CS 194: Distributed Systems
Security

Scott Shenker and Ion Stoica
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1776

1

Attacks

- **Interception (eavesdropping)**: unauthorized party gains access to service or data
- **Interruption (denial of service attack)**: services or data become unavailable
- **Modification**: unauthorized party changes the data or tampers with the service
- **Fabrication**: unauthorized party generate additional data or activity

2

Security Requirements

- **Authentication**: ensures that sender and receiver are who they are claiming to be
- **Data integrity**: ensure that data is not changed from source to destination
- **Confidentiality**: ensures that data is red only by authorized users
- **Non-repudiation**: ensures that the sender has strong evidence that the receiver has received the message, and the receiver has strong evidence of the sender identity (not discussed here)
  - The sender cannot deny that it has sent the message and the receiver cannot deny that it has received the message

3

Outline

- ➢ Cryptographic Algorithms (Confidentiality and Integrity)
- Authentication

4

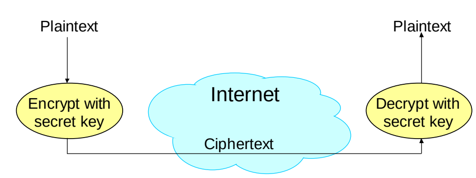
Cryptographic Algorithms

- Security foundation: cryptographic algorithms
  - Secret key cryptography, Data Encryption Standard (DES)
  - Public key cryptography, RSA algorithm
  - Message digest, MD5

5

Symmetric Key

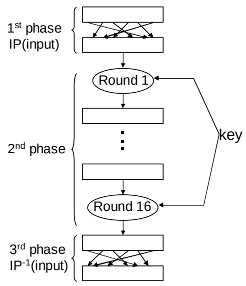
- Both the sender and the receiver use the same secret keys



6

Data Encryption Standard (DES)

- DES encrypts a 64-bit block of plain text using a 64-bit key
- Three phases
  1. Permute the 64 bits in the block
  2. Apply a given operation 16 times on the 64 bits
  3. Permute the 64 bits using the inverse of the original permutation



7

Initial Permutation (IP)

- IP: bit 58 of input becomes 1st bit, bit 50 becomes 2nd bit, etc

58 50 42 34 26 18 10 2 60 52 44 36 28 20 12 4
62 54 46 38 30 22 14 6 64 56 48 40 32 24 16 8
57 49 41 33 25 17 9 1 59 51 43 35 27 19 11 3
61 53 45 37 29 21 13 5 63 55 47 39 31 23 15 7

- IP^{-1}: inverse of IP, e.g., IP(1) = 58, IP^{-1}(58) = 1

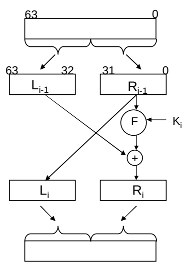
40 8 48 16 56 24 64 32 39 7 47 15 55 23 63 31
38 6 46 14 54 22 62 30 37 5 45 13 53 21 61 29
36 4 44 12 52 20 60 28 35 3 43 11 51 19 59 27
34 2 42 10 50 18 58 26 33 1 41 9 49 17 57 25

8

2nd Phase: Operation In Each Round

- Key *K* is 64 bits
- 16 rounds
- Each round *i* select a 48 bit key K_i from the original 64 bit key *K*. Perform (*F* is a given function):

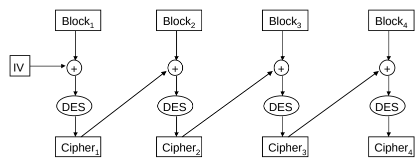
$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$



9

Encrypting Larger Messages

- Initialization Vector (IV) is a random number generated by sender and sent together with the ciphertext



