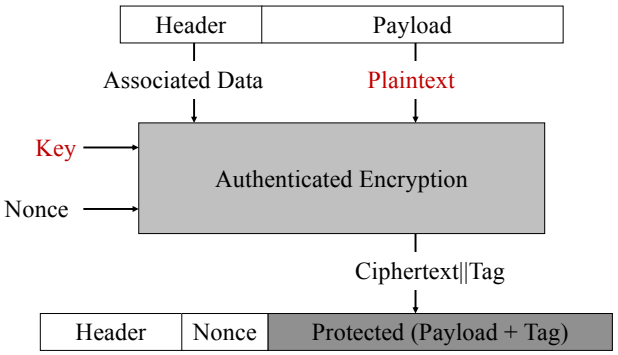
Authenticated Encryption with Associated Data

- Plaintext (Payload) should be encrypted AND authenticated.
- Associated Data (Header) should be authenticated (without encryption).
- Nonce should be distinct for each AEAD operation.



Authenticated Encryption with Associated Data

- Tag also goes through the encryption operation.

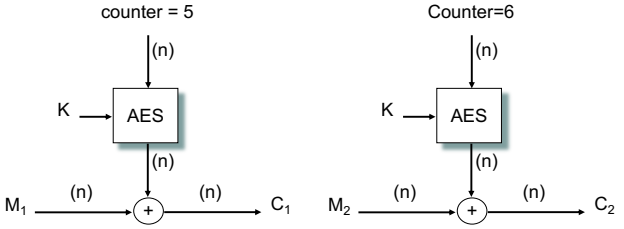


Authenticated Encryption with Associated Data

- Two notable schemes: CCM and GCM
  - CCM: Counter Mode with CBC-MAC (NIST SP 800-38C for WiFi)
  - GCM: Galois/Counter Mode (NIST standard SP 800-38D)

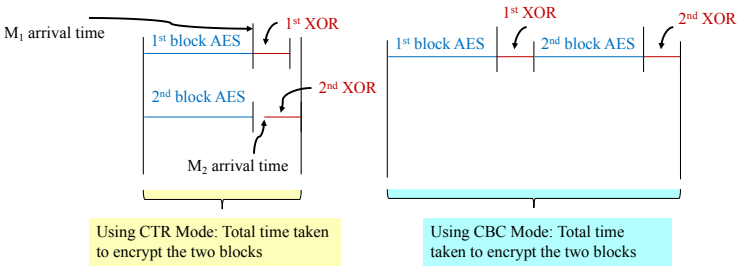
CCM: Counter Mode with CBC-MAC

- CTR encryption example: assuming two plaintext blocks, counter value is initialised to 5, AES is chosen as the block cipher.
  - Counter value increment for each subsequent plaintext block.
  - Parallel/pipelined computation/processing/operation.
  - Encryption/decryption are done using XOR – very fast.



CCM: Parallel/pipelined computation

- Computation time: pipelined vs non-pipelined operations

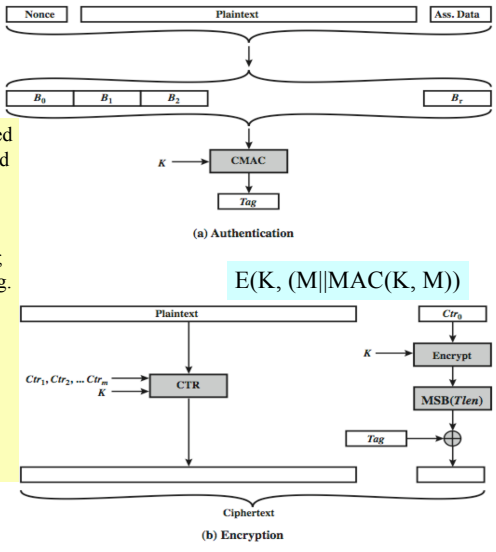


CCM

Authentication Tag is generated using CBC-MAC (also referred to as CMAC):
Tag=CMAC(K, M)
where Plaintext=Payload; M=Nonce||Plaintext||Ass.Data; Ass.Data=Associated Data (e.g. packet headers)

Encryption is by AES CTR mode: E(K, Payload||Tag)

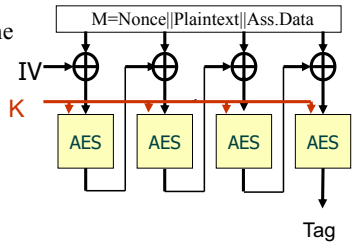
Nonce (random number) to ensure Tag is fresh.



E(K, (M||MAC(K, M))

CCM: Counter Mode with CBC-MAC

- Tlen (Tag Length) most significant bits of the output are XORed with the tag to produce an encrypted tag.
- Same key (can be different) for MACing and encryption.
- Plaintext passes 2 block cipher operations (MAC and encryption).
- CBC-MAC is not pipelinable/parallelizable.



- To speed up the performance further, we have GCM.

GCM

- Galois/Counter Mode (GCM)
- GCM is designed to provide authenticated encryption for high speed applications (can do authenticated encryption at > 10 gigabits per second)
- Combines counter mode with Galois mode
- Counter mode can be pipelined and parallelized in hardware implementations
- Rather than using block chaining to generate tags (MACs), GCM uses a Galois field multiplication that is less computationally intensive than CBC-MAC
- Cost half of the number of AES operations used in CCM
- NIST standard SP 800-38D

GCM

- GCM has two functions, authenticated encryption and authenticated decryption.
- **Authenticated encryption**: Given a selected block cipher & *K*, authenticated encryption function has **3 inputs:**
  - Plaintext, denoted *P*.
  - Additional authenticated data (AAD), denoted *A*.
  - An initialization vector, denoted *IV* (*Nonce* to ensure freshness).
- It generates **two outputs**:
  - A ciphertext, denoted *C,* which has the same length as *P*.
  - An authentication tag, denoted as *T, used to protect authenticity of P, A* and *IV*.

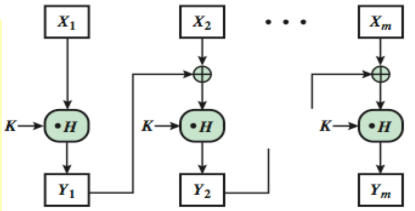
GCM

- **Authenticated decryption**: Given a selected block cipher & *K*, this function has **4 inputs**: *IV*, *A*, *C*, and *T*.
- The output is one of the following:
  - plaintext *P* that corresponds to the ciphertext *C*, or
  - a special error code, denoted *FAIL*.
- The output *P* indicates that *T* is the correct authentication tag for *IV*, *A*, and *C*; otherwise, the output is *FAIL*.

GCM

- Uses two functions:
  - GHASH - a keyed hash function over a binary Galois field.
  - GCTR - a variation of CTR mode (with a particular incrementing function, denoted *inc*).
- GHASH: plaintext xor'ed with feedback and multiplied with ***H*** (a hash subkey) in GF(2^{128}).

Y_0 is a 128-bit length "zero block", 0^{128}. For i=1 ,…, m, $Y_i=(Y_{i-1} \oplus X_i)\cdot H$, where block ***H* is the hash subkey**, generated by applying the block cipher to the "zero" block.
Y_m is the output.



(a) $\text{GHASH}_H(X_1 \| X_2 \| \ldots \| X_m) = Y_m$

GHASH

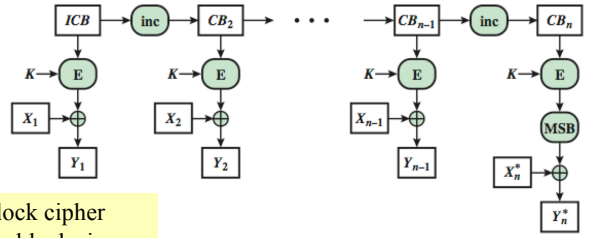
- GHASH function can be expressed as:

$Y_m=(X_1 \cdot H^m) \text{ XOR } (X_2 \cdot H^{m-1}) \text{ XOR } \ldots \text{ XOR } (X_{m-1} \cdot H^2) \text{ XOR } (X_m \cdot H)$

where • designates multiplication in GF(2^{128})

GCM

- Y_i in the two figures, (a) and (b), are not related.
- ICB=Initial Counter Block.
- MSB$_s$(X): bit string consisting of s left-most bits of the bit string X.

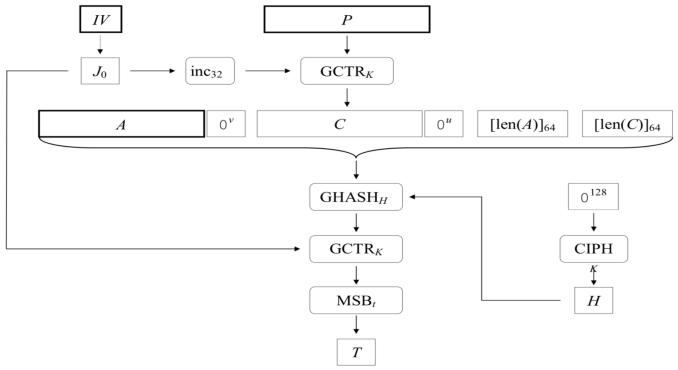


E=CIPH; a block cipher with a 128-bits block size

(b) $\text{GCTR}_K(ICB, X_1 \| X_2 \| \ldots \| X_n^*) = Y_1 \| Y_2 \| \ldots \| Y_{n-1} \| Y_n^*$

GCM

- J_0=ICB (Initial Counter Block)



GCM

- *GHASH* compresses an encoding of AAD (*A*) and the ciphertext (*C*) into a single block, which is then encrypted to produce the authentication tag, *T*=*Tag*.
- *C* can be null, resulting a variant of GCM, called GMAC, generating and verifying tag on non-confidential data.
- J_0 (ICB=Initial Counter Block) is generated from *IV*.
- Inc_{32}: an increment function; it increments the right-most 32 bits of the string with the remaining bits unchanged.
- *CIPH*: a block cipher with a 128-bits block size, and the key size should be at least 128 bits.

Exercise Question – E5.1

For a hash value to be used as a cryptographic checksum, it must be protected with a secret, as it is clear, from the above, that a hash function does not have an embedded key.

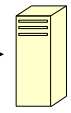
Assuming that a sender, s, is to send a message, M, to a receiver, r. Propose as many different methods as you can to protect the hash value of M to assure the authenticity of M. Comment on the suitability/applicability of each of the methods you propose.

Exercise Question – E5.2

For each of the following applications, please identify what property(ies) the hash function needs to have to ensure the security of the application.

(i) Secure storage of passwords

Psword = myDoB



Password file:
User_A ***
User_B ***

What should we put in there?
What if backup tape is stolen?
What property do we need?

(ii) Protection against viruses

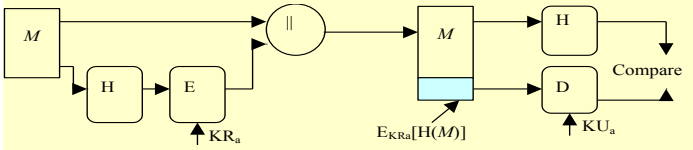
- A software manufacturer wants to ensure that the executable files are received by users without modification.
- They send out each file to users and publish its hash value on an authentic website.

Exercise Question – E5.2 (continue)

(iii) Digital signatures

One party can sign a message, M, and many parties can verify.
Such applications include contract signing, code signing, etc.
A raw signature scheme only signs a few hundred (e.g. 160) bits.
What properties do we need?

Integrity, authentication, and *non-repudiation* are provided.
*This is the essence of the digital signature technique.



Exercise Question – E5.3

- This is a Coin Flipping Over the Telephone problem.

(i) Assuming there is only one car, and Alice and Bob have to decide who can have this car (only one of them can have it, i.e. they cannot share it). Alice and Bob cannot see each other, and they do not trust each other. So they have decided to make a decision by flipping a coin over the telephone. Design a protocol to support this using a hash function.

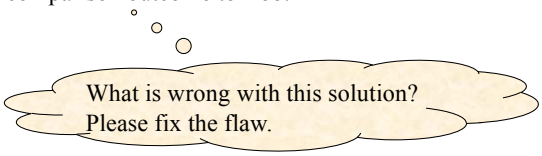
(ii) Identify any factors that you should consider to ensure the security of this protocol.

Exercise Question – E5.3 (hint)

Assumption: Alice and Bob agree that if the outcome is 1 then Bob takes the car, if it is 0 then the car goes to Alice.

Solution 1 (an insecure solution):

- Alice generates a random bit *b*: 0=heads, 1=tails.
- Alice asks Bob: heads or tails?
- Bob sends Alice his choice_B: 'heads' (or 'tails').
- Alice compares *b* with choice_B: if b=choice_B, then outcome=1; if not, outcome=0.
- Alice sends the comparison outcome to Bob.



Conclusions

- Message encryption can not always provide message authentication.
- Message authentication is typically achieved by using a cryptographic checksum.
- A checksum produced from a hash function should be protected with a private key (of a public-key cipher) or a symmetric key.
- A symmetric-key protected checksum may also be produced by using a block cipher in CBC mode, or a keyed hash function such as HMAC.
- Authenticated encryption is for achieving message authentication as well as confidentiality in a way that maximises parallel operations.
- Hash functions are commonly used in many other applications as well.

Topic 6: Digital Signature Algorithms

Understand two mostly used digital signature schemes

Source: Stalling's book, chapter 13

Overview

- Part 1
  - Digital Signature Overview
  - Digital Signature using RSA
- Part 2
  - DSS (Digital Signature Standard, also called DSA - digital signature algorithm)
  - RSA vs DSA
  - Conclusion

Digital Signature Overview

- A **digital signature** is a technique for establishing the origin of a particular message such that any future disputes with regard to what message was sent and who sent it could be resolved by any third party.
- According to the European Community Directive on digital signatures, a digital signature should be:
  - uniquely linked to the signatory
  - capable of identifying the signatory
  - created using means under the sole control of the signatory
  - linked to data to which it relates in such a way that subsequent changes in the data is detectable.

Digital Signature Overview

- A digital signature associates a mark unique to an individual with a body of text.
- Security requirements:
  - message-dependent, inc date/time
    - unreusable
    - ensures content integrity
  - signer-dependent
    - unforgeable
    - ensures origin authentication
  - verifiable: others should be able to verify the validity of a signature.
  - anti-forgery: computationally infeasible to forge.

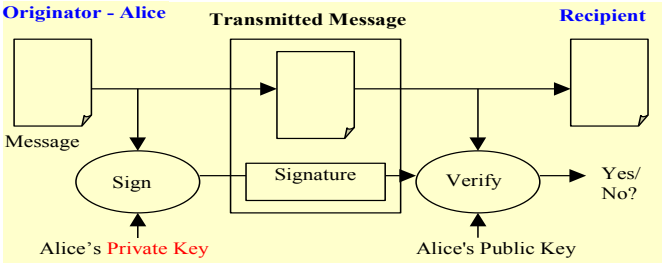
Both integrity and origin authenticity are necessary to ensure **non-repudiation**, i.e. the signer can not falsely deny that he/she has generated the signature.

Digital Signature Overview

- Forgery types
  - Existential forgery: the creation (by an adversary) of any message/signature pair (*M, S*), where *S* was not produced by any legitimate signer.
  - Selective forgery: the creation (by an adversary) of a message/signature pair (*M, S*), where *S* has been *chosen* by the adversary prior to the attack.

