Lecture 8 - Applied Cryptography

CMPSC 443 - Spring 2012
Introduction Computer and Network Security
Professor Jaeger
www.cse.psu.edu/~tjaeger/cse443-s12
Cryptographic Protocols

• Secure distributed applications have some cryptographic protocols that define the constructions and procedures for communicating between two parties
  – E.g., SSL/TLS, IPsec, SSH, Kerberos
  – Each has a set of goals (e.g., confidentiality, integrity, authenticity, non-repudiation)
  – Defined for some set of assumed principals, trust, ...
  – Much of network security is focused on the design and application of these protocols

• Again, let's start with Alice and Bob
Basic (User) Authentication

• Bob wants to authenticate Alice’s identity
  – (is who she says she is)

\[
\text{pw}^A
\]

\[
\text{[Y/N]}
\]
Hash User Authentication

- Bob wants to authenticate Alice’s identity
  - (is who she says she is)

```
h(pw^A)
```

[Diagram]

Alice

([h(pw^A)])

[Y/N]

Bob
Challenge/Response User Authentication

- Bob wants to authenticate Alice’s identity
  - AKA, *digest authentication*

\[
[h(c+p^{wA})]
\]

Network Diagram:
- Alice
- [h(c+p^{wA})]
- [Y/N]
- Bob
- 1
- 2
- 3
User authentication vs. data security

• User authentication proves a property about the communicating parties
  – E.g., I know a password

• Data authentication ensures properties about the transmitted data
  – E.g., guarantees confidentiality of the data

• Now, let’s talk about the latter, data security
Simple Data Integrity?

- Alice wants to ensure any modification of the data in flight is detectable by Bob (integrity)
HMAC Integrity

- Alice wants to ensure any modification of the data in flight is detectable by Bob (integrity)

Alice \[ \rightarrow \] [d,HMAC(K^{AB}, d)] \rightarrow Bob
Signature Integrity

- Alice wants to ensure any modification of the data in flight is detectable by Bob (integrity)

\[
[d, \text{Sig}(K^{-A}_A, d)]
\]
Confidentiality

- Alice wants to ensure that the data is not exposed to anyone except the intended recipient (confidentiality)

\[ E(K^{AB}, d) \]
Confidentiality and Authenticity

- Alice often also must show that she was the actual source of the message (authenticity, including integrity) as well.

Alice \[ E(K^{AB},d) \] [HMAC(\(K^{AB},d\))] Bob
Confidentiality

• Alice wants to ensure that the data is not exposed to anyone except the intended recipient (confidentiality)
• But, Alice and Bob have *never met*!!!!

\[
\text{Alice} \quad \xrightarrow{1} \quad [E(k^x, d), E(K^+_B, k^x)] \quad \text{Bob}
\]

• Alice randomly selects key \( k^x \) to encrypt with
Real Systems Security

• The reality of the security is that 90% of the frequently used protocols use some variant of these constructs.
  – So, get to know them … they are your friends
  – We will see them (and a few more) over the semester

• They also apply to systems construction
  – Protocols need not necessarily be online
  – Think about how you would use these constructs to secure files on a disk drive (integrity, authenticity, confidentiality)
  – We will add some other tools, but these are the basics
Key Exchange

• A simple key exchange between Alice and Bob
• Assisted by Trent (trusted third party), who shares pair keys with both Alice and Bob

1) Alice → Trent : \{Bob\}k_{alice}

2) Trent → Alice : \{k_{session}\}k_{alice} \cdot \{k_{session}\}k_{bob}

3) Alice → Bob : \{k_{session}\}k_{bob}

• This is an OK protocol, but it has a couple of flaws
• Q: What are they?
Key Exchange

• A simple key exchange between Alice and Bob
• Assisted by Trent (trusted third party), who shares pair keys with both Alice and Bob

1) Alice → Trent : \{Bob\}k_{alice}
2) Trent → Alice : \{k_{session}\}k_{alice} \cdot \{k_{session}\}k_{bob}
3) Alice → Bob : \{k_{session}\}k_{bob}

• This is an OK protocol, but it has a couple of flaws
• Q: What are they?

   a) Bob does not know who he is talking to.
   b) Bob can fall victim to a \textit{replay attack}.  

Needham-Schroeder Protocol

1) Alice → Trent : \{Alice + Bob + rand_1\}
2) Trent → Alice : \{Alice + Bob + rand_1 + K_{AB} + \{Alice + K_{AB}\} K_{BT}\} K_{AT}
3) Alice → Bob : \{Alice + K_{AB}\} K_{BT}
4) Bob → Alice : \{rand_2, K_{AB}\}
5) Alice → Bob : \{rand_2 - 1\} K_{AB}

- NS protocol is the basis for many authentication and key agreement systems, e.g., Kerberos
- Addresses the problems in the preceding protocol
  - Use of \( rand_1 \) ensures that Alice is not receiving replay
  - Use of \( rand_2 \) ensures that Bob is not receiving replay
  - Alice is authenticated by \textit{ticket}
  - Specification of identities of Alice and Bob in request and ticket ensure that no ambiguity in identity (mutual auth.)
Needham-Schroeder Protocol (Public Key)

1) Alice → Trent : \{Alice + Bob\}
2) Trent → Alice : \{Bob^+ + Bob\}Trent^−
3) Alice → Bob : \{rand_1 + Alice\}Bob^+
4) Bob → Trent : \{Bob + Alice\}
5) Trent → Bob : \{Alice^+ + Alice\}Trent^−
6) Bob → Alice : \{rand_1 + rand_2\}Alice^+
7) Alice → Bob : \{rand_2\}Bob^+

- The public key version consists of messages to retrieve the public keys from the trusted third party.
- Addresses the problems in the preceding protocol
  - Use of \(rand_1\) ensures that Alice is not receiving replay.
  - Use of \(rand_2\) ensures that Bob is not receiving replay.
  - No ticket is necessary.
    - Trent’s lookup of public keys provides basis for secrets.
  - Specification of Alice, but not Bob in the messages.
    - But only Bob knows his private key -- is that enough?
Key Storage

• Q: Where do you put the keys/passwords that you use for your system?
  – Sticky note on your monitor?
  – File on disk (do you trust WinX to protect your files?)
  – Encrypted on disk (it will be in memory some time)

• Many system use secure secondary storage for keys
  – Smartcards
  – Cryptographic co-processors
  – Trusted Platform Module (TPM)
  – Passive authentication device

• Reality: most systems are broken by loss of keys
  – Cryptography almost never the source of compromise

• **Key escrow** - third party recovery of keys
Key Revocation

• Keys are generally useful for some period of time
  – Sometimes called key decay
  – Cause: cryptanalysis, increased exposure to (unknown) compromise, discovered time

• Q: how do you tell all the other principals/services and the keys are no longer valid?
  – In centralized system, it is often easy (ask central service)
  – In a decentralized system, it is much harder, particularly in the case of certificates (more of this later in PKI ...)

• Solutions
  – push - advertise (authenticated) list of revoked keys
  – pull - query continued validity of key
Take Away

• You have the tools for writing crypto protocols
  – Algorithms are well-understood
  – Must be careful to compose into protocols correctly
  – Issues of key storage and removal must be addressed

• Next week: Look at implementations of authentication