10

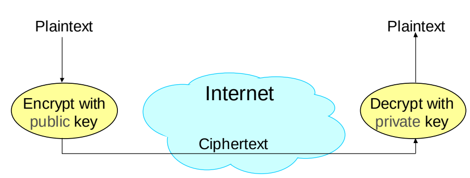
DES Properties

- Provide confidentiality
  - No mathematical proof, but practical evidence suggests that decrypting a message without knowing the key requires exhaustive search
  - To increase security use triple-DES, i.e., encrypt the message three times

11

Public-Key Cryptography: RSA (Rivest, Shamir, and Adleman)

- Sender uses a public key
  - Advertised to everyone
- Receiver uses a private key



12

Generating Public and Private Keys

- Choose two large prime numbers p and q (~ 256 bit long) and multiply them: $n = p*q$
- Chose encryption key e such that e and $(p-1)*(q-1)$ are relatively prime
- Compute decryption key d, where

  $d = e^{-1} \bmod ((p-1)*(q-1))$

  (equivalent to $d*e = 1 \bmod ((p-1)*(q-1))$)
- Public key consist of pair (n, e)
- Private key consists of pair (n, d)

13

RSA Encryption and Decryption

- Encryption of message block m:
  - $c = m^e \bmod n$
- Decryption of ciphertext c:
  - $m = c^d \bmod n$

14

Example (1/2)

- Choose p = 7 and q = 11 → n = p*q = 77
- Compute encryption key e: (p-1)*(q-1) = 6*10 = 60 → chose e = 13 (13 and 60 are relatively prime numbers)
- Compute decryption key d such that 13*d = 1 mod 60 → d = 37 (37*13 = 481)

15

Example (2/2)

- n = 77; e = 13; d = 37
- Send message block m = 7
- Encryption: $c = m^e \bmod n = 7^{13} \bmod 77 = 35$
- Decryption: $m = c^d \bmod n = 35^{37} \bmod 77 = 7$

16

Properties

- Confidentiality
- A receiver B computes n, e, d, and sends out (n, e)
  - Everyone who wants to send a message to A uses (n, e) to encrypt it
- How difficult is to recover d ? (Someone that can do this can decrypt any message sent to B!)
- Recall that

  $d = e^{-1} \bmod ((p-1)*(q-1))$
- So to find d, you need to find primes factors p and q
  - This is provable very difficult

17

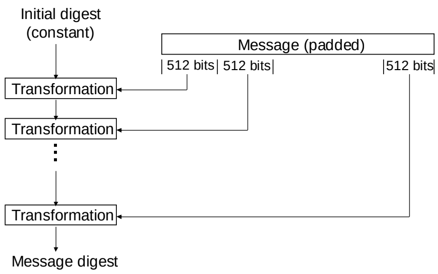
Message Digest (MD) 5

- Can provide data integrity and non-repudation
  - Used to verify the authentication of a message
- Idea: compute a hash on the message and send it along with the message
- Receiver can apply the same hash function on the message and see whether the result coincides with the received hash

18

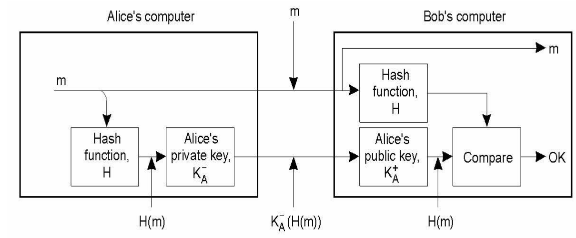
Message Digest Operation

- Transformation contains complex operations (see textbook)



Digital Signature

- In practice someone cannot alter the message without modifying the digest
  - Digest operation very hard to invert
- Encrypt digest with sender's private key
- K_A^-, K_A^+: private and public keys of A



Digital Signature Properties

- Integrity: an attacker cannot change the message without knowing A's private key
- Confidentiality: if needed, encrypt message with B's public key

Outline

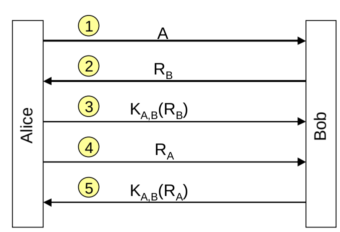
- Cryptographic Algorithms (Confidentiality and Integrity)
- ➢ Authentication