Digital Signature Overview

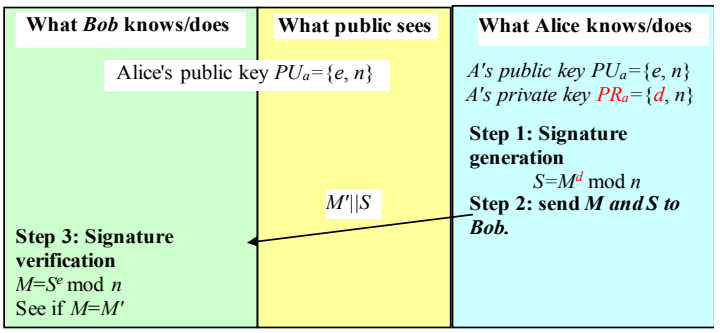
- There are arbitrated digital signatures.
- BUT in most cases, a digital signature is generated using a public-key algorithm.
- PKC based digital signature model:



Digital Signature Overview

- The main idea is
  - only *A* can sign a message, since only *A* has access to the private key.
  - anyone can verify *A*'s signature, since everyone has access to her public key.
- A digital signature scheme consists of:
  - A key generation algorithm
  - A signature (generation) algorithm
  - A signature verification algorithm

Digital Signature Overview – Improper way



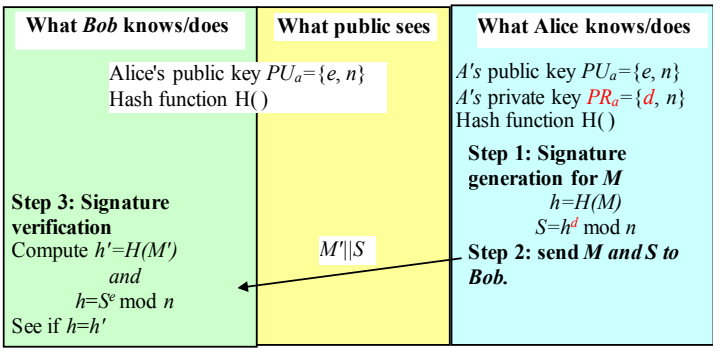
Digital Signature Overview - Improper way

- Performance concern: public-key cipher operations are time consuming and signing long messages is costly.
- Security concern: loopholes for signature forgery
  - *Example 1*
    - Let *s* be a random value, apply the public key (e, n) *to s: P= s^e (mod n) = m*
    - Then *(m, s) is a valid message-signature pair.*
  - *Example 2*
    - Let *m* be the message of which you want to forge a signature.
    - Choose two messages *x and y* such that *xy = m (mod n)*
    - If you could obtain the signatures of *x and y, then,* you can easily forge a signature of *m* by *S(m) = S(y)S(x) (mod n)*

Digital Signature Overview - Proper way

- Use "hash-and-sign" paradigm: signing the hash value of a message.

Digital Signature using RSA - Proper Way



Sometimes, we write in this format: $S=E_{PRa}[H(M)]$ for signature generation, and $h= D_{PUa}[S]=D_{PUa}[E_{PRa}[H(M)]]$ for signature verification.

Part 2 Overview

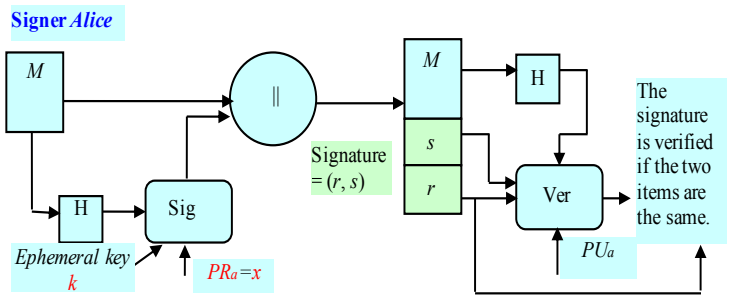
- DSS (Digital Signature Standard, also called DSA - digital signature algorithm)
- RSA vs DSA
- Conclusion

DSS/DSA - Background

- In 1991, the NIST (National Institute of Standards and Technology) proposed the **DSA** (Digital Signature Algorithm) for use in their **DSS** (Digital Signature Standard).
- Unlike RSA, DSA is a digital-signature-only algorithm.

DSS/DSA - Algorithm Overview

- k is a random secret number just generated for this signature.
  - (k, r) = (ephemeral private key, ephemeral public key); only used for this signature.
- (PR_a, PU_a) = sender's (private key, public key).



DSS/DSA - Key Generation

- Have shared global public key values (p, q, g)
  - choose a large prime p with $2^{L-1} < p < 2^L$, where L= 512 to 1024 bits.
  - choose 160-bit prime number q, such that q is a divisor of (p-1).
  - choose $g = h^{(p-1)/q} \bmod p$, where $1<h<p-1$ and $h^{(p-1)/q} \bmod p > 1$
- Users choose (long-term) private and compute public key
  - Choose random private key, x, i.e. $PR_a = x$, with $0< x <q$.
  - Compute public key: $PU_a = y = g^x \bmod p$.

DSS/DSA - Signature Generation

- to **sign** a message M the sender
  - Chooses a random number (the ephemeral key), k with $0 < k < q$; k must be destroyed after use, and must never be reused.
  - Computes the signature pair:
    - $r = (g^k \bmod p) \bmod q$.
    - $s = [k^{-1} (H(M)+xr)] \bmod q$.
    - $Signature = (r, s)$.
- Sends $M||Signature$ to the receiver.

DSS/DSA - Signature Verification

- The receiver has got $PU_a = \{p, q, g, y\}$, and $\{M', r', s'\}$.
- To verify the signature, he computes
  - message hash $H(M')$.
  - mod q inverse of s': $w=(s')^{-1} \bmod q$.
  - $u_1 = [H(M')w] \bmod q$.
  - $u_2 = (r')w \bmod q$.
  - $v = [(g^{u1} y^{u2}) \bmod p] \bmod q$.
- And check if: $v = r'$; if true, then the signature is verified (U prove!).
- Here, M = message to be signed; H(M) = hash of M using a hash function; M', r', s' = received versions of M, r, s.

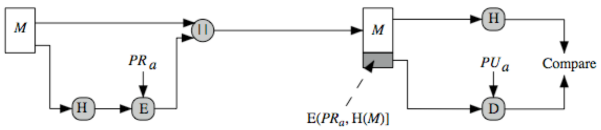
DSS/DSA - a Baby Example

- Key generation
  - Generate q, p and g
    - q=13; p=4q+1=53; g=16;
  - Generate private key: x=3;
  - Compute public key: $y=g^3 \pmod p$ =15;
- Signature Signing
  - assuming H(M)=5;
  - choose k=2;
  - $r = (g^k \bmod p) \bmod q = (16^2 \bmod 53) \bmod 13 = 5$;
  - $s = [\ (H(M)+xr)*k^{-1}] \bmod q = [(5+3*5)*2^{-1}] \bmod 13 = 10$.

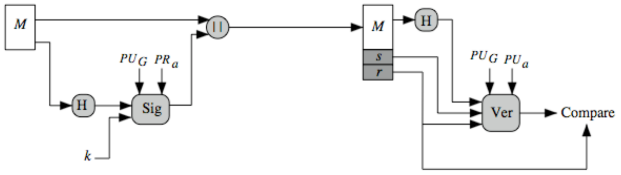
DSS/DSA - a baby example

- Signature Verification
  - computes
    - message hash $H(M')$ =5.
    - mod q inverse of s':
      $w=(s')^{-1} \bmod q = (10)^{-1} \bmod 13 = 4$.
    - $u_1 = [H(M')w] \bmod q = 5*4 \bmod 13 = 7$.
    - $u_2 = (r')w \bmod q = 5*4 \bmod 13 = 7$.
    - $v = [(g^{u1} y^{u2}) \bmod p] \bmod q$
      $= [(16^7*15^7) \bmod 53] \bmod 13 = 5$.
  - Note $v = r'$, so the signature is verified.

RSA vs DSA



(a) RSA Approach



(b) DSS Approach

RSA vs DSA

RSA	DSA
Security is based on difficulty of factoring large numbers.	Security is based on difficulty of taking discrete logarithms.
Can encrypt and sign.	Can only sign messages.
	Some signature computations can be computed a priori, so generally faster.
A RSA signature is about 1k - 2k bits long, depending on the size of the modulus.	A DSA signature is 320 bit long, desirable for applications requiring smaller signature footprints.
Can recover the message digest from the signature.	Cannot recover the message digest from the signature.
	Need to choose a unique secret number k for each message.

Exercise Question – E6.1

(i) Discuss, at the generic level, what are the factors that impact on the security of a digital signature.

(ii) Assuming that the RSA algorithm is used for signature signing, identify all possible ways of forging a signature.

Exercise Question – E6.2

A digital signature scheme may also be implemented using a symmetric-key cipher, but with the assistance of a trusted third party, an Arbitrator.

(i) Design a digital signature protocol using symmetric-key encryption and an arbitrator, but do not expose the content of a message to be signed to the arbiter.

(ii) Compare the signature protocol designed in (i) with the RSA based signature scheme.

Conclusions

- Digital signatures provide message authentication (integrity & origin authentication) and non-repudiation security services.
- There are *two* well-known signature schemes
  - RSA encryption algorithm can be used in reverse to produce a signature.
  - DSA is a signature algorithm based on discrete logarithms and can only be used for signature purposes.
- A signature scheme should be used in conjunction with a hash function to obtain security in an efficient way.

Topic 7: Public Key Infrastructure (PKI)

Understand the PKI technologies for secure distribution of public keys

Source: Stalling's book, chapter 14; also lots of docs on this subject on the Internet.

Overview

- Part 1
  - Public Key Infrastructures (PKI) Overview
- Part 2
  - Digital Certificates
- Part 3
  - Certificate Revocation Lists (CRLs)
- Part 4
  - Certificate Hierarchies
  - Conclusions

PKI Overview

- PKI
  - provides functions, technologies, policies and services that enable practical deployment and wide-scale applications of public-key cryptography (PKC).
  - includes the management and control of public and private keys.
- Security properties/services offered by PKC include:
  - Certificate-based user/entity authentication.
  - Digital signing of electronic documents, emails, software for authentication (integrity) and non-repudiation protections.
  - Encryption, typically for symmetric key distributions.

PKI Overview

- Applications of PKI around us:
  - Web browsers, servers and services, e.g. SSL (secure socket layer).
  - Virtual Private Networks (VPNs), e.g. IPSec.
  - Secure email services, e.g. S/MIME, PGP (Pretty Good Privacy).
  - Secure file storage services, e.g. PGP.
  - Secure electronic transactions, e.g. SET.
  - Visa/Master smartcards.
  - Copyright protection (DRM – Digital Right Management).
  - …

PKI Overview

- When using public-key cryptography, two major issues should be considered:
  - **Issue 1:** How to ensure the security (secrecy and strength) of the private key.
    - The key size should be large enough.
    - The lifetime of the key should guard against brute-force attacks.
    - The key should be kept secret; they should be generated, transported, stored and destroyed (at the end of its lifetime) securely.

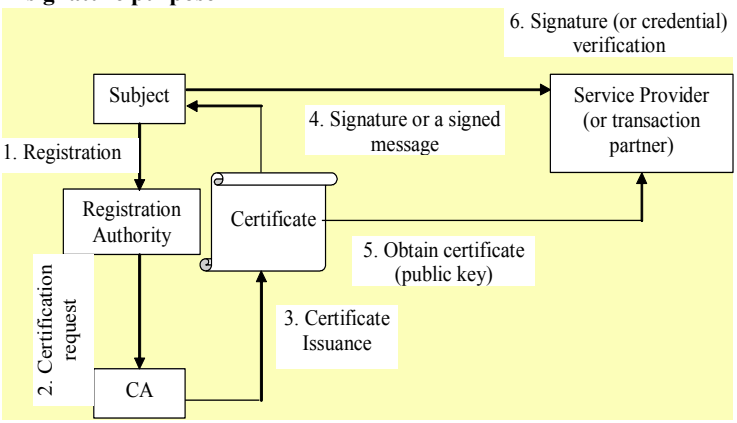
PKI Overview

  - **Issue 2**: How to ensure that a public key is trustworthy, i.e. how could we trust that a given public key indeed belongs to a claimed entity.
    - The solution is to have some trusted entity or authority to sign one's public key ➔ **digital certificate**.
    - Otherwise, communications are vulnerable to man-in-the-middle attack.
- A digital/PKI certificate is **a statement**:
  - certifying that this public key indeed belongs to this identity
  - the owner of this identity possesses the corresponding private key.
- When one uses a digital certificate, s/he must demonstrate that s/he knows the corresponding private key.
- Digital/PKI Credential = PKI certificate + the matching private key

PKI Overview - Main PKI entities

- **Registration Authority (RA)**: verifying the identity of a user requesting for a certificate.
- **Certificate Authority (CA)**: issuing and managing digital credentials
  - credential=private key + certificate
  - Key pair can be generated by a CA or by the requester
- **Data Repository**: typically a LDAP directory, is where certificates and revocation status are *officially* stored.

PKI Overview – A simplified view of acquiring a Cert for signature purpose



PKI Overview – A simplified view of acquiring a Cert for signature purpose

- Assumptions used:
  - 'You' (Subject) and 'the Service Provider (SP)' do not trust (or donot know) each other and You want to send a signed message to SP.
  - You have already got a pair of private and public keys and need to get your public key certified (a certificate for the public key).
  - CA is trusted by both You and the SP.

- Pls note: SP should ALSO have the CA's certificate (why?).

PKI – Main Functions

- SystemSetup: a **credential** service provider (usually CA) should get the policy, procedures and services ready, including key generation/update, **certificates** issuance, distribution and revocation, possibly key recovery, and potential interaction with other providers, e.g. with a registration authority (RA) and other CAs.

PKI Overview – Main Functions

- **SubjectRegistration**: during this process, a subject makes her/himself known to a RA/CA:
  - **Enrollment:** An applicant, e.g. *Alice,* may need to provide the following information (*depending on classes of certificates*):
    - Proof of *Alice*'s identity (email address, driving license, birth certificate, fingerprints, passport, NI number, etc).
    - Alice's public key, KU_{Alice}
  - **Authenticate applications**
    - share information with a third-party database.
    - personal appearance (use of Local Registration Authority).

PKI Overview – Main Functions

- KeyGeneration: a pair of crypto keys are generated **either by the subject or by the CA**, and the CA will certify the public key of the pair.
- CertificateIssuance (Certification): the CA issues a certificate for a subject's public key.
- CertificateVerification (proving the possession of credential): this is performed when a certificate is used to access a service or to perform a transaction.

PKI Overview – Main Functions

- CertificateRevocation: if the private key associated to the public key certified in the certificate is compromised or suspected of being compromised, then the certificate should be revoked.
- Cross-certification: is an operation to allow a pair of CAs to establish a trust relationship through the signing of each other's public keys in a certificate.

