# IK2206 – Internet Security and Privacy
## Chapter 9 – Authentication Systems

### 9.1 Password-Based Authentication

- The course book makes a distinction between "passwords" and "cryptographic keys". In what way do they differ? (Security strength? Ease of memorizing? Entities commonly authenticated (humans or machines?))

| | Password | Cryptographic Key |
|---|---|---|
| General Difference | A password together with the identity's name is sent to the authenticator system. | A machine-generated key (sometimes with help of a password) is used to encrypt data. |
| Security strength? | Very low. Everyone on the network can eavesdrop the password. | As long as the key is kept secret, the security is pretty good. |
| Ease of memorizing? | Depends on complexity. | Not memorisable. |
| Entities commonly authenticated | Humans | Machines |

#### 9.1.1 Off- vs. Online Password Guessing

- What is an "On-line Password Guessing Attack"?

  If an attacker is doing an "online password guessing attack" he/she is basically trying to authenticate with different combinations of passwords.

- What are the protection mechanisms? (Slow it down?)

  Most systems limit the number of tries per unit time. Some systems implement a long wait time after several negative tries. An unusual large number of incorrect login tries should be a sign for system administrators to investigate.

- What is an "Off-line attack"? How is it carried out?

  An attacker collects (parts of) passwords and tries to compare them against a huge list of commonly known passwords (also called dictionary or rainbow table).

- What is a "Dictionary attack"?

  A dictionary is comparable to a language dictionary – only that it contains most commonly used passwords.
  In contrast with a brute force attack, where a large proportion key space is searched systematically, a dictionary attack tries only those possibilities which are most likely to succeed.

### 9.1.2 Storing User Passwords

- What is the problem with storing password databases in clear-text?

  The database with the passwords could be stolen. System administrators could gain access to sensible data.

- What is the difference between a "hashed password" database and an "encrypted password" database?

  Hashed passwords are irreversible. No private key must be kept.
  Encrypted passwords are encrypted by a key that has to be carefully kept private.

- In early Unix systems the hashed password database was made "world readable". What kind of attack was simplified by this approach?

  Offline attacks like dictionary attacks are possible if an intruder gains access to passwords. Commonly known passwords are subject to such attacks.
  **???**

- What could happen if a user forgets his/her password:
  - when a cleartext password database is used?
    - Recover the password from database
  - when a hashed password database is used?
    - Reset the password
  - when an encrypted password database is used?
    - Decrypt the password from database using the secret key
- What could happen if the password used to encrypt a password database is lost?
  - How is it stored? Does it need to be typed in by administrator upon reboot?

    If an appropriate key to encrypt data isn't available anymore, the encrypted data can be considered as garbage. Thus, all passwords are lost if the decryption key or the appropriate password isn't available anymore.
    It hardly depends on the implementation whether a password has to be typed in. This step is theoretically not necessary.

- What would be the problem with a system where servers ask an "authentication storage node" whether a user should be granted access, if the "authentication storage node" does not need to authenticate itself to the asking server? (Course book gives the common configuration of Sun NIS directory server as example.)

  Everyone could place his own "authentication storage node" and claim to be the one-and-only authenticator. An attacker could set up his own authenticator that stores user accounts with known passwords. This is of course a big security hole! A server must not trust any authentication system.

### 9.2 Address-Based Authentication

- Can you think of any situations where address-based authentication is useful?

MAC address based blacklists on WLAN routers.
MAC address based authentication on network switches.

- With address-based authentication, what would happen if an attacker manages to "break into" an account on a machine, which the system considers trustworthy?

### 9.2.1 Network Address Impersonation

- How would it be possible to impersonate a network address? How could such attacks be limited?

  An attacker could manipulate the frame resp. diagram header of transmitted messages. It is also possible to set up routing functionality on an intruder's machine and let the neighboring routers know about our existence (MITM traffic rerouting). Another way to impersonate network addresses is "IP source routing" in which an attacker defines a particular route that a packet has to travel.

  To avoid such attacks, cryptographic authentication between routers is used to make sure that no intruder can fake/reroute traffic. Even then it is hard to reach 100% safety. End-to-end tunneling could be another approach.

## 9.3 Cryptographic Authentication Protocols

- What is meant by the term "Cryptographic Authentication Protocol"? How does the course book define it?

