EITP20 – Secure Systems Engineering

Analysis and Design of Security Protocols (Part I)

Mohsen Toorani

Department of Electrical and Information Technology

February 14, 2020

Security models

From the adversarial power's perspective:

- Information-theoretic security: Adversary has unlimited computational resources.
- Computational security: Adversary has limited computational resources but is allowed to eavesdrop and gain some information.
    - It relies on unproven assumptions. Any unconditional proof requires proving that $P \neq NP$ (complexity classes).
    - Proofs are either in the standard model (the adversary is only limited by the amount of time and computational resources) or an idealized model (random oracle model, ideal cipher model, etc).

Computational security models

- Game-based models: A game is played between an adversary and a hypothetical challenger. A challenger is a probabilistic algorithm that generates all the keys and may respond to queries made by the adversary.
- Simulation-based models: Simulatability between an ideal system that can never be broken, and a real system. An arbitrary probabilistic polynomial-time (PPT) adversary interacts with each algorithm of the cryptosystem and also with an arbitrary PPT environment which represents all other entities that may have access to the algorithms.For security analysis, the adversary and the environment interact with both real and ideal systems, and their outputs are examined.

Security protocols

- A protocol consists of a set of rules or conventions that determine the exchange of messages between two or more principals.
- A security protocol is an abstract or concrete protocol that performs some security-related functions.
- A security protocol applies some security mechanisms, maybe as sequences of cryptographic primitives.
- There are dishonest actors and attackers that are not constrained to follow the protocol rules, and try to prevent protocol executions from reaching their security goals.
- Security protocols can be designated for specific functionalities (e.g. identification or key exchange), or they can be general protocols with many applications (multiparty computation).

Provable Security in reality

- Most techniques used in security proofs do not help in protocol design. A small change in the protocol may require constructing a new proof.
- Relationships between a security model, idealizations and assumptions in proofs, and the real-world attacks.
- Dependability of many proofs on idealized models.
- Security proofs might be incorrect. Many proofs have had flaws:
  - MT-authenticator : STOC'98
  - WC-MAKEP protocol : ASIACRYPT'01
  - JP-MAKEP protocol : FC'02
  - Conference key agreement: PKC'03
  - 3PAKE protocol : FC'05
  - CAFL model : ACISP'14
  - ...

Symbolic/Formal Models

- All the security models considered in previous slides, stem from the cryptographic community. The computational models are originated from papers by Goldwasser & Micali (1984) and Yao (1982).
- Symbolic/formal models stem from the formal methods community, and are credited back to the seminal paper of Dolev and Yao (1983), known also as the Dolev-Yao model.
- They can divided into model checking (finding attacks) or theorem proving (proving the correctness).
- Symbolic models typically offer a limited view of honesty and corruption.
    - This makes it hard to distinguish between the security provided by different protocols, and discerns benefits from storing long-term keys in tamper-proof modules or performing part of computations in a cryptographic coprocessor.

Formal models

- A language is formal when it has a well-defined syntax and semantics. There should be also a deductive system for determining the truth of statements.
- A model (or construction) is formal when it is specified in a formal language.
- Standard protocol notations are not formal and shall be formalized.

Formal modeling and analysis of protocols

- Goal: formally model protocols and their properties and provide a mathematically sound means to reason about these models.
- Basis: suitable abstraction of protocols.
- Analysis: with formal methods based on mathematics and logic, e.g. theorem proving.