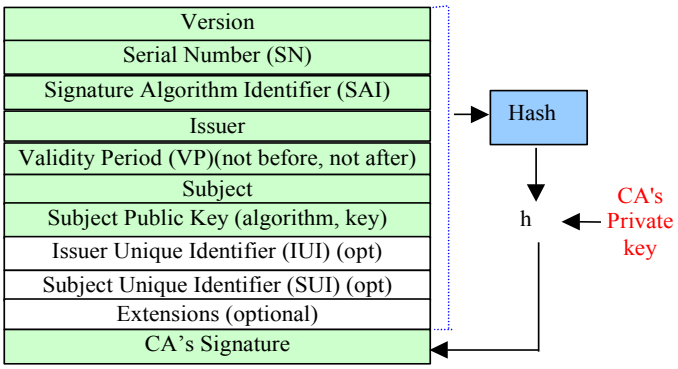
Part 2 Overview

- Digital Certificates

Digital Certificates

- Certification is a secure and scalable way of distributing public keys.
- A digital certificate (or *public-key certificate*, *digital ID*, *certificate*)
  - binds an entity's public key (+ one/more attributes) to its identity (the entity = person, hardware device, software process).
  - is digitally signed by the CA so you need CA's public key to verify the certificate.
  - its contents are application dependent, e.g. a certificate for secure email contains the entity's email address, a certificate for financial purpose may contain credit card number and credit limit, etc.

Digital Certificates - the X.509 v3 certificate format



Digital Certificates - the X.509 v3 certificate format

- Version: current values are v1, v2, v3.
- SN: unique identifier for each certificate generated by issuer (CA).
- SAI: identifying the algorithm, such as RSA or DSA, used by the CA to sign the certificate.
- Issuer: the issuer's name (X.500 'distinguished name').
- VP: a range of time when the certificate is valid.
- Subject: the subject's name (X.500 'distinguished name').
- SPK: the subject's public key and parameters, and the identifier of the algorithm with which the key is used.
- IUI: to allow the reuse of issuer names over time.
- SUI: to allow the reuse of subject names over time.
- Ext: provide a way to associate additional information for subjects, public keys, managing the certification hierarchy

Digital Certificates - An example

Version: 3

Serial Number (SN): 02:41:00:00:01

Signature Algorithm Identifier (SAI):
MD5 digest with RSA encryption

Issuer: C=US, O=RSA Data Security, Inc.,
OU=Secure Server Certification Authority

Validity Period (VP):
---Not Before Date: 16/5/96 12:00:00 AM
---Not After Date: 17/5/96 11:59:59 PM

Subject: C=GB, O=Manchester Univ,
OU=Computer Science

Subject Public Key (SPK):
Public key algorithm: RSA Encryption
Public key: Modulus: 00:92:…..(typically 200 digits)
Exponent: 65537

CA's Signature: 88:d1:…..

Part 3 Overview

- Certificate Revocation Lists (CRLs)

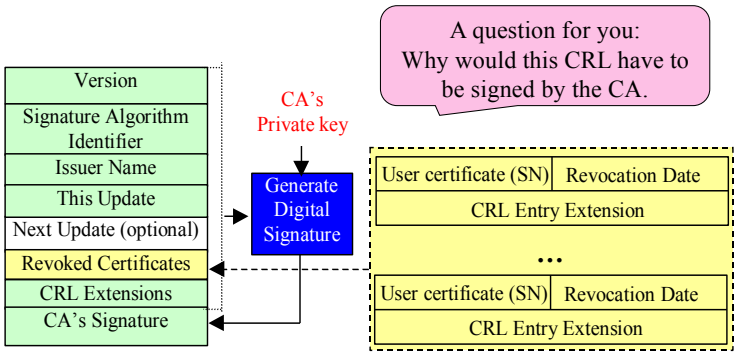
CRLs – What is it and why do we need it?

- CRL is a mechanism to let the world know that certificates are no longer valid. It is a black list of revoked certificates (i.e. prematurely terminated certificates).
- Reasons for revocation include:
  - The corresponding private key has been compromised.
  - CA may have been compromised.
  - Subject's affiliation has changed.
  - Key/certificate no longer needed.
  - …
- Required to reduce
  - risk of impersonation attacks.
  - risk of repudiation attacks.

CRLs – How to revoke it?

- A CRL is a data structure, digitally signed by the issuing CA, containing:
  - date and time of the CRL publications.
  - name of the issuing CA.
  - serial numbers of all the revoked certificates.

CRLs - X.509v2 CRL format



CRLs - X.509v2 CRL format

- Version: v2 should be used if any extension field are present. Otherwise, it can be omitted.
- Issuer Name: the entity that issued and signed the CRL.
- This Update: the date/time of issue of this CRL.
- Next Update: the date/time of issue of next CRL. The next CRL could be issued prior to, but not after, the indicated date.
- User Certificate SN: certificate serial number of a revoked certificate.
- Revocation Date: the effective date of a revocation.
- Extension: X.509 v2 CRL Entry Extension fields have the same sub-fields as X.509 v3 certificates.

CRLs – Deployment issues

- Using CRL is not that straightforward
  - The issuing CA needs to keep the CRL up-to-date.
  - A certificate-using application should obtain the most recent CRL and ensure that the certificate serial number is not on the CRL list; in other words, a certificate is said to be valid *iif* the following verifications are positive:
    - It has a valid CA signature,
    - It has not expired, and
    - It is not listed in the CA's most recent CRL.
  - There are some scalability issues.

Part 4 Overview

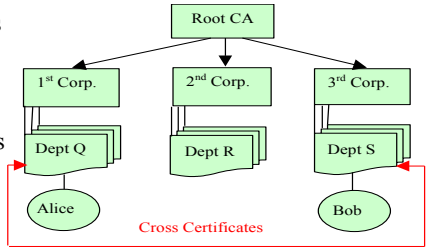
- Certificate Hierarchies
  - About how multiple CAs are organised/used
  - Two trust models
    - Top-down Certificate Hierarchy (Hierarchical Trust Model); used in X.509 PKI
    - Bottom-up Certificate Hierarchy (Peer-to-Peer Trust Model or web-of-trust); used in PGP
- Conclusions

Top-down Certificate Hierarchy

- In most cases, we use more than one CAs, as using one root key to sign certificates
  - is too risky if that one key is compromised.
  - is not scalable when user base is large.
- In some cases, certificate managements may resemble the management structure of an organisation, as depicted in the next slide.
- Certificate hierarchy
  - Start with a root CA (trust anchor) with a root cert/key pair (root-public-key, root-private-key).
  - **Delegate** signing power to subordinate CAs (create more key pairs, sign their public keys with root-private-key, ...)
- The fact that one authority, CA_U, signs on another authority CA_W's cert, $Sign_U(Cert_W)$, signifies that CA_U trusts CA_W.

Top-down Certificate Hierarchy

- RootCA generates certificates for intermediate CAs.
- Intermediate CAs generate certificates for the leaf CAs.
- Leaf CAs generate certificates for the end-entities (users, devices, and applications).



- Alice's Certificate Chain:
  $\{CERT_{Alice}\}S_{DeptQ}+\{CERT_{DeptQ}\}S_{1stCorp}+\{CERT_{1stCorp}\}S_{RootCA}$
- If Bob wishes to authenticate a message signed by Alice, he can proceed 'up' the certificate chain until he finds a certificate he can trust.

Top-down Certificate Hierarchy

- Validating a cert possibly involves validating a chain of certs (called chain of trust).
  - verify all the digital certificates, including the signatures signed by all subordinate CAs in a bottom-up manner until you reach the root CA's signature, or until you reach a subordinate CA that you can trust.

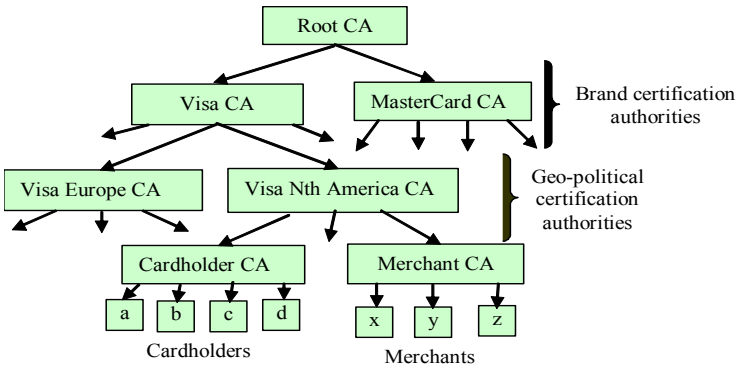
Top-down Certificate Hierarchy - Cross certification

- In this example, the 3rd Corp's Dept S has certified the 1st Corp's Dept Q.
- So, Alice's Certification Chain with **cross certification** is:
  $\{CERT_{Alice}\}S_{DeptQ} + \{CERT_{DeptQ}\}S_{1stCorp} + \{CERT_{DeptQ}\}S_{DeptS} + \{CERT_{1stCorp}\}S_{RootCA} + \{CERT_{DeptS}\}S_{3rdCorp} + \{CERT_{3rdCorp}\}S_{RootCA}$
- Now Bob only has to go up Alice's Certificate Chain to find his dept's certificate.
- Cross certification provides efficient certificate verification.

Top-down Certificate Hierarchy

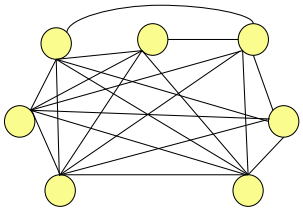
- Certificate types:
  - CA certificates: self-signed (a standalone or root CA), or issued by a superior CA within a hierarchy.
  - End-entity certificates: issued by a CA to subjects.
  - Cross-certification certificates: signed by a peer CA (independent CAs sign each other's certificates to establish peer-to-peer trust relationships).

Top-down Certificate Hierarchy - An Example (SET)



Bottom-up Certificate Hierarchy

- There is no trusted anchor among the CAs.
- Usually end-entities sign certificates for (other) entities they know (serve as CAs).
- Need a mechanism to assess the trust level of each CA/certificate.
- Used in the PGP (Pretty Good Privacy) solution.
- Potentially a fully meshed structure - not scalable.



Exercise Question – E7.1

(a) Investigate an on-line CA and find out what process or procedures that are necessary for you to acquire a public key certificate, how many classes of certificates and what each class can be used for.

(b) X.509 is a top-down approach to public key management. Investigate and describe a bottom-up approach to public key management.

Exercise Question – E7.2

Assuming that Alice has sent a signed message to Bob.
(i) Highlight the steps for verifying a digital certificate.
(ii) Highlight the steps Bob takes to verify the authenticity of the message from Alice.

Conclusions

- Digital certificates allows us to bind a public key to its rightful owner.
- This binding of key with identity allows us to solve the problem of how to distribution authentic public keys.
- Various PKI systems have been proposed - X509 works in a top-down manner.
- A *CA* is primarily responsible for issuing certificates and ensuring the validity of the certificates issued.

Topic 8: Key Management

Understand crypto key management issues and their distribution methods

Source: Stallings book, Chapter 14

Some slides are based on the slides by Lawrie Brown prepared for the text book

Overview

- Part 1
  - Key Management Issues
- Part 2
  - Symmetric Key Establishment: Diffie-Hellman (DH) algorithm/protocol
- Part 3
  - Symmetric Key Distribution using symmetric key encryption
- Part 4
  - Symmetric Key Distribution using public-key encryption
  - A summary and conclusion

Key Management Issues

- Key management is the hardest part of cryptography
  - How should keys be generated so that they can not be easily guessed?
  - How to securely store keys so that they can not be easily stolen?
  - How could keys be delivered to their intended recipients securely?
  - How could two entities agree on, or establish, a key securely?
  - How are keys revoked and replaced?
- For symmetrical ciphers - how to keep keys **secret**?
- For public-key ciphers - how to ensure public keys are **trust-worthy**?

Key Management Issues - Keys spaces

- Number of possible keys (key space) given various constraints:

	6-bytes	8-bytes
Lowercase letters(26)	$3.1*10^{8}$	$2.1*10^{11}$
Lowercase letters & digits (36)	$2.2*10^{9}$	$2.8*10^{12}$
Alphanumeric characters (62)	$5.7*10^{10}$	$2.2*10^{14}$
Printable characters (95)	$7.4*10^{11}$	$6.6*10^{15}$
ASCII characters (128)	$4.4*10^{12}$	$7.2*10^{16}$

Key Management Issues - Keys spaces

- Exhaustive search (assume 10^6 attempts/second):

	6-bytes	8-bytes
Lowercase letters(26)	5 minutes	2.4 days
Lowercase letters & digits (36)	36 minutes	33 days
Alphanumeric characters (62)	16 hours	6.9 years
Printable characters (95)	8.5 days	210 years
ASCII characters (128)	51 days	2300 years

Key Management Issues - Keys spaces

- **Main points**
  - Giving various constraints on the input string can greatly reduce the number of possible keys, therefore making ciphertexts much easier to break!
  - Computer power increases all the time ...
    - If you expect your keys to stand up against brute-force attacks for 10 years, plan accordingly.

Key Management Issues – Key generation

- When people choose their keys, they generally choose poor ones.
  - Which of these two keys are better (more difficult to guess) - *Barney1* or **9(hH/A*?
  - Remember: a smart brute-force attacker doesn't try all possible keys in numeric order; he will try the obvious ones first.
- Good keys are random numbers.

- What is a random number?
  - Given an integer, k>0, and a sequence of numbers, n_1, n_2, …, an observer can not predict n_k even if all of n_1, …, n_{k-1} are known.

Key Management Issues - Key generation

- Ordinary random number generation functions, e.g. `java.util.Random`, may not be good enough for this purpose.
- Use a cryptographically secure pseudo-random-number generator, e.g. `SecureRandom` class in `java.security` package, or a reliably random source.

- Physical sources of random numbers
  - Based on nondeterministic physical phenomena, e.g. atmospheric noise,
  - stock market data, etc.

Key Management Issues - Key generation

- Some pseudo-random numbers are generated from a strong mixing function
  - that takes two or more inputs having some randomness (e.g. CPU load, arrival times of network packets), but produces an output each bit of which depends on some nonlinear function of all the bits of the inputs.
  - Cryptographic hashing functions and encryption algorithms (e.g. MD5, SHA and DES) are all examples of the strong mixing function.

- For example, in a UNIX system, you may use the process state at a given time (date ; ps gaux) as the input to a MD5 function to generate a pseudorandom number, where 'ps gaux' lists all the information about all the processes on the system.

Key Management Issues - Key storage

- You must protect the key to the same degree as all the data it encrypts.
  - Why would one bother to go through all the trouble of trying to break the cipher system if he can recover the key because of sloppy key storage procedures?
  - Why would one spend $10 million building a cryptanalysis machine if he could spend $1000 bribing a clerk?

Key Management Issues - Key storage

- Attackers may defeat access control mechanisms, so encrypt the file containing key
  - Ideally, a key should never appear unencrypted outside the encryption device.
  - Try not to store your key on a medium connected to the network.

