Topic 8

Key Management

Address the problems of key establishment and key distribution

Overview

- Key management issues
- Diffie-Hellman (DH) protocol
- Distribution of secret keys without any use of public-key ciphers
- Distribution of secret keys using public-key ciphers
- A summary of secret key establishment protocols
- Conclusion

source: Chapter 14 of Cryptography and Network Security.

Key Management Issues (1)

- 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 (2) - Keys spaces

- Number of possible keys 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 (3) - 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 (4) - Keys spaces

- **Main points**
  - Giving various constraints on the input string can greatly reduce the number of possible keys, therefore making cryptosystems much easier to break!
  - Computer power double every 18 month….
    - If you expect your keys to stand up against brute-force attacks for 10 years, plan accordingly.

Key Management Issues (5) – 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.
- Ordinary random number generation functions, e.g. `java.util.Random,` may not be good enough for this purpose.
- Use a reliably random source, or a cryptographically secure pseudo-random-number generator, e.g. `SecureRandom` class in `java.security` package.

Key Management Issues (6) - Key generation

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

Key Management Issues (7) - 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 (8) - 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 (9) - 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 the key into two halves, store
    - one half in the machine, and
    - another half in the card.

11

Key Management Issues (10) – Session key establishment

- More often a symmetric key is used, more likely it may be broken.
- Generate and use a symmetric (secret) key for one session only → session key.
- It is desirable to use different session keys in different sessions, as this can
  - limit available ciphertexts for cryptanalysis.
  - limit exposure (both in time period and amount of data) in an event of key compromise.
  - avoid long-term storage of a large number of secret keys by only creating them when needed.

12

Key Management Issues (11) – Session key establishment

- Session key establishment solutions
  - Key agreement (exchange) protocols
    - A shared 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 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.

Key Management Issues (12) – Session key establishment

- There are other issues that should be considered
  - Entity and key authentication
    - Assurance: no other party (outsiders - apart from the entities involved) could gain access to the established session key.
    - 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 Protocol (1) - Overview

- 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 $y = g^x \bmod n$.
  - Solutions known for small n.
  - Solutions computationally infeasible as n grows large.

Diffie-Hellman Protocol (2) - The algorithm

- 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.

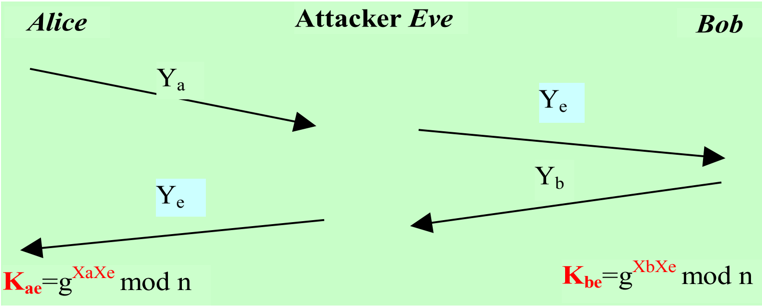
Diffie-Hellman Protocol (3) - The algorithm

Alice		*Bob*
Generate Random & private $X_a < n$; **Calculate** Public $Y_a = g^{Xa}$ mod n	Y_a →	**Generate** Random & private $X_b < n$; **Calculate** Public $Y_b = g^{Xb}$ mod n
Calculate $\mathbf{K_{ab}} = Y_b^{Xa}$ mod n $= g^{XaXb}$ mod n	← Y_b	**Calculate** $\mathbf{K_{ab}} = Y_a^{Xb}$ mod n $= g^{XaXb}$ mod n

Diffie-Hellman Protocol (4) - The algorithm

- 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 (5) - 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* .
- Do you have a solution to make DH immune to this attack?

Distribution without use of PKC (1) - Approach one

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

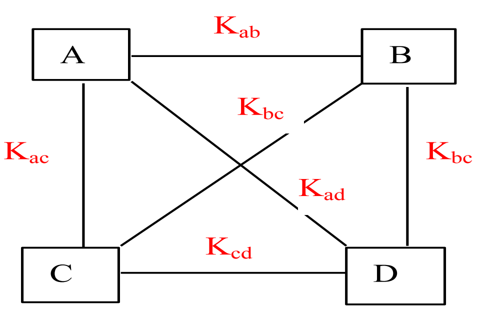


Figure 1: One hierarchical Key distribution.

Distribution without use of PKC (2) - Approach two

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

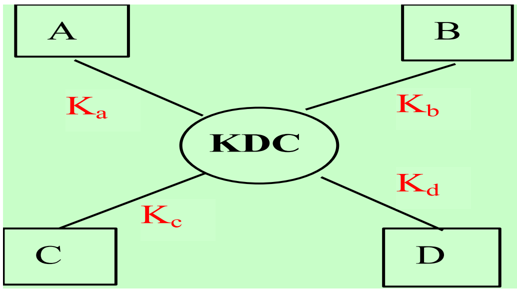


Figure 2: Two hierarchical Key distribution.