Authentication

- Goal: Make sure that the sender an receiver are the ones they claim to be
- Solutions based on secret key cryptography (e.g., DES)
  - Three-way handshaking
  - Trusted third party (key distribution center)
- One solution based on public key cryptography (e.g., RSA)
  - Public key authentication

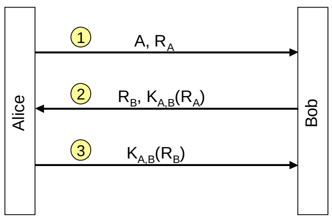
Authentication

- Authentication based on a shared secret key
  - A, B: sender and receiver identities
  - $K_{A,B}$: shared secret key
  - R_A,R_B: random keys exchanged by A and B to verify identities



"Optimization"

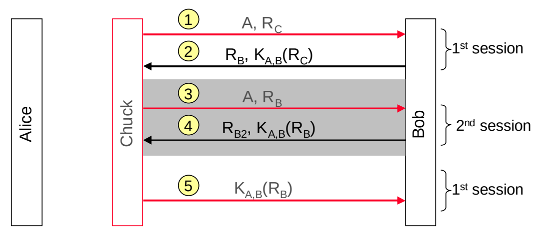
- Is this authentication protocol secure?



25

Reflection Attack

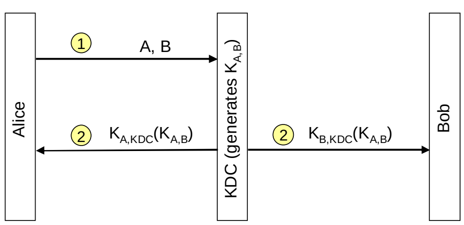
- An attacker (Chuck) can fool Bob in believing that he is Alice!



26

Authentication using KDC (Basic Protocol)

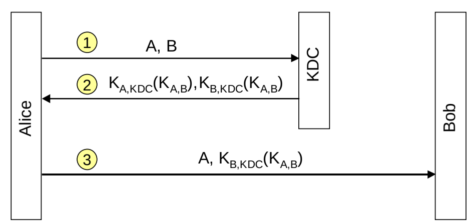
- KDC – Key Distribution Center
- Maintain only N keys in the system: one for each node



27

Authentication using KDC (Ticket Based)

- No need for KDC to contact Bob

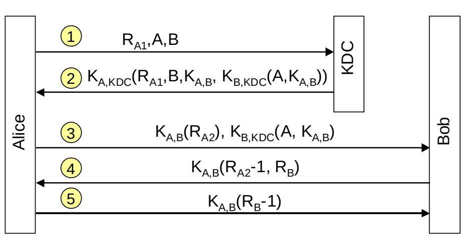


- Vulnerable to replay attacks if Chuck gets hold on $K_{B,KDC}^{old}$

28

Authentication using KDC (Needham-Schroeder Protocol)

- Relate messages 1 and 2: use challenge response mechanism
- R_{A1}, R_{A2}, R_B: nonces
  - **Nonce**: random number used only once to **relate** two messages

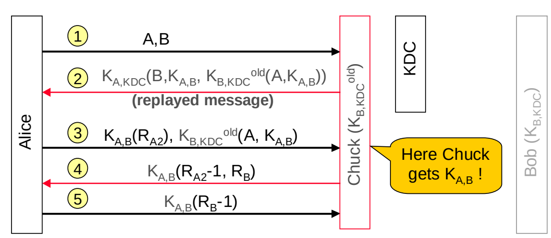


- Vulnerable to replay attacks if Chuck gets hold on $K_{A,B}$

29

What if R_{A1} is Missing?

