Mutual Authentication

Michael Huth
M.Huth@doc.ic.ac.uk

www.doc.ic.ac.uk/~mrh/430/
Introduction

- **AUTHENTICATION**
  - Identification: A claims to have a certain identity
  - Verification: claim checked by B

- **ONE-WAY AUTHENTICATION**
  One party wants to authenticate another, e.g. host wants to authenticate user, receiver wants some assurance that sender is who it claims to be

- **MUTUAL AUTHENTICATION**
  A communicates with B and no one else (i.e. not with an intruder), and vice versa; usually for a session, so often combined with and needed for Session Key Exchange

  **One Correctness Criterion:**
  Mutual Authentication achieved if there is a Session Key K such that A believes (knows?) that K is a shared secret key for communication with B. Similar for B

  **Stronger Correctness Criterion:**
  Mutual Authentication achieved if A believes that B believes that K is a shared secret key for communication with A. Similar for B
Introduction Contd.

Many authentication protocols
- Different cryptosystems, different applications.
- How to compare them?
- What guarantees do they offer?

Protocol Weaknesses
- History of “weaknesses” in published security protocols
- Weaknesses often subtle
- Need for formal methods

Common Modeling Assumptions
- Good cryptography. Correct implementation
- No failures/exceptions
- Trustworthy parties
- No unauthorized release of secrets
- Messages adhere to specified structure
- Traffic Analysis & Denial of Service often ignored
Replay Attacks

- **Simple Replay**
  Copy message, replay it later

- **Suppressed Replay**
  Remove message, replay it later

- **Timed Replay**
  Copy message, replay it within expiry time

- **Replay to Sender**
  Replay message back to sender (Reflection Attack, e.g. for challenge-response protocols)

**INTERLEAVED RUNS**

- Combine messages from different (sometimes concurrent) runs of same protocol

**FRESHNESS/TIMELINESS**

- **Sequence Numbers**
  Need to be maintained for every channel (e.g. guessing Initial Sequence Numbers for TCP/IP spoofing attacks)

- **Timestamps**
  Accept message if timestamp recent enough. May need synchronised clocks. Expiry times sometimes used for synchronization up to $t > 0$ (milli)seconds.

- **Nonces**
  Typically randomly generated numbers, sent in a request, returned in a reply (e.g. in challenge-response protocols).
Authentication Protocols: our notation

- **Communicants**
  - Alice (1st Party - Initiator)
  - Bob (2nd Party)
  - Trent (Trusted 3rd Party)

- **Only**: AliceTrent
  Message is encrypted with shared symmetric key known only to Alice & Trent. Encrypted messages will be coloured.
  **Note**: TrentAlice = AliceTrent

- **Only**: -1
  Message is encrypted with a secret session key
  \( E_{KUb} \) public key of b
  \( E_{KRb} \) private key of b

- **Only**: Bob
  Message is encrypted with Bob’s Public Key. Public Key fields end with PK.

- **Signed**: Alice
  Message is signed with Alice’s Private Signature Key.

- **Nonces**
  End with N (e.g. AliceN). Typically a random number

- **Timestamps**
  End with T (e.g. ExpiryT). Typically time will go down to seconds or even milliseconds).
Standard Notation

- **Communicants**
  - \(A, a\) - Alice (1st Party - Initiator)
  - \(B, b\) - Bob (2nd Party)
  - \(T, t\) - Trent (Trusted 3rd Party)

- **Only:** AliceBob
  - \(E_{Kab}(M)\) or \(\{M\}_{Kab}\)

- **Only:** \(-1\)
  - \(E_K(M)\) or \(\{M\}_K\)

- **Signed:** Alice
  - \(E_{KRa}(M)\) or \(\{M\}_{Ka}^{-1}\)

- **Nonces**
  - \(N_a\) (Alice's Nonce)

- **Timestamps**
  - \(T_a\) (generated by Alice)

- **A → B:**
  - \(ID_a, E_{Kab}[ID_a, K, T_a, E_{KRa}(M)]\)
Example: One-Way Message (Public-Key)