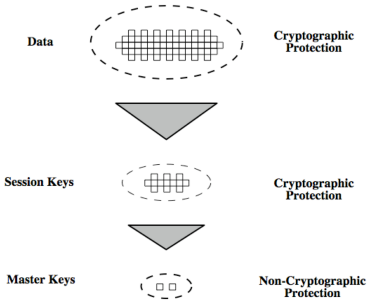
- Key may be resident in memory, so attackers may be able to read if they could get access to the machine
  - Use a physical token to store the key (e.g. a smart card) and protect the token with a PIN number.
  - Card can be stolen, so splitting a key into two halves, store
    - one half in the machine, and
    - another half in the card.
- Splitting a key, K, into two halves (k_1, k_2): $k_2 = k_1$ xor K

Part 2 Overview

- Symmetric Key (Shared Secret) Establishment:
  - Diffie-Hellman (DH) protocol

Key Hierarchy

- Typically have a key hierarchy
  - Master key/secret (key encryption key)
    - used to establish/distribute session keys
  - Session key (data encryption key)
    - used to encrypt data/message
    - for one logical session only



Session Keys

- More often a symmetric key is used, more likely it may be compromised.
- Generate and use a symmetric (secret) key for one session only → session key.
- Using different session keys in different sessions can
  - limit available ciphertexts for cryptanalysis.
  - limit exposure (both in time period and amount of data) in an event of key compromise.
- To avoid long-term storage of a large number of secret keys, we only generate them when they are needed.

Session Key Establishment

- Session key establishment solutions
  - Key agreement (exchange) protocols
    - A shared secret (master or session secret) is derived by the parties as a function of information contributed by each, such that no party can predetermine the resulting value - Diffie-Hellman (DH) protocol.
  - Key transportation protocols
    - Without any use of a public-key cipher
      - Session keys are generated and distributed using symmetric-key cipher and with the help of a third party - the Needham-Schroeder protocol.
    - With the use of a public-key cipher
      - One party creates a secret value (session key), and securely transfers it to the other party using the recipient's public key.

Session Key Establishment

- There are other issues that should be considered
  - Key secrecy and entity/key authentication
    - Assurance: no other party (outsiders - apart from the entities involved) could gain access to the established session key.
    - The session key is established with the intended entities.
    - Key confirmation: asking the other entity (possibly unidentified) to demonstrate that he has the knowledge of the key by
      - producing a one-way hash value of the key; or
      - encrypting some known data (e.g. nonce) with the key.
  - Key freshness
    - Assurance: the key is fresh, i.e. not used before.

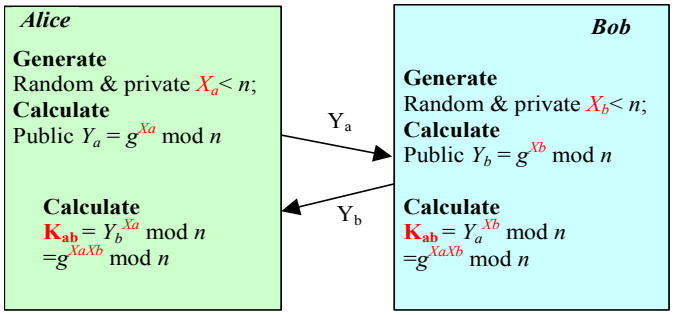
Diffie-Hellman Algorithm/Protocol

- DH was the 1st public-key algorithm ever invented - back in 1976.
- DH key exchange protocol allows two parties who have never met before to exchange messages in public and collectively generate a key that is private to them, and none of the parties could predetermine the key.
- Its security is based on the difficulty of calculating discrete logarithms in a finite field.
  - Given integers y and g and prime number n, compute x such that $y = g^x \bmod n$.
  - This is computationally infeasible if n is sufficiently large.

Diffie-Hellman Algorithm/Protocol

- Assuming two parties, *Alice* and *Bob,* take part in the exchange.
- Initial condition
  - *Alice* and *Bob* agree on two large integers, ***g*** and ***n***;
  - *n* - prime number that serves as the modulus.
  - *g* - random number that serves as the basis, with $1<g<n$.
  - *g* and *n* do not have to be secret.
- Definition
  - *Alice* has private key X_a, and public key Y_a.
  - *Bob* has private key X_b, and public key Y_b.

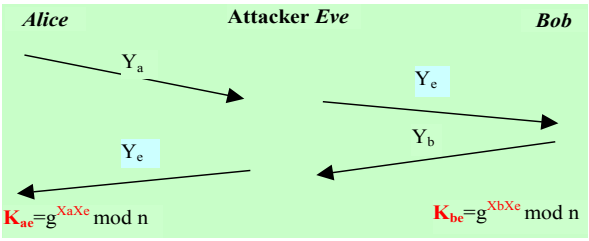
Diffie-Hellman Algorithm/Protocol



Diffie-Hellman Protocol

- It resists passive attacks such as eavesdropper, as calculating a discrete logarithm is a computationally hard problem.
- There is one problem - neither party knows who it shares the secret with! So it is vulnerable to active, man-in-the-middle attacks, as to be illustrated shortly.

Diffie-Hellman Protocol - Man-in-the-middle attack



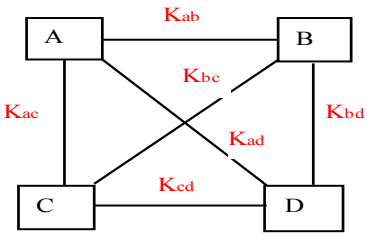
- *Alice* (*Bob*) thought she shares a key with *Bob* (*Alice*), but actually with *Eve*.
- So the attacker *Eve* can intercept and read any messages encrypted without been detected by *Alice* and *Bob* .

Part 3 Overview

- Symmetric key distribution using symmetric key encryption - Needham-Schroeder Protocol.
- This protocol is widely used in single sign on (SSO) solutions, e.g. window domain authentication, Kerberos.

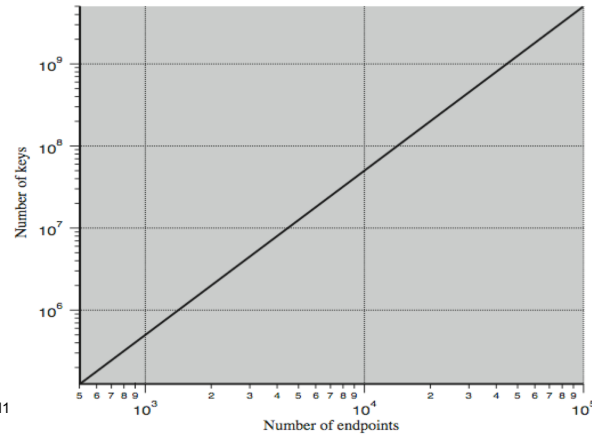
Distribution without use of PKC - Approach One

- *Approach One*: Given *n* users (parties/nodes) to communicate to each other, the system needs $n(n-1)/2$ keys.
- As *n* increases, the number of keys becomes untenable for everyone.
- The n^2 problem!



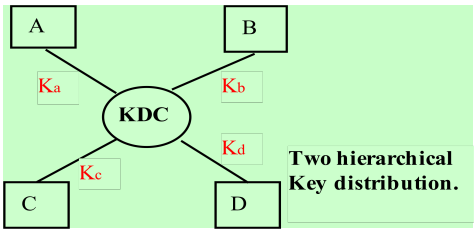
One hierarchical Key distribution.

Distribution without use of PKC - Scalability problem



Distribution without use of PKC - Approach Two

- *Approach Two:* use a key distribution centre (*KDC*) or security server.
  - A key hierarchy, e.g. two hierarchical approach - *master keys* (*long-term keys*) and *session keys* (*valid just for one session*).



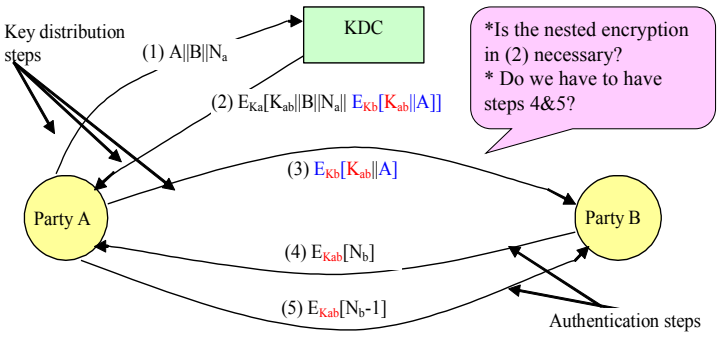
Distribution without use of PKC – Approach Two

- A unique master key, shared between a pair of user/*KDC*, is for session key distribution.
- A session key is to secure a particular session.
- Benefit of using Approach Two
  - Reduces the scale of the problem - reduces the n^2 problem to an n problem, thus making the system more scalable.
- But:
  - The need to trust the intermediaries - KDC.
    - KDC has enough information to impersonate anyone to anyone. If it is compromised, all the resources in the system are vulnerable.
  - KDC is a single point of failure.
  - KDC may be a performance bottleneck.

Distribution without use of PKC - Needham-Schroeder Protocol

- The Needham-Schroeder is a key distribution protocol.
- It uses the Approach two. That is:
  - both parties, *A* and *B*, shares a secret key with the KDC, K_a and K_b;
  - *A* and *B* wishes to establish a secure communication channel, i.e. establish a shared one-time session key K_{ab}, for use between *A* and *B* in this session.
- N_a, N_b are nonces (random challenges), generated by *A* and *B* respectively, to keep messages fresh.

Distribution without use of PKC - Needham-Schroeder Protocol



Distribution without use of PKC - Needham-Schroeder Protocol

(1) A sends a request to *KDC* for a session key to establish a secure channel with *B*.
(2) KDC generate a random number K_{ab}, and replies with the response containing
- session key K_{ab}.
- original request enables A matching the response with the request.
- an item (the session key and A's identity) which only *B* can view.

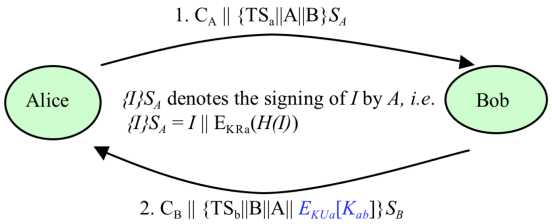
(3) A forwards the item to *B*.
At this point, the session key is securely delivered to A and B, and they may begin secure communication.
(4) B sends a nonce N_b to *A* encrypted using the new session key.
(5) A responds with N_b-1.
Steps (4) & (5) assure B that the message received in (3) was not a replay, i.e. to authenticate A.

Part 4 Overview

- Symmetric Key Distribution using public-key encryption
- A summary and conclusion

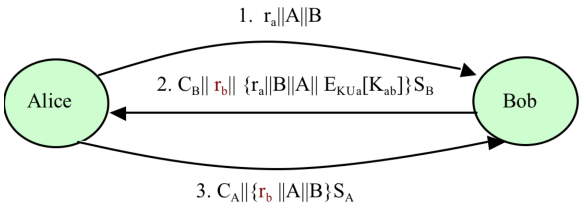
Distribution using PKC – Two passes

- Secret key distribution with mutual authentication using public key cryptosystem + timestamps.



Distribution using PKC (2) - Three passes

- Symmetrical key distribution with mutual authentication using digital signatures + nonces (random numbers).



A summary of secret key establishment protocols

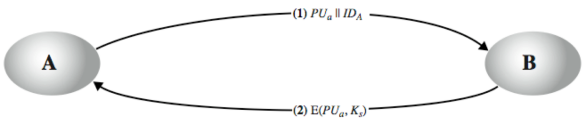
Protocols	ThirdParty	Timestamps	EntityAuth	messages
Diffie-Hellman	No	No	None	2
Needham-Schroeder protocol	KDC (online)	No	Symmetric encryption	5
X.509 (2 pass)	CA (offline)	Yes	mutual	2
X.509 (3 pass)	CA (offline)	No, but with nonce	mutual	3

Exercise Question – E8.1

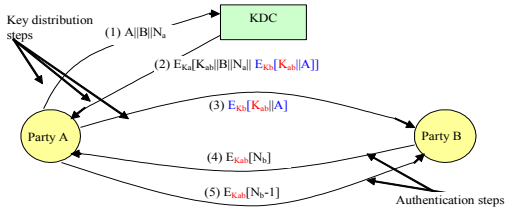
Assuming that Alice is to send a message, *M*, to Bob. *M* is encrypted with a shared key established using the DH protocol. Explain whether Eve could access this message *M*. If so, explain how, and propose a solution to address this vulnerability.

Exercise Question – E8.2

- The following is an extremely simple protocol proposed for symmetric key distribution. It is assumed that A and B has never met before (or there is no key established prior to this communication).
  - Identify as many problems/flaws as you can.
  - Modify the protocol to fix the problems/flaws you have identified.



Exercise Question – E8.3



This is the Needham-Schroeder protocol. Answer the following questions:

i. What are the benefits for A to forward the session key to B (i.e. step 3), rather than letting KDC to directly send the session key to B?

ii. TRY to identify two application areas of the Needham-Schroeder protocol and to elaborate the benefits of using the Needham-Schroeder protocol in these application areas.

Conclusions

- Key management encompasses a number of critical issues to the effective use of cryptosystems.
- A number of protocols exist to support symmetrical key distribution and agreement.
  - Key transport protocols
    - One party creates or otherwise obtains a secret value, and securely transfers it to the other party.
  - Key agreement protocols
    - A shared secret is derived by the parties using information contributed by each, such that no party can predetermine the resulting value.
- Key agreement/distribution protocols can be vulnerable to security attacks, such as the man-in-the-middle and replay attacks, so they should be used with care.

Topic 9: Entity Authentication

Apply authentication techniques to counter impersonation or masquerading attacks

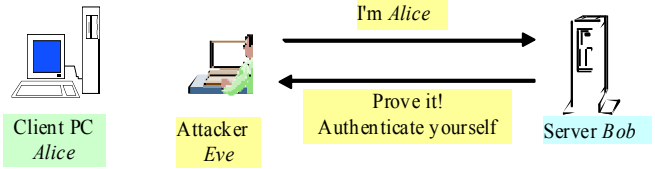
Source: Stallings book, Chapter 15 covers some of the content
An excellent document on e-authentication: NIST Special Publication 800-63-2: Electronic Authentication Guideline available here at:
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf

Overview

- Part 1
  - Authentication Overview
  - Password-based Authentication in General
- Part 2
  - Unix authN Solution
- Part 3
  - Challenge-Response (C/R) AuthN Protocols
  - Token-based Authentication
- Part 4
  - Enterprise-wide Authentication (SSO – Single Sign On)
  - Conclusions

Authentication Overview - Why do we need it

- Authentication is the process of verifying a claimed **identity.**
- If the communication takes place over a network, how could *Bob* be assured that the person claiming to be *Alice* really is *Alice*? If *Bob* is a server, an attacker may be able to log in as *Alice* to access data and services, or to use her account to launch further attacks.



Authentication Overview - What it is for

- **Authentication**
  - User identification/authentication or entity authentication
    - Mutual authentication
      - Who the user is?
      - Which system? – you could talk to anybody
    - The user identity is a parameter in access control decisions - authorisation
    - The user identity is recorded when logging security-relevant events in an audit trail – accounting
    - This is the so called AAA services
  - Message authentication - *we did this already*!
    - The message is from the source it claims to be.
    - The message has not been altered or replayed.