Formal Methods in Information Security

A historical view (with focus on security protocols):

1983 Symbolic attacker model [Dolev-Yao]
Undecidability of secrecy problem for security protocols [Even-Goldreich]
1989 BAN logic for security protocol analysis [Burrows, Abadi, and Needham]
1995 MITM attack on Needham-Schroeder Public Key Protocol using the automatic verification tool Casper/FDR [Lowe]
1998 Inductive approach to verifying security protocols [Paulson]
2001 ProVerif: Efficient symbolic protocol verification tool [Blanchet]
Constraint-based decision procedure for bounded sessions [Millen-Shmatikov]
NP-Completeness of protocol insecurity for finite number of sessions [Rusinowitch-Turuani]
2005 Avispa protocol verification tools [Armando et al.]
2006 Scyther protocol verification tool [Cremers]
2012 Tamarin protocol verification tool [Meier, Schmidt]
⋮ ⋮

Designing a Protocol

- Now we try to design a key establishment protocol. We start from a simple protocol and perform several refinements to make it secure.
- We choose a common scenario:
  - A set of users, any two of whom may wish to establish a new session key for subsequent secure communications.
    Users are not necessarily honest!
  - There is an honest server.
    An honest server never cheats and never gives out user secrets.
- We consider a protocol with three roles: initiator role A, responder role B, and server role S.

Designing a protocol: Preliminaries

- In execution of a protocol, the roles are played by some agents/principals: a (Alice), b (Bob), s (server), i (intruder), . . .
- We denote by i an agent whose long-term keys are known to the adversary (but no agent knows this).
- Security goals for a key establishment protocol:
  - Key secrecy: At the end of the protocol, the session key K_{AB} is known to A and B, and possibly S, but to no other parties.
  - Key freshness: A and B know that K_{AB} is freshly generated.
- Formalization questions:
  - How to formalize the protocol steps and goals?
  - How to formalize *knowledge*, *secrecy*, *freshness*?

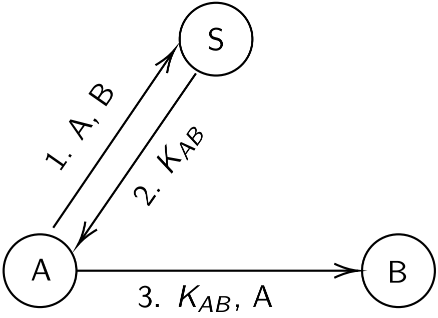
Protocol architectures

It is common to consider two cases for key establishment protocols:

- Key Transport protocols: The key is generated by one of the principals and transferred to the other(s).
- Key Agreement protocols: The principals both (all) contribute information which together establishes the key.

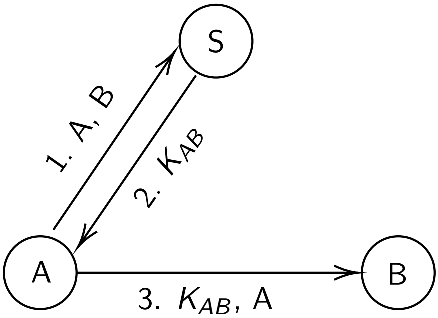
- Key transport protocols are typically associated with symmetric-key protocols with an online server.
- Key agreement protocols are typically associated with asymmetric-key protocols without a server.
- However, none of these situations is necessary.

Example protocol: Protocol 1



- A contacts S by sending the identities of A & B that are willing to share the session key.
- S sends the session key K_{AB} to A.
- A sends K_{AB} to B.

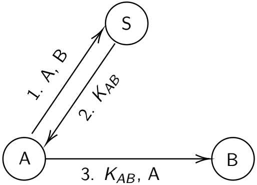
Example protocol: Alice & Bob notation



1. $A \rightarrow B : A, B$
2. $S \rightarrow A : K_{AB}$
3. $A \rightarrow B : K_{AB}, A$

($A \rightarrow B$ is not part of the communicated message)

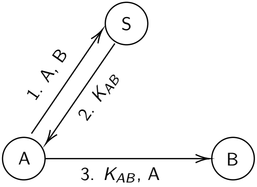
Conventions in protocol specifications



1. $A \rightarrow B : A, B$
2. $S \rightarrow A : K_{AB}$
3. $A \rightarrow B : K_{AB}, A$

- K_{AB} is simply a bit-string representing the session key.
  It contains no information about A and B.
- What if improperly formatted or no message is received?
  Only messages exchanged in a successful protocol run are specified.
- No specification of internal actions of principals, e.g.:
  S generates a fresh K_{AB} or stores it?
- Roles know that received messages are part of which protocol step.