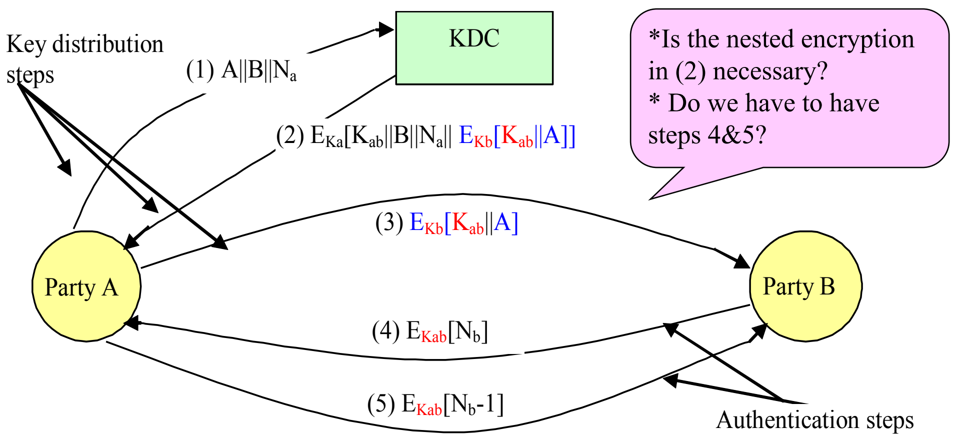
Distribution without use of PKC (3) - Approaches one vs two

- A unique master key, shared between a pair of user/*KDC*, is for session key distribution.
- The session key is to secure the communication.
- Advantage 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.
- Disadvantages of using approach two:
  - 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 (4) - 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.
- N_a, N_b are nonces (random challenges), generated by A and B respectively, to keep the request fresh.

Distribution without use of PKC (5) - Needham-Schroeder Protocol



Distribution without use of PKC (6) - 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.

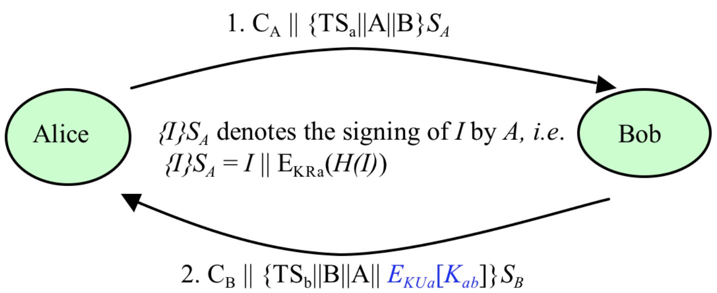
(4): B sends a nonce N_b to *A* encrypted using the new session key.

(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.

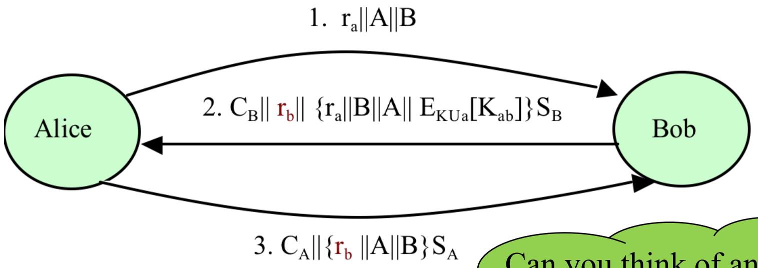
Distribution using PKC (1) – Two passes

- 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).



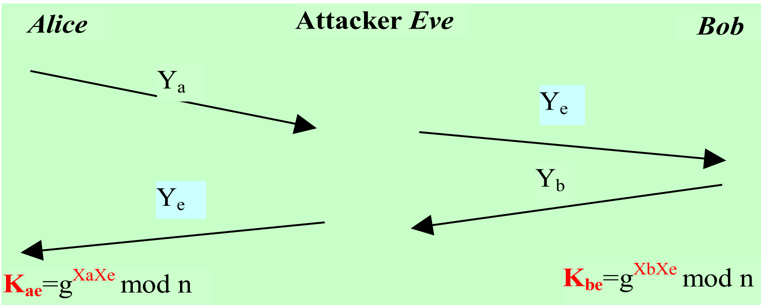
Can you think of any other variant of such authN protocols (using PKC)?

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
Kerberos	KDC (online)	Yes	Symmetric encryption	4
X.509 (2 pass)	CA (offline)	Yes	mutual	2
X.509 (3 pass)	CA (offline)	No, but with nonce	mutual	3

Exercise 8 (a)

i. Imagining Alice is to send a message, *M*, to Bob encrypted with a shared key established using the DH protocol. Explain whether Eve could access this message *M* by launching the man-in-the-middle attack, and if so, how.

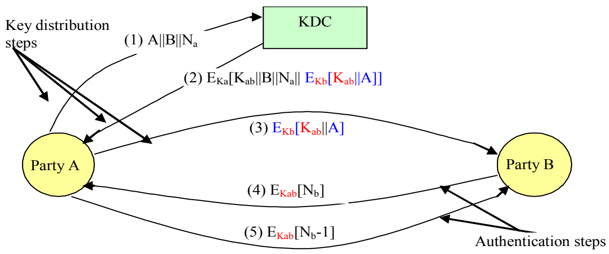


ii. Propose a solution to this vulnerability.

Exercise 8 (b)

- You may familiarize yourself with the Diffie-Hellman key exchange protocol using the Demos facility in CrypTool.
- The facility is available via Menu: “Indiv. Procedures” \“Protocols” \“Deffie-Hellman Demostration”.

Exercise 8 (c)



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.

Conclusion

- 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 algorithms can be vulnerable to security attacks, such as the man-in-the-middle and replay attacks, so they should be used with care.