Introduction to Crypto Protocols : Features and Attacks

Security properties of a protocol

Security properties

- The main security properties are:
  - Secrecy/confidentiality
  - Authentication (entity, data origin)
  - Integrity
  - Non-repudiation
  - Anonymity
  - Fairness
  - Availability
  - Certified delivery

Confidentiality

- Secrecy protects against unauthorized disclosure of information
- Only intended parties will learn the info sent
  - Generally, all secrets are time-bound
  - Nothing remains secret forever
- A protocol preserves the secrecy of one of its parameters if it does not leak this parameter during the execution the protocol
- Parameters could be keys or any sensitive data
- Mechanism to use: encryption

Authentication

- Authentication means proving an identity over the network (principal authentication), or authenticity of data origin (message authentication).

- Authentication is sometimes taken to be of two types:
  - *Message authentication*: Ensuring, that a message received matches the message sent. Sometimes, it means a proof of the identity of the creator of the message.
  - *Entity/Principal authentication*: Corroborating that a principal is the one claimed. Timeliness is important.

Example

- Consider the following:

  A (user) ➔ B (IoT): CMD, $HMAC_{kAB}\{CMD\}$

- A's claim: CMD was received by B
- B's claim: CMD originated from A

Are these claims valid?

Authorization/access control

- The authorization property stipulates which principal has access to what resource or operation.
- It distinguishes between legal and illegal accesses.
- Legal principals are granted authorization to the resource/operation in question while illegal ones are denied access to the resource or operation.

Integrity

- Integrity is the property of ensuring that information will not be accidentally or maliciously altered or destroyed.
- Integrity protects against unauthorized creation, alteration or destruction of data.

- If it were possible for a corrupted message to be accepted, then this would show up as a violation of the integrity property and we would have to deem the protocol to be flawed.

Non-repudiation

- Non-repudiation is defined as the impossibility for one of the entities involved in a communication denying having participated in all or part of the communication.
- It provides a protection against false denial of having been involved in the communication.
- The general goal of non-repudiation is to collect, maintain, make available, and validate irrefutable evidence concerning a claimed event or action in order to resolve disputes about the occurrence or non-occurrence of that event or action.
- Non-repudiation is related to authentication but has strong proof requirements.

Anonymity

- The property of anonymity provides a principal with the ability to make anonymous transactions, which cannot be tracked by another principal.
- More generally, we say that if a system is anonymous over some set of events E, then it should have the following property.
  - Whenever an event from E occurs, then an observer, though he may be able to deduce that an event from E has occurred, will be unable to identify which event.
- In the case of e-commerce protocols, this property allows consumers to make anonymous purchases which cannot be tracked by a bank or a merchant to identify the purchaser.

Fairness

- In fair protocols, agents require protection from each other, rather than from an external hostile agent.
- In electronic contract signing, for instance, we will want to avoid one of the participants being able to gain some advantage over another by halting the protocol part-way through.
- Bob could, for example, refuse to continue after Alice has signed up, but before he has signed.
- Some efficient fair protocols are conceived to run between two agents and occasionally call upon a trusted third agent in case of disputes.

Availability

- This property deals with the availability of certain resources manipulated by the protocol.
- For instance, for a key-exchange protocol, we would be confident that a session will indeed be established.
- That is, if Alice requests the server to set up a session key between her and Bob, then the system must subsequently reach a state in which Alice and Bob both have knowledge of the fresh session key.
- Generally, to verify the availability property in crypto-protocols, we have to restrict the capabilities of the intruder. In particular, we cannot allow the intruder unlimited ability to kill messages.