Protocol 1: security issues



1. $A \rightarrow B : A, B$
2. $S \rightarrow A : K_{AB}$
3. $A \rightarrow B : K_{AB}, A$

- First problem with this protocol? Secrecy

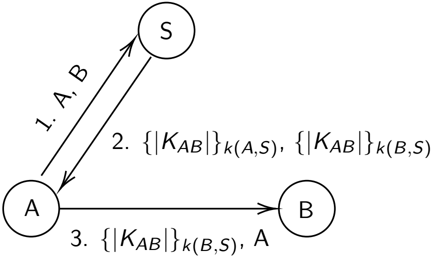
  The session key K_{AB} shall be transported to A and B, but not readable by other parties.

Assumption 1

The adversary can eavesdrop on all sent messages.

Countermeasure: using cryptography

Second Refinement: Protocol 2


S
1. A, B
2. $\{|K_{AB}|\}_{k(A,S)}$, $\{|K_{AB}|\}_{k(B,S)}$
A
B
3. $\{|K_{AB}|\}_{k(B,S)}$, A

- Refinement: using cryptography
- Assume that S pre-shares a secret key $k(U,S)$ with each user U:

  $k(A,S)$ with A

  $k(B,S)$ with B

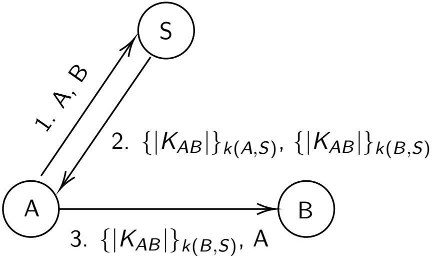
  and encrypts message 2 using these keys.
- Problems with this protocol? Eavesdropping? No

Assumption 2

Cryptography is perfect and encrypted messages may be read only by recipients who have the required decryption key.

Assumption 2 allows us to abstract away the details of cryptographic algorithms.

Protocol 2: security issues


S
1. A, B
2. $\{|K_{AB}|\}_{k(A,S)}, \{|K_{AB}|\}_{k(B,S)}$
A
B
3. $\{|K_{AB}|\}_{k(B,S)}$, A

- Problem: The principal's names are not bound to K_{AB}.
- Adversary may not only eavesdrop on sent messages, but also capture and modify them!

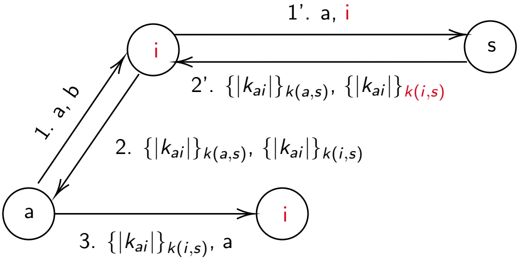
Another assumption on adversarial power

Assumption 3

The adversary has complete control over the network.

- The adversary is able to intercept messages on the network and send modified or new messages to anybody (under any sender name).
- The adversary has complete control of the network channel(s) over which protocol messages flow.
- Although there may be only a few messages involved in an honest protocol execusion, there are infinitely many variations where the adversary can participate.
- These variations involve an unbounded number of messages, and each must satisfy security requirements of the protocol.

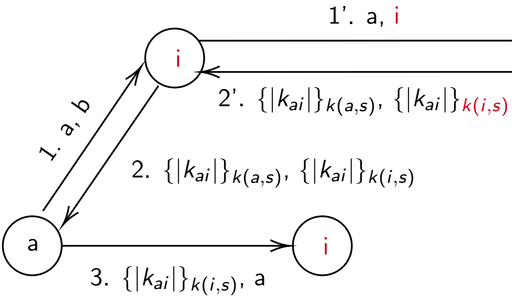
Protocol 2: security issues


1'. a, i
i
s
2'. $\{|k_{ai}|\}_{k(a,s)}, \{|k_{ai}|\}_{k(i,s)}$
1. a, b
2. $\{|k_{ai}|\}_{k(a,s)}, \{|k_{ai}|\}_{k(i,s)}$
a
i
3. $\{|k_{ai}|\}_{k(i,s)}$, a