**From:** Alice
**Text:** Buy 100
**Signed:** Alice

\[ E_{KRa(ID_a, M)} \]  
Version 1

**Only:** Bob
**Text:** Buy 100
**Signed:** Alice

**Who:** Alice
**AlicePK:** 666
**ExpiresT:** 31 Dec 09
**Signed:** Trent

\[ E_{KUb(E_{KRa(M)})}, E_{Kt(ID_a, KUa, T_t)} \]  
Version 3

Network Security (N. Dulay & M. Huth)
Example: One-way authenticated message, using symmetric-key crypto

From: Alice
TalkTo: Bob
AliceN: 66

Only: AliceTrent
AliceN: 66
Key: -1

Only: BobTrent
From: Alice
Key: -1

Text: Buy 100
Arbitrated Digital Signature (Encrypted)

From: Alice
To: Bob
Signed: Alice

Only: Bob
Text: Buy 100
Signed: Alice

From: Alice
Trent T: 11/1/02 12:45
Signed: Trent

Only: Bob
Text: Buy 100
Signed: Alice

Network Security (N. Dulay & M. Huth)
Mutual Authentication (with Public Key)

AlicePK: 666

BobPK: 999

Only: Bob
Text: Hi Bob

Only: Alice
Text: Hi Alice
Man-in-the-Middle Attack

AlicePK: 666
MaxPK: 888
BobPK: 999
MaxPK: 888

Only: ?Bob?
Text: Hi Bob

Only: Bob
Text: Hi Bob

Only: ?Alice?
Text: Hi Alice

Network Security (N. Dulay & M. Huth)

Mutual Authentication (5.11)
Interlock Protocol: “some” protection against man-in-middle-attack

AlicePK: 666
MaxPK: 888
BobPK: 999
MaxPK: 888

Only: ?Bob? Signature ("Hi Bob")
Only: Bob ????
Only: ?Alice? Signature ("Hi Alice")
Only: Alice ????

Only: Bob Text: Hi Bob

Network Security (N. Dulay & M. Huth)
Mutual Authentication (5.12)
Andrew Secure RPC Handshake: exchange of fresh shared key

From: Alice
AliceN: 66

Only: Bob
BobAlice
BobN: 99

AliceN+1: 66 + 1

Key: –1

May also generate and transmit fresh nonce to be used as AliceN in future rounds as below

From: Alice
AliceN: 66

Only: Alice
AliceBob
AliceN: 66

Key: –1

BobN+1: 99 + 1

Only: –1
AliceN: 66
Mutual Authentication (5.14)

Needham-Schroeder: nonce-based mutual authentication (1978)

From: Alice
TalkTo: Bob
AliceN: 66

Only: AliceTrent
TalkTo: Bob
AliceN: 66
Key: -1

Only: BobTrent
TalkTo: Alice
Key: -1

Only: -1
BobN: 99

Only: -1
BobN-1: 99 - 1

Network Security (N. Dulay & M. Huth)
NS (Denning-Sacco Flaw)

From: Alice
TalkTo: Bob
AliceN: 66

Only: AliceTrent
TalkTo: Bob
AliceN: 66
Key: –1

Only: BobTrent
TalkTo: Alice
Key: –1 (Cracked, e.g. compromised old Session key)

Only: –1
BobN: 88

Only: –1
BobN–1: 88 - 1

Network Security (N. Dulay & M. Huth)
Denning-Sacco Timestamp Fix (1981)

From: Alice
TalkTo: Bob

Only: AliceTrent
TalkTo: Bob
CurrT: 16/1/02,14:59
Key: –1

Only: BobTrent
TalkTo: Alice
Key: –1
CurrT: 16/1/02,14:59

Only: –1
BobN: 99

Only: –1
BobN–1: 99 – 1

Network Security (N. Dulay & M. Huth)
Wide-Mouthed Frog: distribution of fresh shared key

From: Alice
Only: Trent
TalkTo: Bob
Key: 1
AliceT: 16/1/02 8:39

Only: Bob
TalkTo: Alice
Key: 1
TrentT: 16/1/02 8:40

1

2

Network Security (N. Dulay & M. Huth)
Mutual Authentication (5.17)
Wide-Mouthed Frog: Flaws

