Distributed Systems Security

Authentication Principles - 1

Prof. Steve Wilbur
s.wilbur@cs.ucl.ac.uk

Lecture Objectives

- Define authentication
- Identify types of protocols needed
- Identify threats
- Examine classic protocols based on SKC and identify weaknesses
**Authentication**

- Assurance that messages are from claimed originator
- Generally implies that original message has not been tampered with - message integrity
- Does not necessarily imply secrecy

- **Mutual authentication**: Two parties satisfy themselves of each others identity usually for long term (session or transaction) interaction
- **One-way Authentication**: One party is authenticated, eg. your login to Unix

**Cryptography**

- Use cryptography to achieve these functions
- Need keys to be distributed
- Key distribution different for PKC and SKC

- Need Key Distribution Protocols
- Need PKC and SKC protocols
- Also, need message oriented (single-ended) protocols and stream oriented (two-way) protocols
Basic Protocol Map

We will use the following map to keep our bearings as we explore the various protocols.

Key Distribution
- Shared Key Distribution
- Public Key Distribution

One-way Authentication
- One-way SKC
- One-way PKC

Two-way Authentication
- Two-way SKC
- Two-way PKC

Notation

A, B  Principals - “good guys” eg. Alice, Bob
U, V  Domains - “organisations” eg. UCL, IBM
E    Eavesdropper - “bad guys”, eg. Eve
S    Security server/service
Kx   Personal key of “x”
Ks   “Session” key
{Data}K Data encrypted with key K
SKC Key Distribution

For A, B to communicate securely they need to share a key.

This could be achieved by:
1. Providing pair-wise keys for all possible communications to all relevant parties.
2. Shared key selected by A and physically transmitted to B.
3. Third party selects key and physically delivers it to A and B.
4. If A and B already have secure communication, one party can select a new key and transmit using old key.
5. If A and B have secure communication to third party S, S can provide shared key via these secure connections.

For a population of N users, approach 1 requires \(\frac{N(N-1)}{2}\) keys.

May be just about feasible for small populations, but e.g. \(N=1,000\) needs about 500,000 keys and \(N=10,000\) needs about 50M keys.

Also, keys used for long periods become more vulnerable to cryptanalysis, so would need to change them periodically/frequently.

Physical delivery is generally inappropriate for routine key distribution in distributed systems, \(\therefore\) 2 and 3 are not suitable.
SKC Key Distribution - 3

- Approach 4 can be used, but needs an existing secure session
- Approach 5 is attractive. It requires that S shares a key with each member of population
- Thus, need to distribute N-1 keys, not N(N-1)/2
- Hierarchy of keys:
  - Session keys
  - End user/application personal keys shared with first level KDC
  - Repeated for higher level KDCs

Two-way SKC Authentication

- Assume A, B already have their own personal keys, Ka and Kb (do not know each other’s key)
- Each key is shared with trusted third party, S, such that S knows private keys of both A and B
- S known as Authentication Server (AS) or Key Distribution Centre (KDC)
- Protocol needed to distribute session key securely and mutually authenticate A and B
- Note: A and B both trust S, since S holds their personal keys
Two-Way Authentication

Protocol is broadly of form below
Ka, Kb are Personal Keys
Ks is the Session Key

Needham & Schroeder
SKC Protocol

Authentication with SKC
1. A→S: A, B, Ia1
2. S→A: [Ia1, B, Ks, [Ks, A]Kb]Ka
3. A→B: [Ks, A]Kb
4. B→A: [Ib]Ks
5. A→B: [f(Ib)]Ks
6. A↔B: [Data]Ks
Needham & Schroeder
SKC Protocol - 2

- Steps 1 to 3 are used to distribute session key to A and B
- Step 3 also indicates to B that S has only distributed this key to A (and B)
- \( \therefore \) Steps 3 to 5 deal with mutual authentication and live-ness indicating to both parties that message 3 was not a replay

- Can extend it to deal with multiple domains (see over)
- KDCs use similar protocol with a super-KDC they all trust

Needham & Schroeder
Multiple Domains

- As before, plus:
- \( Su, Sv \) are Authentication Servers
- \( Ka, Kb \) are secret keys of A, B known only to owner & Su, Sv resp.
- \( Ksas \) is conversation key between authentication servers

**Authentication with Secret Keys**

1. A \( \rightarrow \) Su: A, B, Ia1
   1a. Su \( \rightarrow \) Sv: \( \{Ks, B, A, Ia1\}Ksas \)
   1b. Sv \( \rightarrow \) Su: \( \{\{Ks, A\}Kb, Ia1, A\}Ksas \)
2. Su \( \rightarrow \) A: \( \{Ia1, B, Ks, \{Ks, A\}Kb \}Ka \)
3. A \( \rightarrow \) B: \( \{Ks, A\}Kb \)
4. B \( \rightarrow \) A: \( \{lb\}Ks \)
5. A \( \rightarrow \) B: \( \{l(Ib)\}Ks \)
Needham & Schroeder
Protocol Issues

u What is purpose of nonce?
u What forms of attack are possible?
  o Simple replay
  o Backward replay
  o Nonce attacks