Authentication Overview - Methods

- Methods for user identification/authentication:
  - Where you are (location authentication - physical location/specific terminal, e.g. based on IP addresses).
  - Something you know (passwords, PIN).
  - Something you have (keys – soft tokens, and hard tokens (smart cards) - may require special hardware).
  - Something you are (biometrics - fingerprint matching, voice recognition, face recognition, iris scanning, etc - require special hardware).
  - Combined (or multiple) methods may be used for a higher level of assurance.

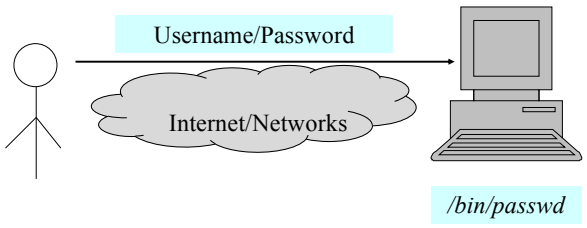
Itself can not support e-authentication

Authentication Overview - Prominent schemes

- Client-server authentication solutions
  - Password-based authentication.
  - Smart-card-based (token-based) authentication.
    - Symmetric key based
    - PKI based - digital signatures and public keys - X.509 certificates.
- Enterprise-wide authentication solutions (the issue of single-sign-on)
  - Kerberos (a password centric solution).
  - RADIUS (a centralised AAA service).
- Shibboleth (authenticating access to multiple enterprises/organisations) – outside the scope of this module.
- Different authentication schemes provide different levels of assurance.
- There is a trade-off between the level of security vs complexity vs cost.

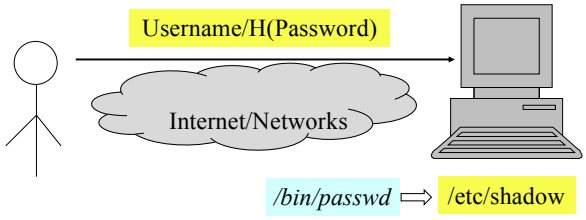
Password-based authentication – Plaintext PW method

- Plaintext passwords are transmitted and stored in the server.
- What are the conditions under which this method can be used securely? What/where are the threats (what is the threat model)?

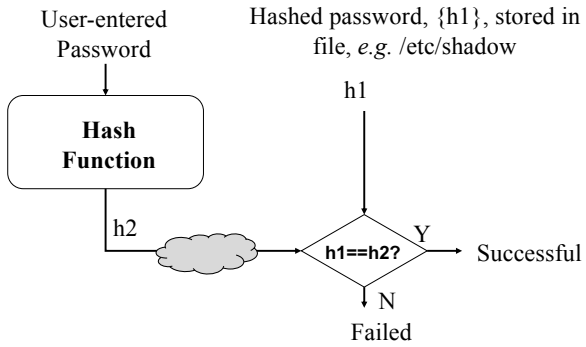


Password-based authentication – Hashed PW method

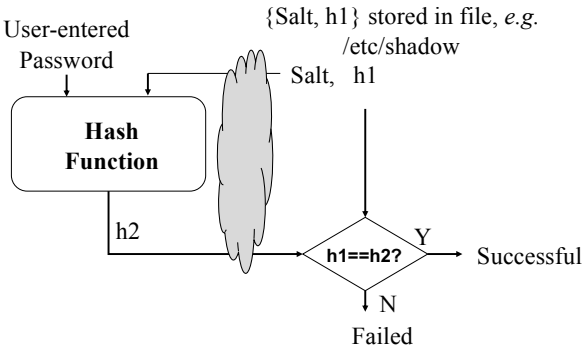
- Hashed passwords are transmitted and stored in the server, but in a root directory.
- What are the conditions under which this method can be used securely? What/where are the threats (what is the threat model)?



Password-based authentication – Hashed PW method



Password-based authentication – Hashed PW with Salt method



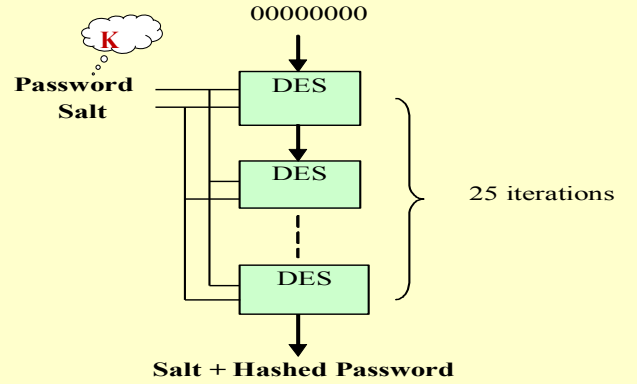
Part 2 Overview

- Unix authN Solution

Unix AuthN Solution – Crypt() algorithm

- Unix system uses the 'Hashed PW with Salt' method.
- Original Hash function is the **Crypt()** algorithm
  - modified based on the DES algorithm.
  - 8 character password form 56-bit key (for each character, 1 bit for parity check and 7 bits for ASCII coding).
  - 12-bit salt to
    - perturb the DES algorithm, so that DES chip can not be used to (dictionary) guess the passwords.
    - make precompiled dictionary attacks harder (by a factor of 4,096).
    - prevent an identical password from producing the same encrypted password.
  - The 64 bits output are unpacked into a string of 11 printable characters, called *the encrypted password*.

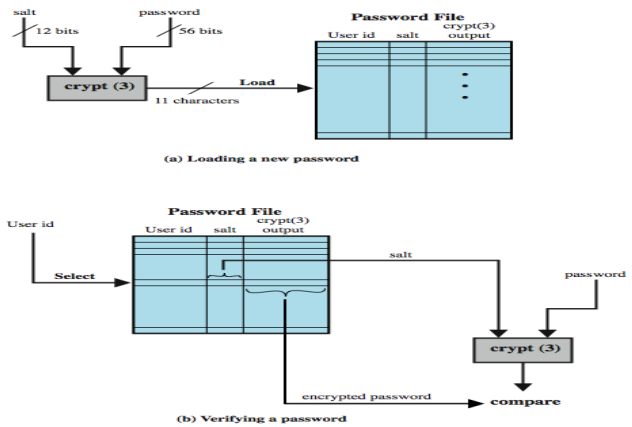
Unix AuthN Solution - Crypt() algorithm



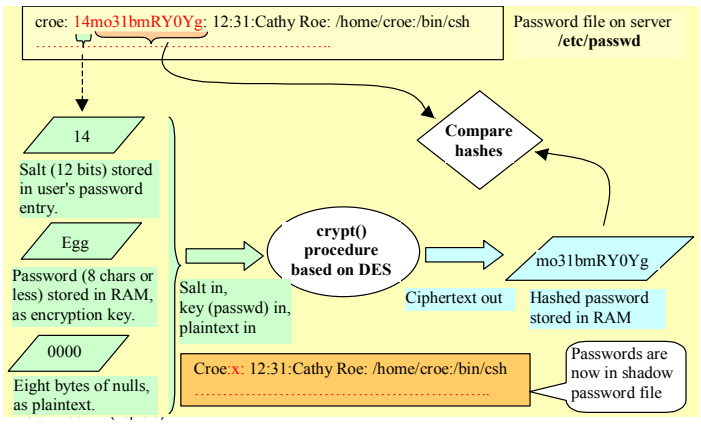
Unix AuthN Solution – AuthN process

- When a user have an account created or changes the password, the */bin/passwd* program
  - selects a salt based on the time of day; the salt is converted into a two-character string and is stored along with the encrypted password.
  - the password is used as the encryption key to encrypt a block of zero bits using *crypt( )* to generate the encrypted password.
- When a user tries to log in, the program */bin/login* takes the password the user typed, and the salt from the password file, to generate a fresh encrypted password, and compares the newly generated one with the one stored in the server. If the two encrypted results match, the system lets you in.

Unix AuthN Solution – AuthN process



Unix AuthN Solution - AuthN process



Unix AuthN Solution – Shadow Password File

- The Unix password file
  - is */etc/passwd*. Each entry in the file is for one account and has several fields separated by colons:
    - User Name (croe): the account name.
    - Password (14mo31bmRY0Yg): the hashed password (mo31bmRY0Yg) preceded by a salt value (14) to be used with the password. Salts make the password
      - more difficulty to guess, and
      - the hashing algorithm slower!
    - User ID (UID=12): a number assigned to this user name for system use in identifying the account.
    - Group ID (GID = 31): a number for the user's group.
    - Home Directory (/home/croe).
    - Shell (/bin/csh): the user's default shell program.

Unix AuthN Solution – Shadow Password File

- **Problem**: /etc/passwd file need to be accessed by processes, so if anyone could get access to the file, then s/he could crack on the passwords at his/er spare time.
- **Countermeasure:** a shadow password file, /etc/shadow, is used, which contains the encrypted passwords, and is put in a root account; put an x (or other placeholder) in the original /etc/passwd file. That is,

/etc/passwd file contains**:**
An example: User1:x:9111:9201:user1:/home/user1:/bin/bash
Meaning: UserName:x *(indicate that the password is stored in the /etc/shadow file)*
:UserID:GroupID:FullName:HomeDirectory:UserShell.

/etc/shadow file contains**:**
An example: User1:1/uTQhcV4$2E…../:13030:0:99999:7:::
Meaning:
UserName:*hashedPassword*:passwdLastChanged:PasswdMayBeChanged:PasswdMustBeChanged:PasswdChangeWarning:DisableAccount:DisabledSince:Reserved.

Unix AuthN Solution - Improved Unix Solution

- But attacks still possible - if you run some software processes with root privileges …, and if the attacker can take over such a program ...

- More problem: An attacker can eavesdrop on a network to get your login ID and encrypted password and later replays (re-send) it to gain access to the system - the replay attack.
  - To perform this attack, the *attacker needs* to
    - eavesdrop on the network (or access to the password file).
    - modify the client/logon software so that it does not encrypt the encrypted -password, but rather replay it directly;
  - *Usually we assume* that
    - the LAN is secure, i.e. eavesdropping can be noticed!
    - You do not bring your own client software in!
  - So we tend to overlook this problem in a LAN environment.

Unix AuthN Solution – Current Unix Solution

- Many *NIX systems now
  - use hash functions such as MD5
  - allows longer password length and longer encrypted passwords
  - use longer salt
  - use PAM (Pluggable Authentication Module) – a flexible way to support the use of many different authentication methods

Part 3 Overview

- Challenge-Response (C/R) AuthN Protocols
- Token-based authentication

C-R AuthN Protocols

- A protocol specifies a set of rules governing the interactions among two/more entities to accomplish a given task.

- For example, an ATM Cash Withdrawal Protocol:

1. Insert ATM card
2. Enter your PIN and make a transaction request
3. PIN Correct?
   Yes – Next step action
   No - Try again (repeated fails will lead to taking your card in)

C-R AuthN Protocols

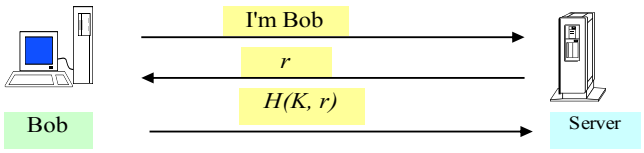
- A protocol consists of
  - Protocol messages facilitating the communications between the entities.
  - Operations for performing some defined actions (algorithms/functions), including generating/verifying a protocol message.
- An AuthN protocol specifies a set of rules governing the interactions among two/more entities in order to accomplish an authN task.

C-R AuthN Protocols

- They vary depending on a number of factors, e.g.
  - What is the threat model?
  - Should it support mutual authentication?
  - Should it establish a session key?
  - What key(s)/functions are used? - public keys, symmetric keys, passwords, hash functions, block ciphers, …
  - Any performance requirement?
  - Any privacy requirement?
  - Should scalability be considered?
  - Any other factors in the underlying environment that should be considered? and so on.

C/R AuthN Protocols – A C/R protocol for OTP authN

- A C/R protocol (Challenge-Handshake Authentication Protocol, CHAP) is based on a secret (password/symmetric key/private key).
- Demonstrate the knowledge of a secret without revealing the secret.
- Hashed value in the response serves as an OTP (OneTimePassword).
- Commonly used between a client and an infrastructure edge device, e.g. remote access server, a VPN server.

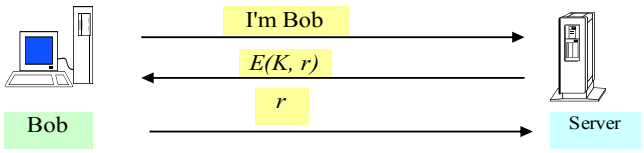


r is a nonce, *K* is a shared key, and H is a hash function

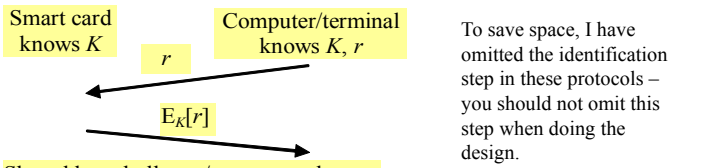
MS-CHAP: identical to CHAP except for using password to replace the key K.

C/R AuthN Protocols – More protocol variants

- *The challenge is an encrypted item and the user is challenged to perform* the decryption, demonstrating the knowledge of the secret key.
- K can be a symmetric key, a key derived from the user's password, or Bob's public key.

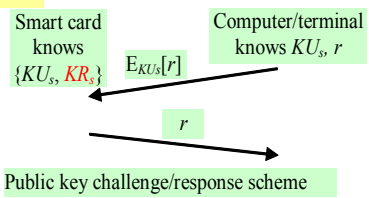


C/R AuthN Protocols – More protocol variants



To save space, I have omitted the identification step in these protocols – you should not omit this step when doing the design.

If smart card returns a correct value (or *r)*, it means: (a) the card was at present, (b) correct PIN was entered (assuming PIN is used).



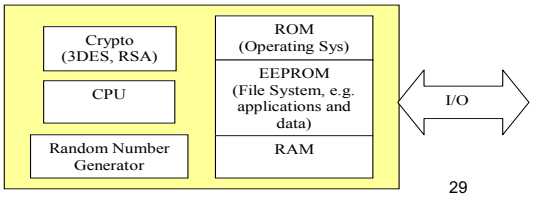
Token-based Authentication

- Various forms
  - PIN protected memory card
    - Enter PIN to get the password
  - PIN and smart phone based token
    - Google authentication
  - Smart cards with embedded CPU and memory
    - Carries conversation with a terminal (a computer or a card reader)
    - Card reader creates a random challenge
    - Enter PIN to encrypt/decrypt the challenge using the card

Smart Cards – Types

EPROM - Electrically Programmable R O M
EEPROM - Electrically Erasable Programmable R O M

- Memory cards
  - Contain EEPROM and ROM.
  - EEPROM holds data that changes with time.
  - ROM holds card number, card-holder name.
- Chip and PIN cards
  - On-board memory (RAM, ROM, and EEPROM).
  - On-board Crypto coprocessors for performing crypto operations, random number generator for key generations, and intelligence for making decisions.