From: Alice
Only: TrentAlice
TalkTo: Bob
Key: –1
AliceT: 15/1/02 8:39

Only: BobTrent
TalkTo: Alice
Key: –1
TrentT: 15/1/02 8:40

From: Bob
Only: BobTrent
TalkTo: Alice
Key: –1
TrentT: 15/1/02 8:40

Only: AliceTrent
TalkTo: Bob
Key: –1
TrentT: 15/1/02 8:41

From: Alice
Only: AliceTrent
TalkTo: Bob
Key: –1
TrentT: 15/1/02 8:41

Only: BobTrent
TalkTo: Alice
Key: –1
TrentT: 15/1/02 8:42

Network Security (N. Dulay & M. Huth)

Mutual Authentication (5.18)
Wide-Mouthed Frog: Flaws Contd

Only: AliceTrent
TalkTo: Bob
Key: $-1$
TrentT: 15/1/02 8:41

Only: BobTrent
TalkTo: Alice
Key: $-1$
TrentT: 15/1/02 8:42
Neuman-Stubblebine: fresh symm. key by trusted server, mutual authentication

From: Alice
AliceN: 66
From: Bob
BobN: 99

Only: TrentBob
TalkTo: Alice
AliceN: 66
ExpiryT: 8:34

Only: AliceTrent
AliceN: 66
TalkTo: Bob
ExpiryT: 8:34
Key: −1

Only: BobTrent
TalkTo: Alice
ExpiryT: 8:34
Key: −1

BobN: 99

Mutual Authentication (5.20)
Neuman-Stubblebine Re-Authentication

Only: BobTrent
TalkTo: Alice
ExpiryT: 8:34
Key: -1
AliceN2: 77

Only: -1
AliceN2: 77
BobN2: 111

Only: -1
BobN2: 111

Network Security (N. Dulay & M. Huth)

Mutual Authentication (5.21)
Woo-Lam (Public-Key): one-way authentication of initiator of protocol

From: Alice
TalkTo: Bob

Who: Bob
BobPK: 999
Signed: Trent

Only: Bob
From: Alice
AliceN: 66

Who: Alice
AlicePK: 666
Signed: Trent

Only: Alice
From: Alice
TalkTo: Bob
AliceN: 66
Key: −1
Signed: Trent

Only: −1
BobN: 99
Woo-Lam (Message Exchanges)
Yahalom: distribution of fresh symm. Key by trusted server, mutual authentication

<table>
<thead>
<tr>
<th>From: Alice</th>
<th>From: Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>AliceN: 66</td>
<td>TrentBob</td>
</tr>
<tr>
<td>TalkTo: Alice</td>
<td></td>
</tr>
<tr>
<td>AliceN: 66</td>
<td></td>
</tr>
<tr>
<td>BobN: 99</td>
<td></td>
</tr>
</tbody>
</table>

| Only: TrentBob |
| TalkTo: Alice |
| AliceN: 66 |
| BobN: 99 |
| Key: -1 |

| Only: BobTrent |
| TalkTo: Alice |
| AliceN: 66 |
| BobN: 99 |
| Key: -1 |

Network Security (N. Dulay & M. Huth)
How to Verify Timeliness of a Timestamp?

- Given a time $T$ received in a message, how do we check it for timeliness?
Summary

- Cryptographic Protocols are very hard to get right. Subtle errors abound.
- If you modify a protocol to shield against known attack, how do you know the modification does not introduce new attacks?
- Important to state assumptions & goals explicitly.
- Keep protocol as simple as possible.
- Know why you are encrypting.
- Know when to sign and when to encrypt and which to do first.
- Take care about freshness and time management.
- Don't assume a message has a particular format unless you can verify it.
- Avoid binary messages with multiple interpretations, e.g. paradox attack of Neuman-Stubblebine protocol.
- Don't sign or decrypt data for someone else.
Summary Continued

Scalability. Will the protocol scale to many parties?

Known Key attack: Future runs of protocol should not fail if old session key is cracked (e.g. Denning-Sacco attack of NS)

Timestamps: ideally, should be verified by generator only

Global clock services need to be secure

What assurances do we have at the end of the authentication step?

Do not send critical data under the session key before both parties are assured

Is the re-authentication process simple?

Watch out for parts of message being used for different messages

Man-in-the-Middle

Others: Cut-and-Paste, Reflection, Interleaved Runs