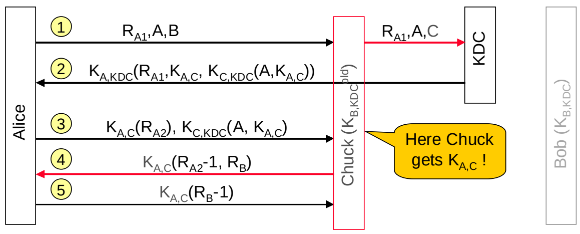
- Assume Chuck intercepted
  - $K_{A,KDC}(B,K_{A,B}, K_{B,KDC}^{old}(A,K_{A,B}))$
  - Knows $K_{B,KDC}^{old}$



30

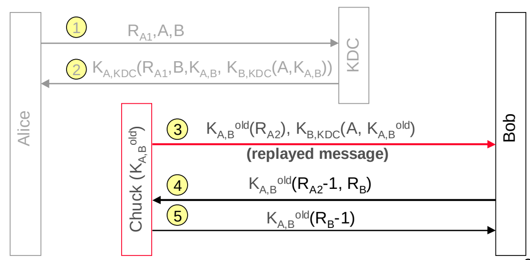
What if B is Missing from Message 2?

- Assume Chuck intercepts message 1



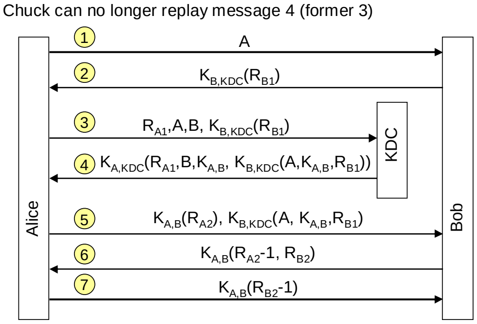
What if Chuck gets $K_{A,B}^{old}$?

- Assume Chuck intercepted
  - $K_{A,B}(R_{A2}), K_{B,KDC},(A,K_{A,B})$
  - Knows $K_{A,B}^{old}$



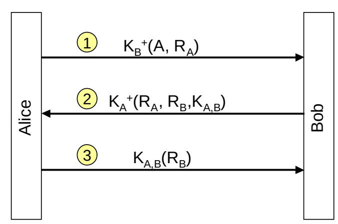
Defend Against leaking of $K_{A,B}$

- Message 5 (former 3) contains an encrypted nonce ($K_{B,KDC}(R_{B1})$) provided by Bob
- Chuck can no longer replay message 4 (former 3)



Authentication Using Public-Key Crypthography

- K_A^+, K_B^+: public keys



Secure Replicated Servers

- A client issues a request to a group of replicated servers
- Servers can be subject to Byzantine failures
- How does the client gets the answer?

Strawman Solution

- Servers gets replies from all servers…
- … and take majority voting
- Problem: client needs to authenticate each server (violates replication transparency)

Solution: Secret Sharing

- Secret sharing: none of users know the entire secret
- Intuition:
  - Assume we want to tolerate c failures (some of them can by Byzantine failures)
  - Need to combine responses such that c+1 correct servers are sufficient to get the correct response

37

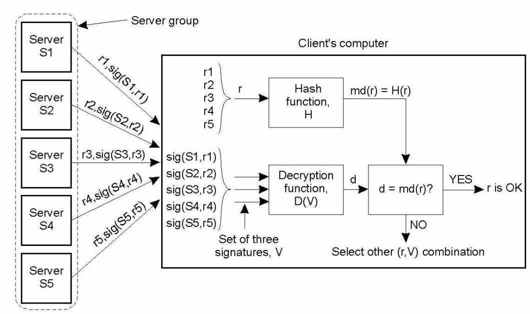
(k,n)-threshold Signature Scheme

- One public key K^+
- n **shares** of corresponding private keys, K_i^-, 1 <= i <= n
- Encrypted value v with each of private key shares, i.e., $v_i=K_i^-(v)$
- A client can decrypt value v using K^+ only if it knows at least k values of v_i

38

Solution: Secret Sharing

- Assume 5 replicated servers that tolerate 2 corrupted servers



39