  The basic idea of cryptographic authentication is that A proves its identity to B by performing a cryptographic operation on a quantity B supplies.

## 9.4 Who Is Being Authenticated?

- This is not always as trivial as it may seem. What about using WPA, IPSec, TLS or Web-login? Is a user or a machine being authenticated? Or an application? What if it is a multi-user system? Who is granted access to the system?

| | Who is being authenticated? | |
|---|---|---|
| | Human | Machine |
| WPA | | X |
| IPSec | X | |
| TLS | (X)[1] | X |
| Web Login | X | |

[1] Identification and authentification based on public key cryptography is possible.

### 9.5 Passwords versus Cryptographic Keys

- How could a secret key (e.g. DES) be generated from a password?

  A possible transformation of a user's password to a secret key is to do a cryptographic hash from the password and take 56 bits (=DES key length) of the result.

- Why is it more difficult to generate a private key from a password than a secret key from a password?

  If we'd use a password as seed for a RSA key generator it would take an unacceptable amount of time to find two primes. In practice, private keys are symmetrically encrypted by the user's password. The decryption operation takes much less time than the operation of finding the two prime numbers.

### 9.6 Eavesdropping and Server Database Reading

- Consider how it would be possible to design a "secret key protocol" which is both secure from "eavesdropping" and from an "intruder reading the 'server password database'"? (See "12.2 Lamport's hash" for a protocol, which can claim to accomplish this.)

  One-Time-Passwords (OTP) as described by Lamport makes use of the irreversibility of hash functions. The client calculates $hash^n$(password) using an n given by the server. With each reauthentication, n is decreased by one. As soon as n reaches 0, the OTP must be reinitialized.

  Lamport's hash is limited by the initial value for n. If n is very large, then the calculation of the initial value $hash^n$(password) is too time-consuming, if n is small, then the scheme needs to be reset soon.

  http://www.cse.scu.edu/~tschwarz/coen350/lamportHash.html
  http://de.wikipedia.org/wiki/Einmalkennwort

- Why would not this be a problem for "public key protocols"?
  - Or would it?

    ???

### 9.7 Trusted Intermediaries

- Why do we need trusted third parties (trusted intermediaries)?

  If we had no trusted intermediaries, nodes would have to generate store keys of all communication partners, means, each node has to store n-1 keys.

- To add a new node to a group of nodes (where each pair of nodes has a secret key) how many new keys needs to be installed?

  If a new node were added to the network, n keys would need to be generated so that each node shares a secret with each of the other nodes.

- What is the difference between a KDC and a CA? In what systems (public key or secret key) are they used?

  KDC's store secret keys (=symmetric cryptography) while CA's store public keys (=asymmetric cryptography) of the participating nodes.

### 9.7.1 Key Distribution Centers (KDCs)

- To add a new node to a group of nodes utilizing a KDC, how many new keys needs to be installed?

  1 new key needs to be placed at the KDC

- What is a "ticket"?

  A ticket contains encrypted authentication information. The holder of the ticket can typically not see its content but he/she can use it to authenticate against another system. The authenticator uses its private key to decrypt the ticket and check the authentication information.

- How do two principals (e.g., users or nodes) authenticate each other in a KDC system?

  1. Node A asks the KDC for a key with which it can talk to B. This communication is, of course, encrypted with A's key.
  2. The KDC authenticates A and chooses a random number $R_{AB}$ . This number is used as key for the encryption between A and B. The KDC encrypts this new key with A resp. B's key for delivery.
  3. **That's it???**

### 9.7.2 Certificate Authorities (CAs)

- What is the role of a CA?

  A certificate authority (CA) generates public key certificates. Certificates issued by a certain CA let others (external parties) rely upon signatures made by the private key that corresponds to the public key that is certified. In this model of trust relationships, a CA is a trusted third party that is trusted by both the subject (owner) of the certificate and the party relying upon the certificate.

- What is a certificate?