Smart Cards – Types

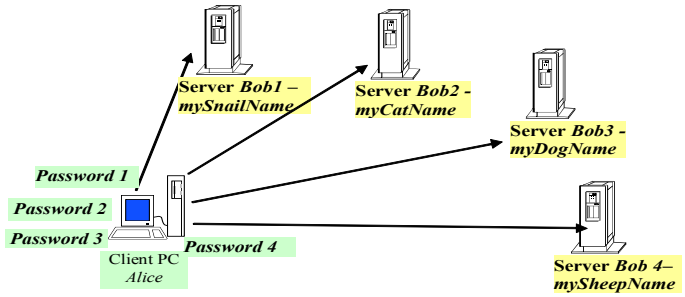
  - Provide robust security – more difficult to tamper with and uneconomical to counterfeit.
  - Supports multiple applications (e.g. ePayment, ePassport, eIDs, Health, Transport, etc).
- Interface
  - Contact
  - Contactless (using RFID).
  - Hybrid
- Operating Systems (OS)
  - Java Cards
  - Multos
  - SLCOS
  - etc

Part 4 Overview

- Enterprise-wide authentication
- Conclusions

Enterprise Authentication

- The task of authentication is imposed on all the servers.
- For users, either the same password for accessing all the servers, or different passwords for different servers.

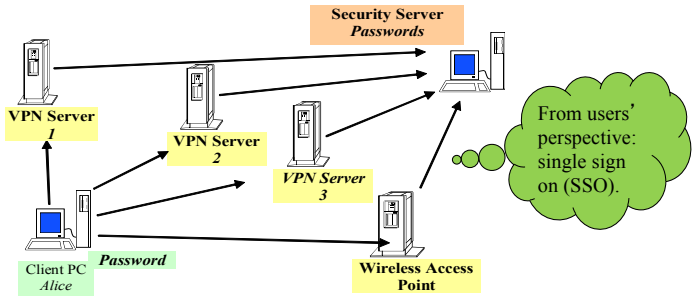


Enterprise Authentication

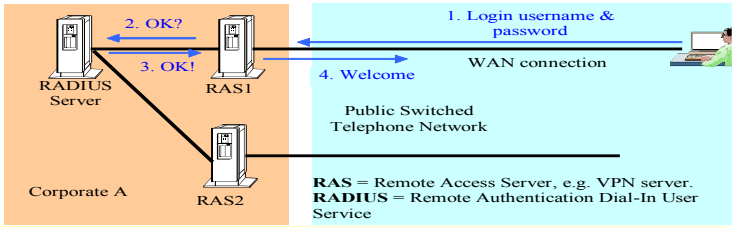
- Centralised authentication
  - **One central authority** (**security server or authentication server**) in **each domain/organisation** performs the task of authentication
    - One password per user;
    - Service servers do not need to handle authentication;
    - More secure as authentication servers only interact with service servers.
    - Unified enforcement of domain/organisation-wide security policies, e.g. AAA services.
- A number of systems exist, e.g.
  - Radius - Remote authentication for dial-in user service
    - Initially designed for remote dial-in, but now extended to handle enterprise-wide AAA services…..
  - Kerberos

Enterprise authentication

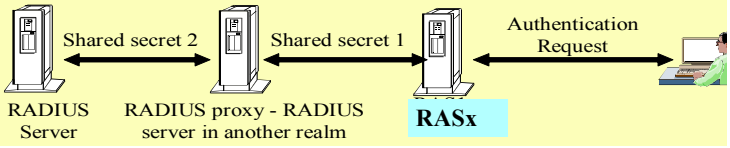
- The security server is responsible for providing AAA services to edge devices, e.g. VPN servers and wireless access points.



Enterprise authentication – RADIUS

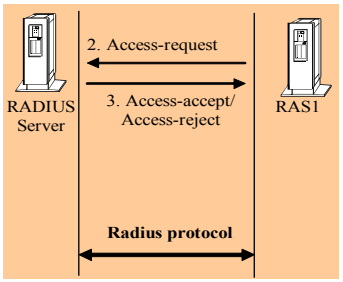


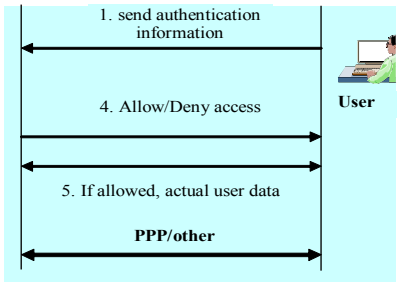
The model of trust: RADIUS uses shared secrets to achieve mutual authentication and confidential communication between RADIUS entities.



Enterprise authentication - RADIUS

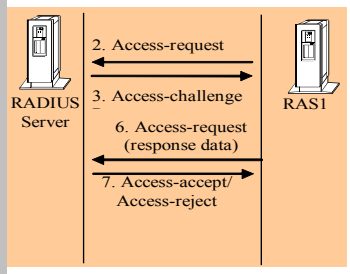
- Simple user authentication and authorisation using RADIUS
- PPP=Point-to-Point Protocol

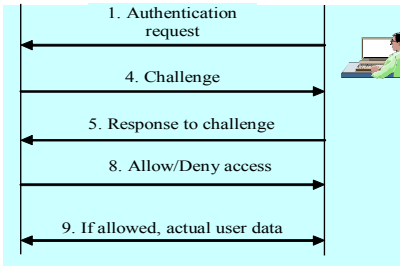




Enterprise authentication - RADIUS

- Challenge-response authentication and authorisation using RADIUS



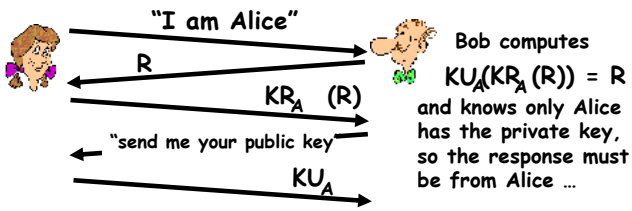


Enterprise authentication - RADIUS

- RADIUS protocol:
  - Client forwards the user access request to a RADIUS server
  - Server
    - Replies with *reject access* or *allow access* based upon a user supplied password/credential.
    - Challenge (when challenge-response protocol is used, e.g. CHAP).
    - If challenge-response is used, client forwards Challenge to the user, and the user sends their Response to the client that then forwards it to the server.
    - One RADIUS server may act as a client to another RADIUS server for consultation, etc.

Exercise Question – E9.1

Many authentication protocols are based on the use of a symmetric key/secret. What if that 'trust relationship' has not been established? Using nonce (random number) and a public-key cipher can be a solution. Is the following authentication protocol secure? Justify your answer.



Exercise Question – E9.2

- Design two different X.509 certificate based authentication protocols to support mutual authentication between two entities, Alice and Bob.

Conclusions (1/3)

- Passwords are the most basic authentication mechanism
  - The security level is only as good as the passwords you select.
  - They are vulnerable to guessing unless precautions ensure there is a large enough set of possible passwords and each potential one in the set is equally likely to be chosen.
  - Challenge-response techniques allow the system to vary the password therefore less vulnerable to guessing attacks; OTPs, an example of this technique, are particular effective against guessing and replaying attacks.
- Authentication can also be achieved with public-key cryptography for which public key certificates are needed - X.509 standard.

Conclusions (2/3)

- There are also other forms of authentication: biometrics measures physical characteristics of the user; location requires the verifier to determine the location of the user.
- In practice, some combination of these methods can be used. This depends on the resources available to the verifier and the user, the strength of the authentication required, and external factors such as lows and customs.
- System designers have to balance convenience and security. Ease-of-use is an important factor in IT systems. However, convenient practices may introduce new vulnerabilities.
- There are authentication issues when multiple systems or multiple sites are involved.

Conclusions (3/3) – Applications of PKC

- Confidentiality – PKC allows encryption, and therefore protection against unauthorised disclosure of info.
- Signatures and integrity – PKC can be used to generate a signed token for a message so that the recipient of the message can determine if the message is authentic, or has been modified.
- Signatures and non-repudiation – PKC signatures can be used to determine if a message has indeed been sent by a particular sender.
- Identification and authentication – PKC can be used to determine an identity of a user and a service.
- Timestamping - PKC can be used to generate a time stamp to certify that a particular message has indeed been sent at a particular time and date.

Topic 10: A Security System

Exploring real-life security solutions:
PGP (Pretty Good Privacy)

Sources:
1. https://en.wikipedia.org/wiki/Pretty_Good_Privacy
2. http://www.facweb.iitkgp.ac.in/~sourav/PGP.pdf

Overview

- Part 1
  - PGP overview
- Part 2
  - PGP services (PGP cryptographic functions)
  - PGP message format
- Part 3
  - PGP key rings and trust model
- Part 4
  - PGP message transmission and reception
  - Conclusions

PGP Overview – What it is for

- PGP is a general purpose application to protect files – files in storage and emails.
- There are two most notable security packages which can be used to protect emails:
  - PGP (Pretty Good Privacy)
  - S/MIME (Secure/Multipurpose Internet Mail Extension)
- The two packages essentially provide **the same security functionality**, with two major differences
  - PGP uses bottom-up approach to public key management (web-of-trust model), whereas S/MIME uses a top-down approach (PKI X.509).
  - PGP does not include the sender's public key with each message, rather it includes a key ID.

PGP Overview – Why I choose PGP

- S/MIME is recommended by NIST, while PGP widely used for personal e-mail and file protections.
- Based on well-known crypto algorithms.
- Documentation & source code are freely available.
- Can be used not just for email protections but also for protecting file storage.
- Uses a different trust model (from X.509), which has benefits but also problems.

PGP Overview – Email security problems

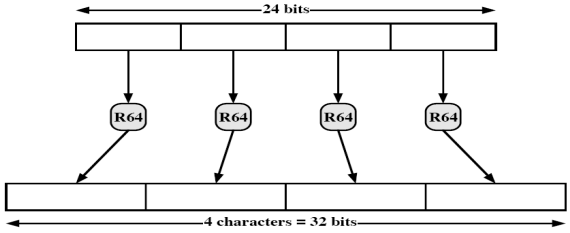
- Message interception
- Message interception and subsequent replay
- Message content modification
- Message origin modification
- Message content forgery (by an outsider or by a recipient)
- Message origin forgery (by an outsider or by a recipient)
- Denial of message transmission/origin (repudiation of transmission/origin)
- Denial of message reception (repudiation of receipt)

PGP Overview – Capabilities/Services

- Message confidentiality service (disclosure protection): hybrid encryption.
- Message integrity: digital signature.
- Origin authentication & non-repudiation of origin (if public keys certified): web-of-trust.
- E-mail compatibility (encoded character set is universally representable at all sites): radix-64 conversion
- Segmentation: segment/reassemble a long message
- Compression: compresses a message after applying the signature but before encryption
- Trust model (web-of-trust) for key management

PGP Overview - Compatibility

- Encrypted message may contain some arbitrary octets.
- Many email systems only allow the use of ASCII text.
- Radix-64 conversion is used to convert (table-map) each group of three octets of binary data into four printable ASCII characters.



PGP Services

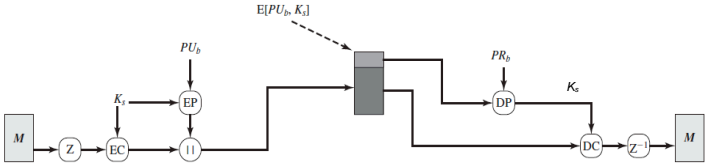
Functions/services	Algorithms Used	Description
Digital signature/ Message authentication	DSS/SHA, RSA/SHA (MD5, …)	Message (M) is hashed, timestamped and signed using DSS or RSA
Message encryption/ confidentiality	AES, CAST, IDEA, 3DES, …, with Diffie-Hellman, and RSA	M is encrypted using a one-time session key; session key is established using D-H or encrypted using RSA with recipient's public key.
Compression	ZIP	
E-Mail compatibility	Radix-64 conversion	Local form (e.g. 8-bit binary stream) ←-→ printable ASCII characters
Segmentation	-	Performs segmentation/reassembly to accommodate maximum message size limitations

Part 2 Overview

- PGP services (PGP cryptographic functions)
- PGP message format

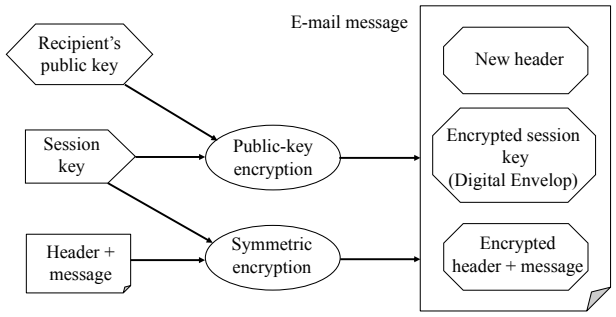
PGP Services – Confidentiality Only

- Message content is encrypted with a symmetric session key.
- The session key is encrypted with the recipient's public key.
- Notation: Z/Z^{-1} = compression/decompression, EC/DC=symmetric-key encryption/decryption, EP/DP=public-key encryption/decryption, '||'=concatenation



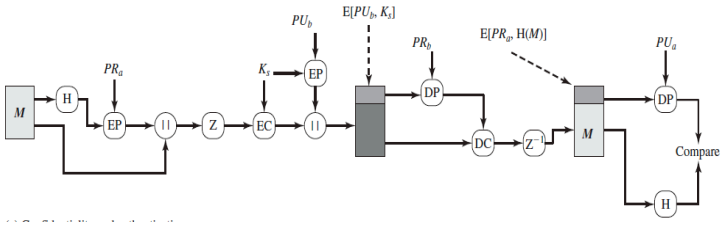
PGP Services – Confidentiality Only

- Message Confidentiality

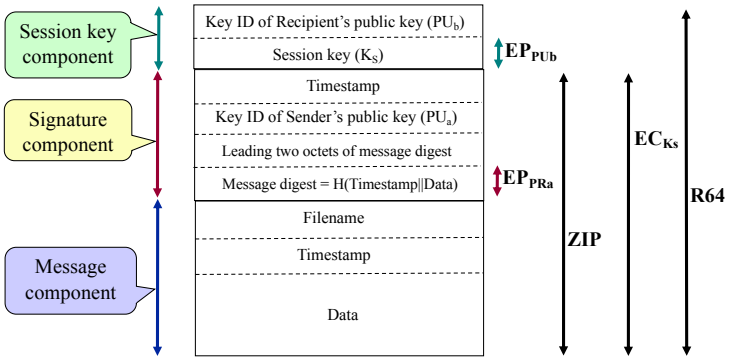


PGP Services – Confidentiality and Authentication

- The message is digitally signed for the sender authentication
- The message content is encrypted for its confidentiality



PGP Message Format



PGP Message Format

- Session key component: include the session key and the identifier of the recipient's public key that was used by the sender to encrypt the session key.
- Message component: include the actual data to be transmitted, as well as a file name and a timestamp that specifies the time of the message creation.

PGP Message Format

- Signature component:
  - Timestamp – when the signature was generated.
  - Message digest,160-bit SHA-1 digest, encrypted with the sender's private key. This signature is calculated over the signature timestamp concatenated with the data portion of the message component.
  - Leading two bytes of message digest is to enable the recipient to determine if the correct public key was used to decrypt the message digest for authentication by comparing this plaintext copy of the first two bytes with the first two bytes of the decrypted digest.

