

CS 5523 Lecture 18: Security Case Studies

- Discuss laboratory 2
- Discuss laboratory 3
- Review scenarios
- Needham-Schroeder
- Brief summary of common algorithms
- Kerberos

Simplified overview of secret key encryption:

Encrypted message:

$$E(K, M) = \{M\}_K$$

Decrypted message:

$$D(K, E(K, M)) = D(K, \{M\}_K) = M$$

It is hard to get M from $\{M\}_K$ without knowing K

Scenario 1. Secret communication with shared secret key

Alice and Bob share a secret key K_{AB} . Alice wants to send a secret message M to Bob.

1. Alice uses K_{AB} and an agreed encryption function $E(K_{AB}, M)$ to encrypt and send message M to Bob
2. Bob reads the encrypted messages using the corresponding decryption function $D(K_{AB}, M)$

How can Bob and Alice safely get the shared key K_{AB} ?

How can Bob know that M wasn't a replay?

Scenario 2.

Authenticated communication with a server

Alice wants to access Bob's files on a local file server. Sara is a trusted authentication server that holds passwords and current secret keys.

1. Alice sends a message to Sara asking for a ticket to access Bob
 2. Sara sends Alice a response encrypted with K_A that is a ticket encrypted with K_B and a new secret key K_{AB} for communication:
 $\{\{ticket\}_{K_B}, K_{AB}\}_{K_A}$
 3. Alice decrypts response with K_A
 4. Alice sends ticket, her ID and request R to Bob: $\{ticket\}_{K_B}, Alice, R$
 5. Bob decrypts ticket using K_B (the ticket was $\{K_{AB}, Alice\}_{K_B}$)
- This is the simplified scenario for Kerberos. K_{AB} is the session key.

Simplified overview of public key encryption:

Keys come in pairs K_1 and K_2 . Keep one public and one private.
If you encrypt with K_1 , you can decrypt with K_2 and vice versa:

$$D(K_2, E(K_1, M)) = M$$

and

$$D(K_1, E(K_2, M)) = M$$

Scenario 3.

Authenticated communication with public keys

Bob has generated a public/private key pair. There is a trusted authority that gives out key certificates

1. Alice accesses a key distribution center to obtain a public key certificate with Bob's public key. Alice extracts Bob's public key K_{Bpub}
 2. Alice creates a new secret key K_{AB} and encrypts $\{K_{AB}, \text{known string}\}$ with K_{Bpub}
 3. Alice sends $\{\{\text{unique keyname}\}, \{K_{AB}, \text{known string}\}_{K_{Bpub}}\}$ to Bob.
 4. Bob decrypts $\{K_{AB}, \text{known string}\}_{K_{Bpub}}$ using K_{Bpriv}
 5. Bob and Alice now communicate with K_{AB}
- This is the scenario for the widely used hybrid cryptographic protocol.

Scenario 4. Digital signatures with a secure digest function

Alice wants to sign document M so that any recipient can verify it came from Alice. This assumes that Alice has a private-public key pair. A digest is like a checksum.

1. Alice computes a fixed-length digest $Digest(M)$.
2. Alice encrypts $Digest(M)$ with her private key certificate with Bob's public key and makes $\{M, \{Digest(M)\}_{K_{Apriv}}\}$ available.
3. Bob reads $\{M, \{Digest(M)\}_{K_{Apriv}}\}$, extracts M and computes $Digest(M)$.
4. Bob applies K_{Apub} to $\{Digest(M)\}_{K_{Apriv}}$ to obtain $Digest(M)$ and compares the value with his computed value.

Figure 7.4
Alice's bank account certificate

1. Certificate type	Account number
2. Name	Alice
3. Account	6262626
4. Certifying authority	Bob's Bank
5. Signature	$\{Digest(\text{field 2} + \text{field 3})\}_{K_{Bpriv}}$

Figure 7.5
Public-key certificate for Bob's Bank

1. Certificate type	Public key
2. Name	Bob's Bank
3. Public key:	K_{Bpub}
4. Certifying authority:	Fred – The Bankers Federation
5. Signature	$\{Digest(\text{field 2} + \text{field 3})\}_{K_{Fred}}$

Figure 7.13
X509 Certificate format

<i>Subject</i>	Distinguished Name, Public Key
<i>Issuer</i>	Distinguished Name, Signature
<i>Period of validity</i>	Not Before Date, Not After Date
<i>Administrative information</i>	Version, Serial Number
<i>Extended Information</i>	