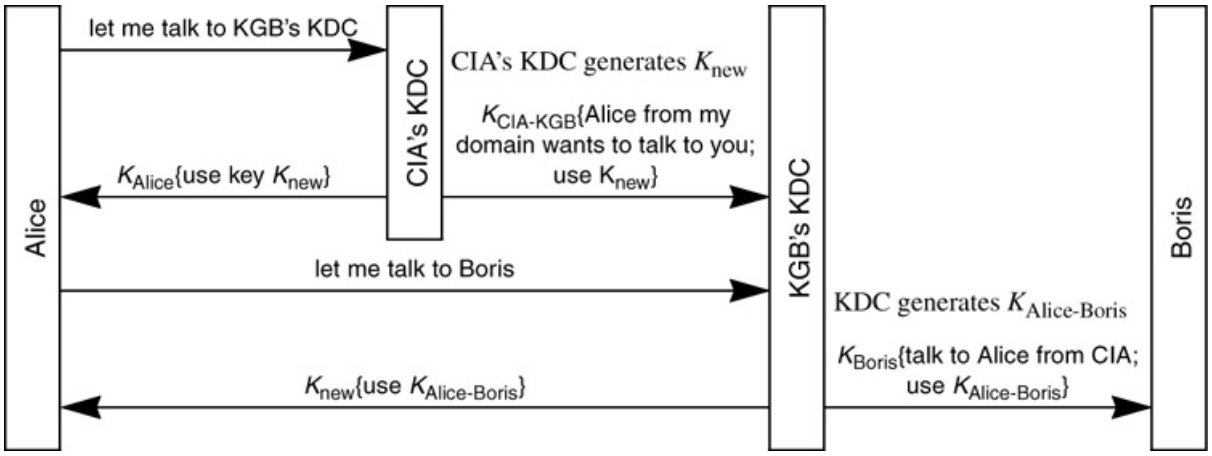
  A certificate is an electronic document which uses a digital signature to bind a public key with an identity. Elements of a certificate are: name, address, issuer, expiration date and a serial number. Further, all contents are signed by a CA. The certificate can be used to verify that a public key belongs to an individual.

- What are the advantages of CA's over KDC's?
  - A CA does not need to be online. KDC's should at least be online when hosts restart.
  - The previous fact makes CA's much more secure than KDC's.
  - Public key certificates are – in contrast to the secret keys of a KDC – not security sensitive. They can be stored on a public directory.

- A compromised CA cannot decrypt conversations.

#### 9.7.4.1 Multiple KDC Domains

- How could Alice and Bob authenticate each other if they are associated with different KDC domains?




The most important point is that KDC's of different trust domains share a key ($K_{CIA-KGB}$) with which they encrypt key exchanges.

- Can you see any scalability issues when establishing a large authentication infrastructure based on KDCs (or CAs)?

  One of the main disadvantages of CA's are their large CRL's. Furthermore, the handling of CRL's is always a tradeoff between the actuality of the entries and the refresh effort.

  **More???**

- Are there any additional security issues when establishing a shared secret over multiple KDC hops? (See the "CIA-MIT-KGB" example in the course book.)

  Trusts between different trust domains should not reflect an "any-to-any" trust model. This could be very dangerous. Trusts are always unidirectional, from one hop to the other.

  **More???**

#### 9.7.4.2 Multiple CA domains

- How could Alice and Bob establish a shared secret if they are associated with different CA domains?

  Alice obtains Bob's CA certificate. This certificate states that the public key of the CA is $P_1$ and that it is certified by her own CA.

  Alice obtains Bob's certificate. This certificate contains his public key $P_2$ and is signed by Bob's CA using $P_1$.

### 9.8 Session Key Establishment

- Can you think of any security system which only provides authentication, but does not protect the subsequent conversation between Alice and Bob? Web-login? Dial-up (PPP) to your ISP? HTTPS to your Internet Bank? Mail retrieval with POP from your Mail server?

  Unfortunately, many "home-brewed" web authentication forms implement authentication only.

- If Alice and Bob share a long term secret (based on shared or public key cryptography), why would they still like to create session keys to encrypt (and integrity protect) data of their conversation?

  Keys sort of "wear out" after some time. The more time and data an intruder gets, the more likely he/she'll succeed in breaking the encryption.

- Why is it desirable to let the session key be a secret key (rather than a pair of public and private keys)?

  Session keys are usually used to encrypt bulk data. It is by far more efficient to use symmetric encryption in this case. Public key operations are computationally more expensive.

### 9.9 Delegation

(A bit out of scope, but interesting) How do you think your file is protected when sending it to a printer on a network? Is it?