A example Unknown Key-Share (UKS) attack on protocol 2:

- Intruder i modifies message 1 from a to s. The server generates two encrypted session keys including k_{ai} which is encrypted with key $k(i, s)$ known by i.
- a cannot distinguish between encrypted messages meant for other agents so will not detect and will believe the protocol was successfully completed with b.
- The intruder (i) who knows k_{ai} can masquerade as b and learn all information that a sends intended for b.

Protocol 2: security issues


1'. a, i
s
i
2'. $\{|k_{ai}|\}_{k(a,s)}, \{|k_{ai}|\}_{k(i,s)}$
1. a, b
2. $\{|k_{ai}|\}_{k(a,s)}, \{|k_{ai}|\}_{k(i,s)}$
a
i
3. $\{|k_{ai}|\}_{k(i,s)}$, a

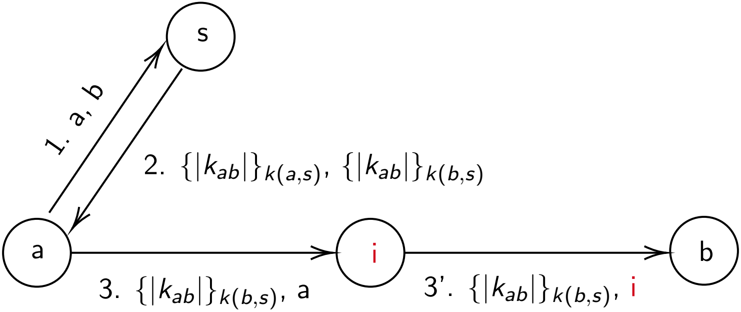
The above attack succeeds only if i is a legitimate user of the system, registered with s.

Assumption 4

The adversary may be a legitimate protocol participant (an insider), or an external party (an outsider), or a combination of both.

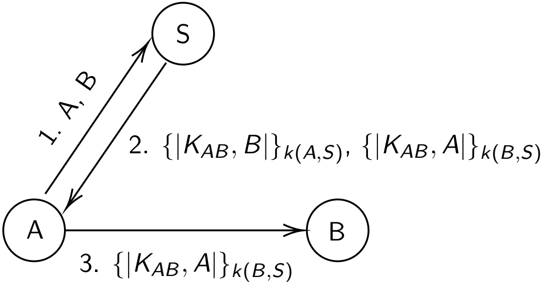
Protocol 2: another attack

Here is another UKS attack on protocol 2 that is even easier than the previous attack scenario: The intruder simply modifies message 3 and sends message 3' to b which includes his name i (or any other entity's name):


s
1. a, b
2. $\{|k_{ab}|\}_{k(a,s)}, \{|k_{ab}|\}_{k(b,s)}$
a
i
b
3. $\{|k_{ab}|\}_{k(b,s)}$, a
3'. $\{|k_{ab}|\}_{k(b,s)}$, i

Third refinement: Protocol 3

- To thwart the previous attack, names of principals who shall share K_{AB} must be bound cryptographically to the session key.


S
1. A, B
2. $\{|K_{AB}, B|\}_{k(A,S)}$, $\{|K_{AB}, A|\}_{k(B,S)}$
A
B
3. $\{|K_{AB}, A|\}_{k(B,S)}$

- Improvement: An adversary cannot attack the protocol by eavesdropping or modifying the messages sent between honest parties (two previous attacks fail).
- But there are still other problems!