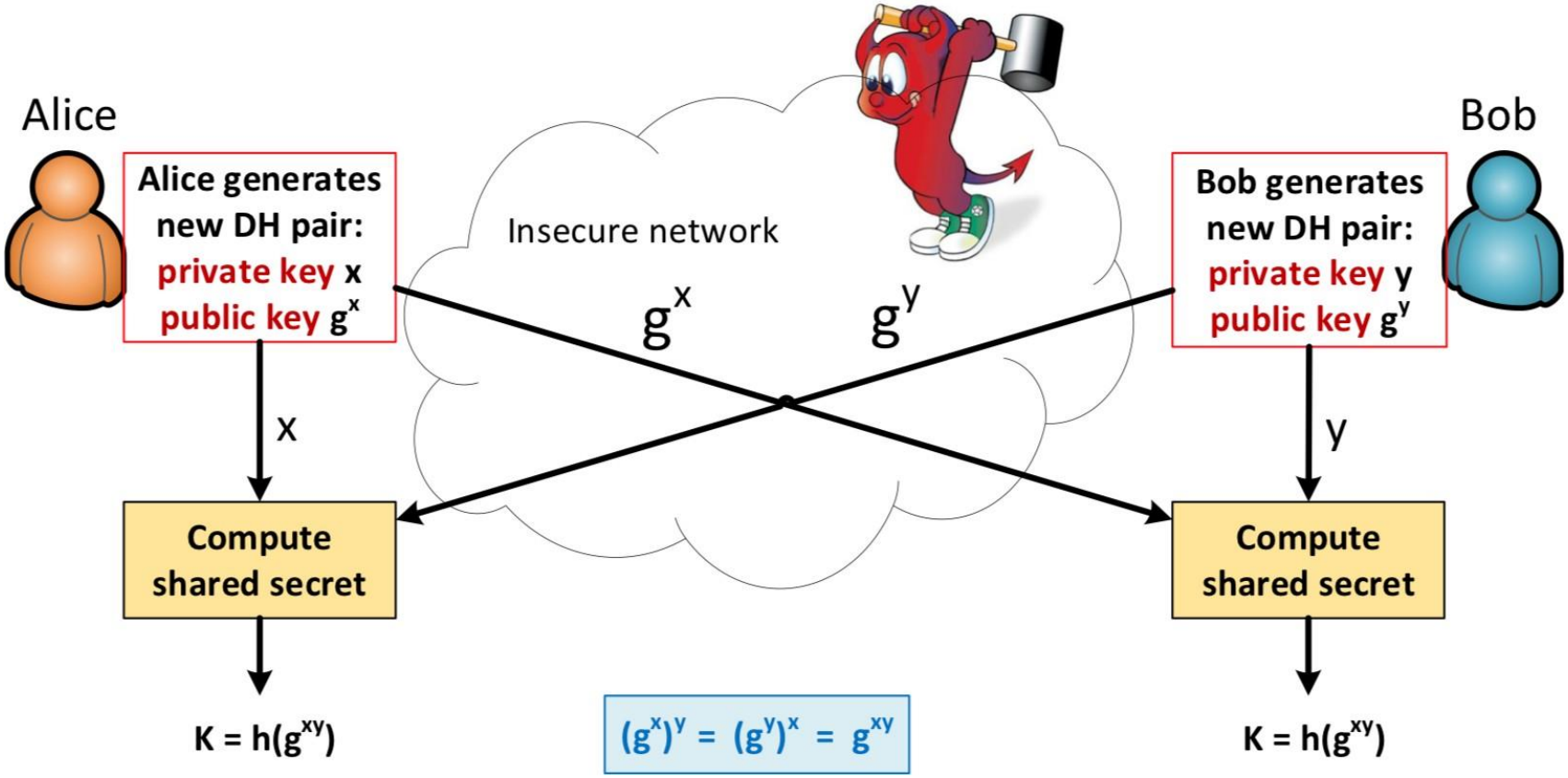
Goals in key establishment

- **Key establishment**: a process whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use

- **Good key**: The shared session key is a good key for A to use with B, only if A has the following assurances:
  - the key is fresh (**key freshness**)
  - the key is known only to: A, B, and mutually trusted parties (**key authentication / key confidentiality**)

- **Key confirmation**: Good key + proof that the intended principal has access to the key

Diffie-Hellman key exchange



DH problems

- Who has access to the established key?
- Do we achieve authentication? Key or message confidentiality? Integrity?