Part 3 Overview

- PGP key rings
- PGP trust model

Key Rings

- Key Identifiers (Key IDs)
  - Each user can have multiple public/private key pairs, so they need identifiers (IDs).
  - How could a recipient identify which public key should be used for an incoming message?
    - Send the public key with the message, or
    - Use a key identifier to identify a particular key in a key server.
  - PGP uses the latter approach, i.e. use a **user ID** plus a **key ID** to identify a public key.
  - Key ID = (KU_a mod 2^{64}), i.e. the least significant 64 bits of the public key.

Key Rings

- Each user maintains a pair of data structures (i.e. a pair of key rings) in his/her system:
  - A private-key ring to store the private/public key pairs owned by this user.
  - A public-key ring to store the public keys of other users known to this user.
- Both key rings can be indexed by either User ID or Key ID.
- The user's private key is encrypted using the symmetric algorithm, CAST-128 (128-bit key), and the key-encryption key is the first 128 bits of the hash code of a passphrase, *Pi, chosen by the user. That is,* the encrypted private key = $EC_{H(Pi)}[PR_i]$, where H is a hash function.

Private-Key Ring

- Used to store the public key & private key pairs owned by this user.
- A table of rows; each entry for one key pair containing:
  - Timestamp: when the key pair was generated.
  - Key ID: identifier of this key (index).
  - Public key: the public key for the entry.
  - Encrypted private key: the private key is encrypted using the hash value of a passphrase.
  - User ID: usually the user's email address; may be different for different key pairs (index).

Public-Key Ring

- Used to store public keys of other users.
- A table of rows; each entry for one key containing:
  - Timestamp: when the entry was generated.
  - Key ID: identifier of this key.
  - Public Key: the public key for this entry.
  - Owner Trust
  - User ID: identifier for the owner of this key.
  - Key Legitimacy
  - Signature: the signature signed on this certificate (i.e. the binding between this public key and its owner identified by 'User ID').
  - Signature Trust

General Structures of Private-/Public-Key Rings

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
…	…	…	…	…
T_i	PU_i mod 2^{64}	PU_i	$EC_{H(Pi)}[PR_i]$	User i
…	…	…	…	…

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature	Signature Trust
…	…	…	…	…	…	…	…
T_i	PU_i mod 2^{64}	PU_i	$Trust\_flag_i$	$User_i$	$Trust\_flag_i$		
…	…	…	…	…	…	…	…

* Field is used to index the tables.

PGP Trust Model - Web-of-Trust

- Trust is captured by the three attributes
  - **Owner Trust**: the degree to which this PGP user trusts the key's owner to sign public-key certificates for other users; the value is assigned by this user.
  - **Signature Trust**: the trust the signer (of the certificate) has on the identity of the user identified in the certificate; the value is assigned by the PGP system.
  - **Key Legitimacy**: the degree to which this PGP user trusts that this public key belongs to this owner. The higher the value, the stronger the binding between this 'User ID' and this 'Public Key'; the value is calculated by the PGP system based on some heuristics defined by the user based on the trust (signature trust) on the certificate or chain of certificates.

PGP Trust Model

- **Owner Trust**
  - Assigned by this PGP user (YOU).
  - Indicates the degree YOU trusts the key's owner to sign other public-key certificates.
  - For example, possible values:
    - *unknown user*
    - *partially trusted user*
    - *fully trusted user*

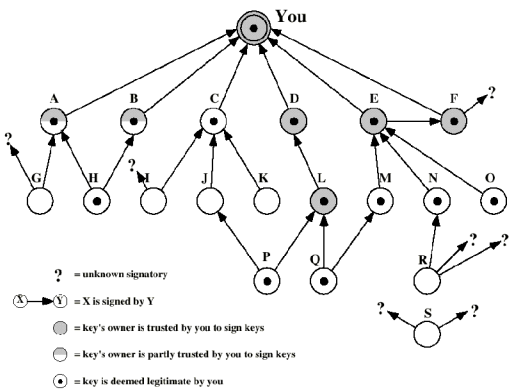
PGP Trust Model

- **Signature Trust**
  - This value is public on the key server along with the certificate signed.
  - It reflects the trust the signer (of the certificate) has on the identity of the user identified in the certificate. If Alice signs (i.e. certifies) Bob's key, this is the value she declares she puts trust in Bob's identity.

PGP Trust Model

- **Key Legitimacy**
  - Computed by the PGP system.
  - For example, to set full Key Legitimacy (Key Legitimacy = 1) for a pubic key, the public key should
    - be certified by at least one certificate with the value of Signature Trust being 'fully trusted',
    - OR be certified by at least two certificates with the value of Signature Trust being 'partially trusted'.

PGP Trust Model



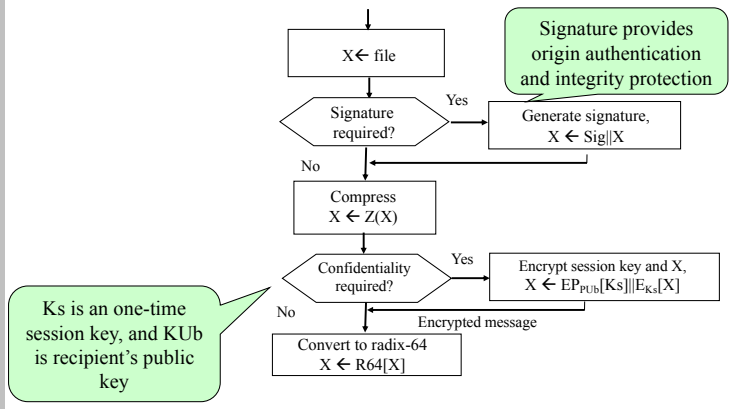
PGP Trust Model – Public key revocation

- A public key should be revoked, if
  - the private key corresponding to the public key is suspected to have been compromised, or
  - the owner no longer needs the certificate.
- This is done by the owner generating and disseminating a revocation certificate containing the public key to be revoked, which is signed with the corresponding private key.

Part 4 Overview

- PGP message transmission and reception
- Conclusions

PGP – Transmission of a PGP message



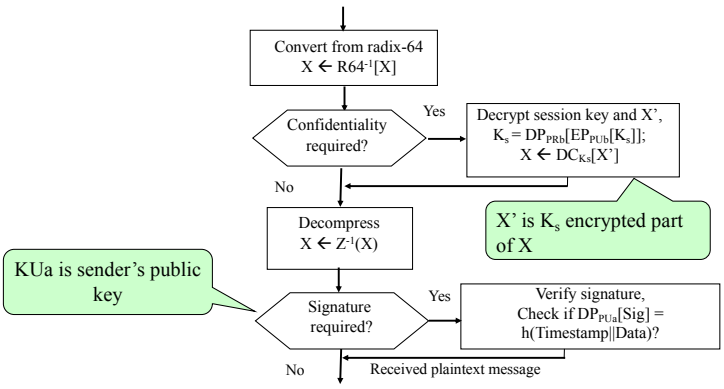
PGP – Generation of the signature component

- Signing a Message
- PGP retrieves the sender's private key from the private-key ring using the User ID as an index. If no User ID was supplied, retrieve the first private key.
- PGP prompts the user for a passphrase to recover the encrypted private key.
- The signature component of the message is constructed.

PGP – Generation of the session key component

- PGP generates a session key and encrypts the message using the session key.
- PGP retrieves the recipient's public key from the public-key ring using her User ID as an index.
- The session key component of the message is constructed.

PGP – Reception of a PGP message



PGP – Decryption of the incoming message

- PGP retrieves receiver's private key from the private-key ring, using the Key ID field in the session key component of the message as the index.
- PGP prompts the user for the passphrase to recover the private key.
- PGP recovers the session key and use it to decrypt the message.

PGP – Authentication verification of the incoming message

- PGP retrieves sender's public key from the public-key ring, using the Key ID field in the signature key component of the message as the index.
- PGP decrypts the signature to recover the message digest received.
- PGP computes a fresh message digest based on the message received and compares the two message digests.

Exercise Question – E10.1

In secure email systems, such as PGP (pretty good privacy), the order of message protection operations performed by a sending entity is to first sign the outgoing/email message, then compress the signed message, and finally encrypt the compressed message. Answer the following questions:

(i) What are the justifications (or benefits) for using this order of operations, and

(ii) Would this order of operations be suitable for all cases of applications?

(iii) Assuming that Alice sends Bob an email, M. Express the whole message that is sent by Alice using protocol notation format.

Exercise Question – E10.2

Contrast the two trust models in terms of their respective merits and weaknesses:

- PKI X.509
- PGP web-of-trust

Conclusions

- Security issues affect many aspects of our lives.
- There are many interesting security solutions out there for you to explore.
- Hope, through these exemplar security solutions, you could get a flavour of security problems, and methods/approaches to address the problems.

Topic 11: Access Control

Study mechanisms for controlling access to resources

Sources:

Computer Security by Matt Bishop.

Related information is also available on the Internet, e.g. http://csrc.nist.gov/staff/Kuhn/towards-std.pdf.

The RBAC slides are from Prof. Ravi Sandhu's slides; Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, "Role-Based Access Control Models." IEEE Computer, Volume 29, Number 2, February 1996, pages 38-47.

Overview

- Part 1
  - Basic concepts
- Part 2
  - Discretionary Access Control (DAC)
- Part 3
  - Mandatory Access Control (MAC)
- Part 4
  - RBAC (Role Based Access Control)
  - Conclusion

Access Control Basic Concepts

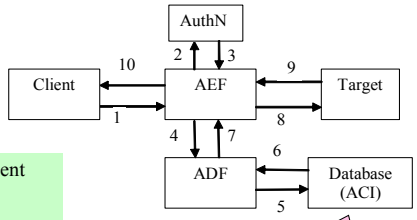
- An access control system is to
  - Keep the bad guys out.
  - Let the good guys in (who can read, who can modify, …).
  - To enforce a specified access control policy preventing unauthorised access to data, services or other resources.
- Resources may be in different forms and the control should be provided at multiple levels:
  - At the system level, e.g. security kernel in an operating system, database management systems (DBMS).
  - At the network level, e.g. firewalls.
  - At physical level, e.g. lockers, biometric scanning.

Access Control Basic Concepts

- Access control = AuthN + AuthZ.
  - Authentication (AuthN) establishes the identity of a subject (*who*).
  - Authorization (AuthZ) specifies and enforces that each object is accessed correctly and only by those that are allowed to do so
    - An **access control policy** specifies *who* (subject) can perform what access operations (access types/modes) on what (object).
    - These rules are enforced at run-time by an AuthZ Decision Engine.

Access Control Basic Concepts

Identification: Who I am.
AuthN: Prove it!
AuthZ: Decide if you have the right permission.



- AEF = Access control Enforcement Function (enforce AC decision);
- ADF = Access control Decision Function (grant/deny based on AC policy);
- ACI = Access Control Information (AC policy).
- Target = resources (data, services, etc)

A simple AC rule (non-RBAC):
<Alice, Write, FileA>
Another simple AC rule (RBAC):
<Alice, Staff>, <Staff, Write, FileA>

Access Control Basic Concepts

- A subject: an active entity requesting for resource access
  - The identifier (ID) of a user.
  - The ID of a process/software executing programs on behalf of a user.
  - A role - a group of users, a collection of privileges or a functional entity in an organisation, etc.
- An object: a passive entity and target of protection, e.g.
  - a file or directory of files, a data structure (e.g. a stack, inter-process messages, network packets).
  - a table of an operating system, instructions (especially privileged instructions).
  - memory, I/O device, host machine, private network, etc.

Access Control Basic Concepts

- Access operations (or access types/modes): permitted operations; resources dependent, e.g.
  - for file access: read, write, execute, append (does not imply read access), delete, …
  - for network access: grant, deny, redirect, …

Access Control Basic Concepts

- Design principles of access control systems:
  - Check every access: every access **by a subject to an object** should be checked; granting the subject the privilege previously does not mean that the subject should retain this privilege indefinitely.
  - Allow least privilege: subjects should be given just enough privileges to perform their tasks; unnecessary access should not be allowed even if this extra access is useless or harmless.
  - Verify acceptable usage: invalid operations should be denied; it is important to check that an operation to be performed on an object is a valid one.

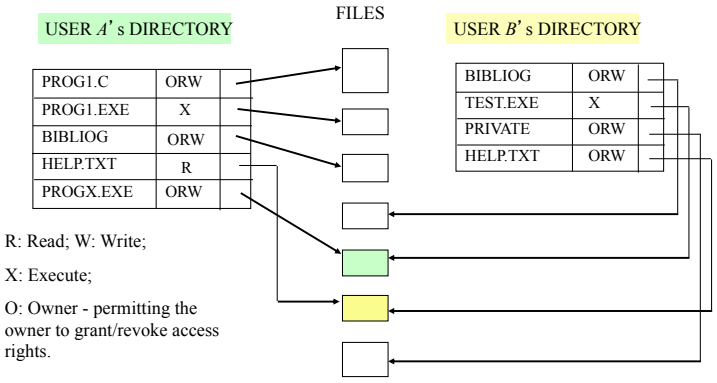
Access Control Basic Concepts

- Access control mechanisms/models
  - Discretionary Access Control (DAC)
    - Identity-based access control
  - Mandatory Access Control (MAC)
    - Classical model: clearance-based access control
  - Role Based Access Control (RBAC)
    - A user's permissions are determined by users' roles, rather than identity or clearance level.
    - Roles can encode arbitrary attributes.
    - Scalable solution by resembling organisational structures in large organisations.

Part 2 Overview

- DAC mechanisms
  - Part 2.1
    - Directory access: lists all the files a user (subject) can access
  - Part 2.2
    - Access control matrix: table of subjects, objects & rights
    - Access control list: an object versus subjects
    - Capability list: a subject versus objects
  - Part 2.3
    - Procedure-oriented access control: information hiding.

DAC - Directory Access



DAC - Directory Access

- Directory access
  - One list per subject, showing all the objects that are allowed to access by the subject;
  - The list can become too large if there are many shared objects;
  - Revocation of access rights can be time-consuming;
  - Pseudonyms or file naming inconsistency;
  - The approach can not efficiently address most object protection scenarios.

DAC - Access Control Matrix

- An entry in the matrix specifies the set of access operations a subject *s* can perform on an object *o*.
- Each column: ACL for one object - defines who can perform what operation.
- Each row: A subject's capability list – defines, for a given subject, what operations allowed on what objects.

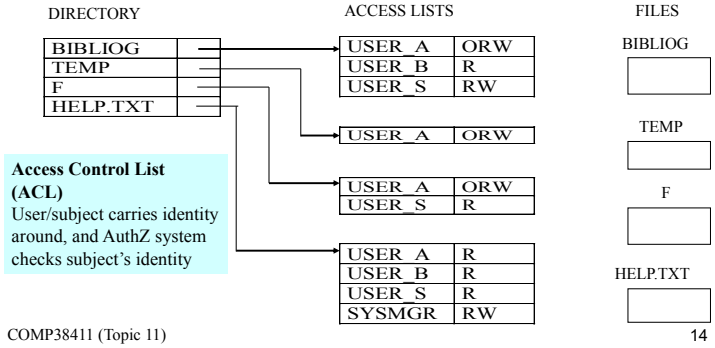
User_B's Capability List

	BIBLIOG	TEMP	F	HELP.TXT
USER_A	ORW	ORW	ORW	R
USER_B	R	-	-	R
USER_S	RW	-	R	R
USER_T	-	-	-	R
SYS_MGR	-	-	-	RW
USER_SV	-	-	-	O

Access Control List (ACL) for F

DAC - Access Control List

- An Object List lists *subjects* and their *access rights*



DAC - Access Control List

- ACL is a decomposition of the matrix by columns.
- One list per object, showing a set of pairs, {*Subjects, Rights*}, indicating the named *subject* can access the associated *object* using any of the *rights*.
- Usually, if a subject is not named, s/he has ***no*** rights over file – the principle of Fail-Safe Default.
- A shared public object can have a very short access list, explicitly naming the few subjects who should have access rights different from the default.
- To shorten a list, specific users can have explicit rights and all other users (defined using 'wildcard') can have a default set of rights.