Some popular encryption schemes:

- **TEA** = tiny encryption algorithm – Wheeler and Needham, 1994 uses 32 rounds with combinations of XOR, text shifts
- **DES** = Data Encryption Standard – National Bureau of Standards 1977 – obsolete because of short keys – successfully cracked by brute force attacks in 1997 and a machine was built in 1998 that could crack keys in 3 days. Replaced by AES (Advanced Encryption Standard, NIST 1999)
- **RSA** – Rivest Shamir and Adelman – public key encryption based on factoring products of large primes – widely used (RSA's patent just expired)

Some popular encryption schemes (continued):

- **3DES** = triple-DES – ANSI 1985 – apply DES 3 times with two keys – very slow
- **IDEA** = International Data Encryption Algorithm – Lai and Massey, 1990 – based on group algebra with 8 rounds of XOR
- **MD5** = used for data digests – Rivest 1992 – four rounds applying 4 nonlinear functions to each of 16 32-bit segments
- **SHA** - NIST 1995 – based on Rivest's MD4 algorithm to produce a 160 bit digest

Figure 7.14
Performance of encryption and secure digest algorithms

	Key size/hash size (bits)	Extrapolated speed (kbytes/sec.)	PRB optimized (kbytes/s)
TEA	128	700	-
DES	56	350	7746
Triple-DES	112	120	2842
IDEA	128	700	4469
RSA	512	7	-
RSA	2048	1	-
MD5	128	1740	62425
SHA	160	750	25162

Instructor's Guide for Cuihua, Dullman and Kuehng Distributed Systems: Concepts and Design, Edn. 3
© Addison-Wesley Publishers 2000

Figure 7.15
The Needham-Schroeder secret-key authentication protocol

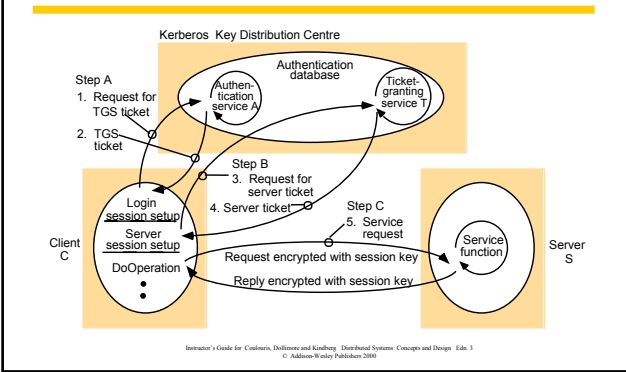
Header	Message	Notes
1. A->S:	A, B, N_A	A requests S to supply a key for communication with B.
2. S->A:	$\{N_A, B, K_{AB}\}_{K_S}$ $\{K_{AB}, A\}_{K_B}$	S returns a message encrypted in A's secret key, containing a newly generated key K_{AB} and a 'ticket' encrypted in B's secret key. The nonce N_A demonstrates that the message was sent in response to the preceding one. A believes that S sent the message because only S knows A's secret key.
3. A->B:	$\{K_{AB}, A\}_{K_B}$	A sends the 'ticket' to B.
4. B->A:	$\{N_B\}_{K_{AB}}$	B decrypts the ticket and uses the new key K_{AB} to encrypt another nonce N_B .
5. A->B:	$\{N_B - 1\}_{K_{AB}}$	A demonstrates to B that it was the sender of the previous message by returning an agreed transformation of N_B .

Instructor's Guide for Cuihua, Dullman and Kuehng Distributed Systems: Concepts and Design, Edn. 3
© Addison-Wesley Publishers 2000

Kerberos:

- Follows Needham and Schroeder very closely
- Uses time values as nonces
- When user logs in, the login program sends user's name to the kerberos authentication server
- If user is known, server replies with a session key and a nonce encrypted in the user's password and a ticket for TGS
- After login program authenticates the information, it can erase the user's password from memory

Figure 7.16
System architecture of Kerberos



Kerberos (continued):

- When a program needs a service, it requests a ticket from TGS
- Server machines must take care to store their keys in a safe place
- Kerberos is implemented as a server running on a "secure" machine
- DES encryption is used, but it can be replaced
- Kerberos is scalable – world divided into realms
- Processes can authenticate themselves to servers in other realms through their local TGS
- Ticket lifetimes of 12 hours are typically used

For next time:

- Finish reading Chapter 7
- Start reading Chapter 9 of Core Java Volume 2