- Next step: authenticated DH

Example: key establishment

- STS (station-to-station) protocol

1. A → B: g^x
2. A ← B: g^y, $\{Sig_B(g^y, g^x)\}_{K_{AB}}$
3. A → B: $\{Sig_A(g^y, g^x)\}_{K_{AB}}$

- Analysis (from A's perspective):
  - The signature in message 2 can only be formed by B
  - Message 2 is not a replay (x is fresh)
  - Signature alone does not prove that B knows K_{AB}. That's why we need the encryption with K_{AB} (key confirmation)

- Another important aspect of a protocol: **connections** between messages
  - What connects above messages?

Protocol efficiency

- Two types:
  - Computational efficiency
    - Main cost: cryptographic algorithms (public/symmetric operations), number of operations
    - Potential bottleneck
  - Communications efficiency
    - Number of messages / rounds
    - Length of messages

Goal: forward secrecy

- Assume that attackers will store all past communications : including session establishment + messages in a session
- We also assumed before that session keys may leak sometimes
  - Communications protected by those keys will also leak as a consequence
- But a long-term key may also leak
  - This should not leak session keys established by the key before the compromise
  - Such a feature is called forward secrecy

Forward secrecy

- Generally achieved through the use of **ephemeral public keys**
  - Keys that are used for key-establishment only, and destroyed immediately afterwards
- For key-agreement protocols: easily achieved if the long-term key is used only to authenticate protocol messages (example: STS)
- For key transport protocols: cannot be achieved if the long-term key is used to encrypt session keys

Forward secrecy – more examples

- A key-transport protocol
  - K_T is an ephemeral public key chosen by A
  - $E_T(.)$ denotes encryption using K_T
  - h is a one-way hash function

1. A → B: K_T, N_A, $Sig_A(K_T, \mathbf{B})$
2. B → A: $E_T(K_{AB})$, $Sig_B(\mathbf{h}(K_{AB}), A, N_A)$

Are the above **bold** items significant?

MAC

- MAC-then-Encrypt: Compute the MAC on the cleartext, append it to the data, and then encrypt the whole (TLS)

- Encrypt-and-MAC: Compute the MAC on the cleartext, encrypt the cleartext, and then append the MAC at the end of the ciphertext (SSH)

- Encrypt-then-MAC: Encrypt the cleartext, then compute the MAC on the ciphertext, and append it to the ciphertext (IPSec)

Off-the-record (OTR) messaging

WPES 2004 paper

http://www.cypherpunks.ca/otr/

Off-the-record (OTR) messaging

- Goal: enable casual conversation over instant messaging (IM)
  - Assume Alice and Bob talking to a room – alone
  - No one else hears or knows what they say – or prove what they said
  - Use: forward secrecy + repudiability

- Protocol overview
  - Assume Alice, Bob know each other's long-term signature keys
  - Steps:
    1. Authenticated Deffie-Hellman
    2. Message transmission
    3. Key discard/renewal

OTR – steps

- Step 1
  - A ➔ B: g^x, $Sig_A(g^x)$
  - B ➔ A: g^y, $Sig_B(g^y)$
  - Shared secret: SS = hash(g^{xy}), EK = hash(SS), MK = hash(EK)
- Step 2
  - A ➔ B: $\{M\}_{EK}$, $MAC_{MK}(\{M\}_{EK})$
- Step 3
  - A ➔ B: $g^{x'}$, $MAC_{MK}(g^{x'})$
  - B ➔ A: $g^{y'}$, $MAC_{MK}(g^{y'})$
  - New secrets: SS′ = hash($g^{x'y'}$), EK′ = hash(SS′), MK′ = hash(EK′)
  - Discard: SS, EK, MK, x, y

OTR – subsequent messages

$$
\begin{aligned}
A \rightarrow B: &\quad g^{x_1} \\
B \rightarrow A: &\quad g^{y_1} \\
A \rightarrow B: &\quad g^{x_2}, E(M_1, k_{11}) \\
B \rightarrow A: &\quad g^{y_2}, E(M_2, k_{21}) \\
A \rightarrow B: &\quad g^{x_3}, E(M_3, k_{22})
\end{aligned}
$$

- Here $k_{ij} = \text{hash}(g^{x_i y_j})$