DAC - Access Control List

- If there are many subjects, you may use groups or wildcards in ACL to shorten the list, e.g.
  - UNICOS:
    - Entries are (*user, group, rights*)
    - If *user* is in *group*, has rights over file
    - '*' is wildcard for *user* and *group*
      - (holly, *, r): holly can read file regardless of her group.
      - (*, groupA, w): anyone in groupA can write file.
  - UNIX
    - Three classes of users: owner, group, others (the rest)
- Rights revocation
  - Owner deletes subject's entries, or rights from subject's entry, in ACL.

DAC – Capability

- A Capability list/ticket is a row from the Access Control Matrix
  - An example:

USER_S	BIBLIOG {RW}	F {R}	HELP.TXT {R}

  - A capability ticket specifies a given subject can perform what operations on what objects.
  - User carries around tickets; each user may have a number of tickets.
  - User may grant rights to another user, and also grant the right to grant rights.
  - Excellent for implementing 'temporary access' - access rights given to a subject are only valid for a certain duration.

DAC – Capability

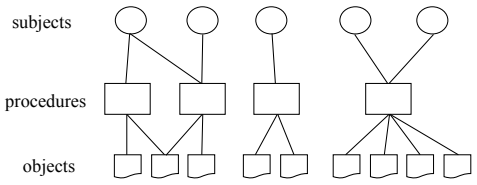
- Issues
  - Capability revocation
    - Scan all C-lists, remove relevant capabilities – too expensive.
    - Other methods – temporary access, etc, more in the research domain.
  - Must protect capability tickets against unauthorised alterations either by users or processes
    - as otherwise, subject could change rights encoded in a capability ticket or object to which they refer.
    - often done by using a cryptographically protected checksum using a key known to OS, or by having a trusted entity to hold/store the tickets.

DAC – Capability versus ACL

- Both theoretically similar:
  - Q1: Given a subject, what objects can it access, and what operation?
  - Q2: Given an object, what subjects can access it, and what operation?
  - C-List answers Q1, and ACLs answers Q2, better.
- In the past, more attentions have been on Q2 (largely because the use of non-distributed multi-user systems), which is the reason ACL-based systems are more common than C-based systems.
- Q1 is becoming more important in some applications, e.g. incident response and large-scale distributed systems, so there is a possibility 'Capability Lists may come back'.

DAC - Procedure-oriented

- Objects are encapsulated, permitting only certain specified accesses via program execution (e.g. you can only access a web server via a browser).



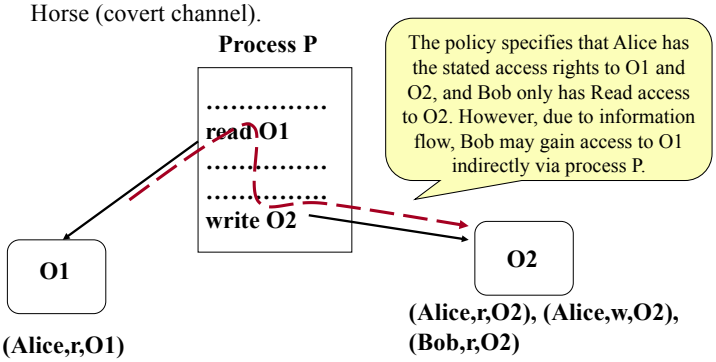
- Accesses to an object are done through a software procedure - a trusted interface.
- The penalty is performance cost.

Part 3 Overview

- Mandatory Access Control (MAC)

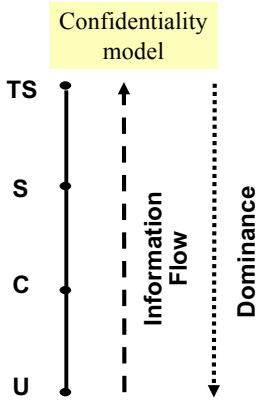
Mandatory Access Control (MAC) – Motivation

- DAC does not have information flow control -> Risk of Trojan Horse (covert channel).



MAC

- DAC cannot protect data against Trojan Horses embedded in programs, but MAC can, preventing unauthorised information leakage (ensuring confidentiality).
- Rights are determined by security labels:
  - For objects: **classification** (security/sensitivity).
  - For subjects: **clearance** (trust) levels.
- *C* is no more restrictive than *S*; *C* is dominated by *S*; information may flow from *C* to *S*.
- Also referred to as *multilevel security*.



{Top Secret (TS), Secret (S), Classified (C), Unclassified (U)}, where TS > S > C >U.

MAC

- A system-wide policy decrees who is allowed to access what.
- Used to control information flow in highly confidential environments such as military contexts and/or to prevent malicious software from modifying highly sensitive information such as access control policies.

- Subjects and objects are classified into different levels of trust and sensitivity:
  - Subjects are assigned clearance levels.
  - Objects are classified into categories each assigned with a classification level.
  - Only the subjects with a certain clearance level are granted with access to objects with a given security level.

MAC – BLP Model

- Mandatory control rules for **secrecy/confidentiality**
  - No read up (**read down**): object's classification must be below (or equal to) subject's clearance.
  - No write down (**write up**): object's classification must be above (or equal to) subject's clearance.
- The above rules can be formally expressed as: each subject *s* in *S* and each object *o* in *O* has a fixed security level $C(s)$ and $C(o)$ (denoting clearance and classification level), we have:
  - Subject *s* may have *read* access to an object *o* only if $C(s) \geq C(o)$.
  - Subject *s* can write to an object *o* only if $C(o) \geq C(s)$.
  - Subject *s* who has *read* access to an object *o* may have *write* access to an object *p* only if $C(p) \geq C(o)$.
- This is also referred to as **Bell-LaPadula (BLP) Confidentiality Model**

MAC – Biba Model

- Mandatory control rules for **integrity** (here security=integrity)
  - **Read up**: object's classification must be above (or equal to) subject's clearance.
  - **Write down**: object's classification must be below (or equal to) subject's clearance.
- The above rules could be formally expressed as: each subject *s* in *S* and each object *o* in *O* has a fixed integrity level $I(s)$ and $I(o)$ (denoting clearance and classification level), we have:
  - Subject *s* can read an object *o* only if $I(o) \geq I(s)$.
  - Subject *s* can modify (have *write* access to) object *o* only if $I(s) \geq I(o)$.
  - If subject *s* has *read* access to object *o* with integrity level $I(o)$, *s* can have *write* access to object *p* only if $I(o) \geq I(p)$.
- This is also referred to as **Biba Integrity Model.**

Part 4 Overview

- RBAC (Role Based Access Control)
  - Part4.1: Background
  - Part4.2: RBAC
- Conclusion

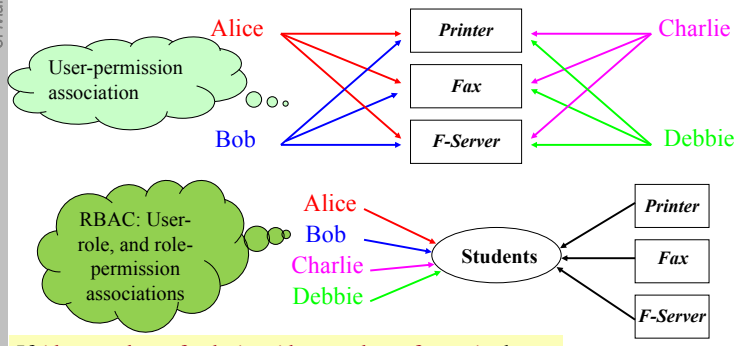
RBAC – Why do we need it

- Managing access control in a large organisation can be challenging
- When the numbers of subjects and objects are high, permission assignments can be complex and error-prone.
- If user population is highly dynamic, managing *grant* and *revoke* operations can be time-consuming.
- End users are often not the owners of resource objects being managed; the organisation is.
- Control is often based on employee functions rather than data ownership.

RBAC – Why do we need it

- RBAC provides a valuable level of abstraction - **roles**
- The idea is
  - Set permissions based on **roles**
  - Assign users to a role or a set of roles
- Because the roles within an organisation are relatively stable; they change less frequently than user turnover, RBAC can
  - simplify the task of access control management
  - reduces cost and potential errors in administrative process

RBAC - Why do we need it



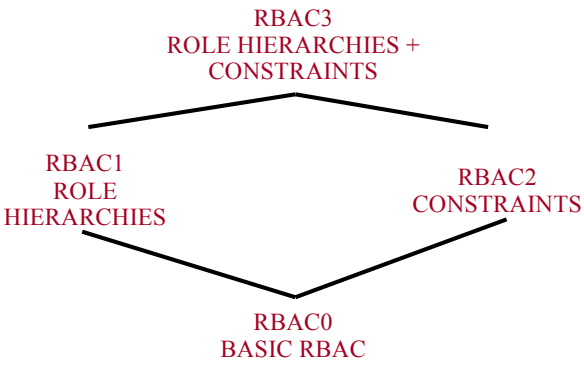
If (the number of roles) < (the number of users), the number of operations necessary to grant and revoke access privileges is greatly reduced.

RBAC – Security principles

- Least privilege
  - No more privilege than necessary to perform his/her job function.
- Separation of duties (conflict of interest constraints)
  - Static separation of duty: a user should not be authorized for both roles, e.g., student and staff.
  - Dynamic separation of duty: a user should not act simultaneously in both roles, e.g., casher and customer.

RBAC – Functional capabilities

- RBAC96 Family of Models

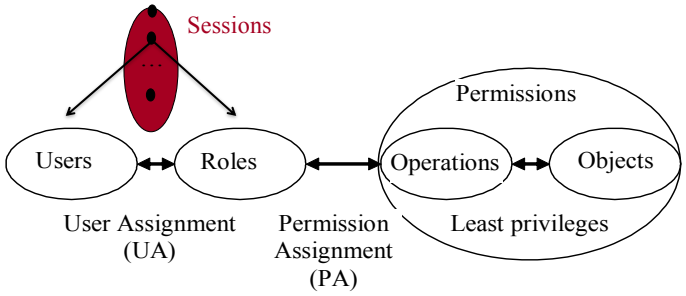


RBAC – RBAC0

- Users are assigned to roles, permissions are granted to roles, and users acquire permissions by being members of roles.
- A role is
  - a collection of users and
  - a collection of permissions
- Permissions are positive
- No negative permissions or denials
  - negative permissions and denials can be handled by constraints

RBAC – RBAC0

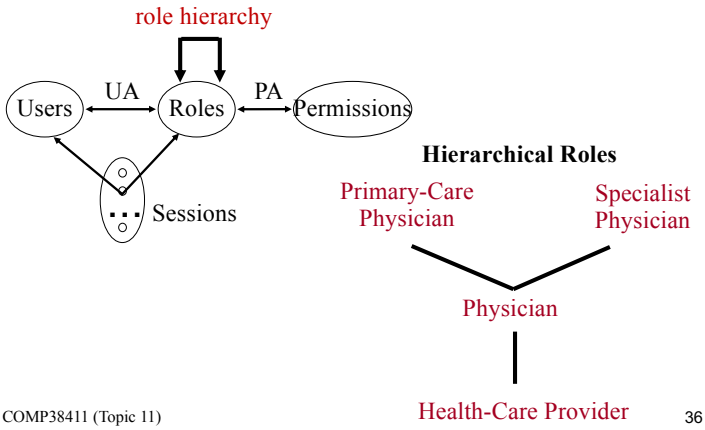
- A user can invoke multiple sessions; in each session a user can invoke any subset of roles that the user is a member of.



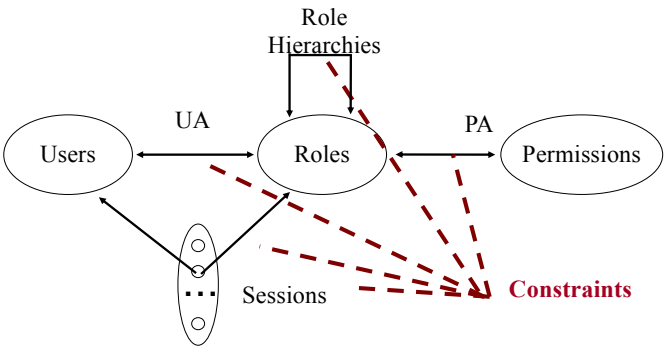
RBAC – RBAC0

- Configuring RBAC involves the following tasks
  - Determining permissions.
  - Determining functional roles based upon tasks, responsibilities, and qualifications, etc.
  - Assign users to the roles (UA)
  - Assign permissions to roles (PA)
- Ability to support many-to-many UA and PA relations
  - A user can have many roles; a role can have many users.
  - A permission can be assigned to many roles; each role can have many permissions.

RBAC – Role Hierarchy



RBAC – Constraints



RBAC – Constraints

- Mutually exclusive roles
  - Static exclusion: the same individual can never hold both roles
  - Dynamic exclusion: the same individual can never hold both roles in the same context
- Mutually exclusive permissions
  - Static exclusion: the same role should never be assigned both permissions
  - Dynamic exclusion: the same role can never hold both permissions in the same context

RBAC – Constraints

- Cardinality constraints on User-role Assignment (UA)
  - At most k users can belong to the role
  - At least k users must belong to the role
  - Exactly k users must belong to the role
- Cardinality constraints on Permissions-role Assignment (PA)
  - At most k roles can get the permission
  - At least k roles must get the permission
  - Exactly k roles must get the permission

Exercise Question – E11.1

Contrast the following AC mechanisms, Directory, Access Control List, Capability and Procedure-oriented AC, **in terms of how easy it is** (for each case, you should consider what the system needs to do to accomplish the operation):

i) to determine authorised access during execution.
ii) to add access for a new subject.
iii) to delete access by a subject.
iv) to create a new object to which all subjects by default have access.
v) to revoke partial rights of a subject in the system.

Exercise Question – E11.2

- Rewrite an AC policy using MAC to mitigate the risk of trojan horse shown in the MAC-Motivation slide.

Conclusions

- Several mechanisms can be used to enforce access control, organised on a per-object or per-subject basis - they may be suited for different scenarios or application contexts.
- DAC models are flexible in terms of policy specification, and are typically used by OS and DBMS in commercial world. However, they do not provide information flow control, thus vulnerable to trojan horse attacks.
- MAC can protect against Trojan Horse attacks, and is typically used in military applications, and/or for constructing trusted computing systems.
- RBAC resembles the management structures of large organisations; it defines a user's permissions based on the user's roles rather than his identity (DAC) or clearance (MAC).