Needham & Schroeder
Weakness

u Simple replay of msg 3 to B by E may cause confusion at A if session has closed, but otherwise is relatively harmless

u However, if an old session key has been compromised and E can suppress selected messages to A, then replay of msg 3 will cause B to have session with E thinking it is A

u Denning suggested use of timestamps to overcome this

u Because nonces give no indication of freshness of message
**Denning SKC Protocol**

- A, B are parties involved
- S is Authentication Server
- Ka, Kb are personal keys of A, B known only to owner and S
- T is timestamp
- Ks is “conversation key” or “session key” generated by S
- ”,” indicates message composition or concatenation

### Authentication with SKC

1. A->S: A, B
2. S->A: \(\{T, B, K_s, \{K_s, A, T\}K_b\}K_a\)
3. A->B: \[K_s, A, T\]K_b
4. B->A: \[I_b\]K_s
5. A->B: \[f(I_b)\]K_s
6. A<>B: \([\text{Data}]K_s\)

**Denning SKC Protocol - 2**

- Basically same protocol as Needham & Schroeder, except timestamp generated by S used instead of nonce
- Message considered valid if on receipt:
  \[|\text{Clock} - T| < \Delta t_1 + \Delta t_2\]
  where
  - \(\Delta t_1\) is max. allowed discrepancy between KDC and local clock
  - \(\Delta t_2\) is max. network delay
- Provided B’s personal key not compromised, only replay of message 3 is possible and timestamp thwarts this attack
Denning Protocol Issues

- Clocks must be synchronised, so need secure clock synchronisation protocol
- If recipient clock can be advanced, accidentally or by sabotage, protocol messages could be replayed again at a valid time
- Frequent clock synchronisation with KDC is one solution
- Neuman and Stubblebine [1993] proposed protocol to remove this requirement using nonces again

Neuman and Stubblebine SKC Protocol

- A, B are parties involved
- S is Authentication Server
- K_a, K_b are personal keys of A, B known only to owner and S
- T_b is time limit for session key
- I_a, I_b are nonces
- K_s is “conversation key” or “session key” generated by S
- “,” indicates message composition or concatenation

Authentication with SKC
1. A→B: A, I_a
4. A→B: [A, K_s, T_b]K_b, [I_b]K_s
5. A↔B: [Data]K_s
Neuman and Stubblebine
SKC Protocol - 2

- Impervious to clock sabotage or session key cracking
- Assumes Ka and Kb not compromised
- Nonce Ia is bound to Ks within short space of time via protocol synchronisation not clock sync.
- Similarly, Ib is bound to Ks
- Tb provides a validity period for the session key
- \{A, Ks, Tb\}Kb acts as a ticket or authenticator for A with B, indicating session key and validity period

Neuman and Stubblebine
SKC Protocol - 3

- Can avoid repeated KDC exchanges by use of ticket within validity period
- Tb is relative to B’s clock so no clock sync. issue

**Creation of new session**
1. A->B: \{A, Ks, Tb\}Kb, Ia’
2. B->A: Ib’, \{Ia’\}Ks
3. A->B: \{Ib’\}Ks
4. A<->B: \{Data\}Ks
Single-ended Authentication

- In some applications parties are not necessarily available simultaneously, e.g. e-mail
- Ideally, we would like to have mutual authentication so that A knows only B can read message and B knows that it could only have come from A
- If not possible to have 2-way dialogue, assurances may be weaker
- Note: this is not strictly one-way authentication

Needham & Schroeder Single-ended Authentication

- Single-ended system, e.g. e-mail
- As before, plus:
- TS is sender’s timestamp
- Sn is serial number of message fragment
- Recipient must check for possible replays (via max. clock asynchrony and estimated delivery delay)

Authentication with Secret Keys
1. A→S: A, B, Ia1
2. S→A: {Ia1, B, Ks, {Ks, A}Kb }Ka
3. A→B: {Ks, A}Kb, {TS,S1, Mess1}Ks, {S2,Mess2}Ks, ...

Page 12
E-mail Protocols

- This can form the basis of secure e-mail protocols
- However, e-mail is often distributed by the originator to several recipients so there are additional threats and additional service requirements
- What might they be?

- See Pretty Good Privacy (PGP) and Privacy Enhanced Mail (PEM) for more details

Further Reading

  - Key Distribution: pp 141-149
  - Authentication: pp 303-311
  - Pretty Good Privacy: pp 356-374

  - Privacy Enhanced Mail: pp 422-426
Further Reading - 2


- D Denning, “Cryptography and Data Security”, Addison-Wesley, 1982