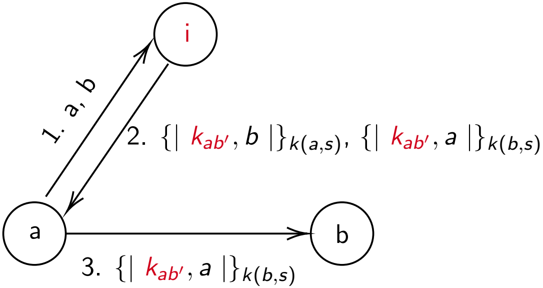
Replay attacks

- A class of attacks will be feasible if old keys (or other security-related data) can be replayed in subsequent sessions. (Replay attack comes from Assumptions 1 and 2)
- Difference in quality between long-term keys and session keys.
- Reasons for using session keys:
  - Key distribution problem: $O(n^2)$ keys needed for n principals.
  - Encrypted messages are vulnerable to attack (cryptanalysis).
  - Communications in different sessions should be separated. It should not be possible to replay messages from previous sessions.

Assumptions 4

The adversary is able to obtain the session key K_{AB} used in any sufficiently old previous run of the protocol.

Protocol 3: Replay Attack and Session Key Compromise

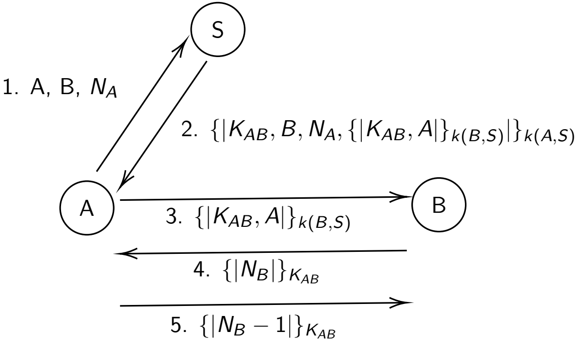

i
1. a, b
2. $\{\mid k_{ab'}, b \mid\}_{k(a,s)}, \{\mid k_{ab'}, a \mid\}_{k(b,s)}$
a
b
3. $\{\mid k_{ab'}, a \mid\}_{k(b,s)}$

- i masquerades as s and replays $k_{ab'}$ (an old key from a previous session between a and b).
  - By Assumptions 1 and 2, the adversary would know and replay the encrypted messages in which $k_{ab'}$ was transported to a and b.
- After the protocol execution, adversary can decrypt, modify or insert messages encrypted with $k_{ab'}$: No confidentiality or integrity!
  - By Assumption 4, the adversary can be expected to know kab'.

Protocol 3: Thwarting the replay attack

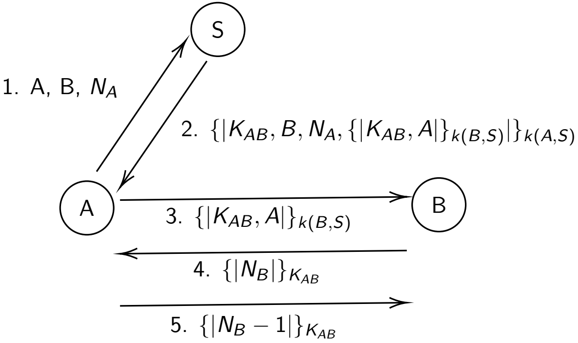
- The replay attack can still be regarded as successful even if the adversary has not obtained the value of $k_{ab'}$:
  - i gets a and b to accept an old session key, and then replay messages protected by kab' sent in a previous session.
  - Provided that a and b do not check the key, and only follow the protocol!
- Various techniques might be used to defend against replay attacks, e.g. incorporating challenge-response.
- A **nonce** is a random value generated by a principal to preserve the freshness.

Fourth refinement: Protocol 4


S
1. A, B, N_A
2. $\{|K_{AB}, B, N_A, \{|K_{AB}, A|\}_{k(B,S)}|\}_{k(A,S)}$
A
B
3. $\{|K_{AB}, A|\}_{k(B,S)}$
4. $\{|N_B|\}_{K_{AB}}$
5. $\{|N_B - 1|\}_{K_{AB}}$

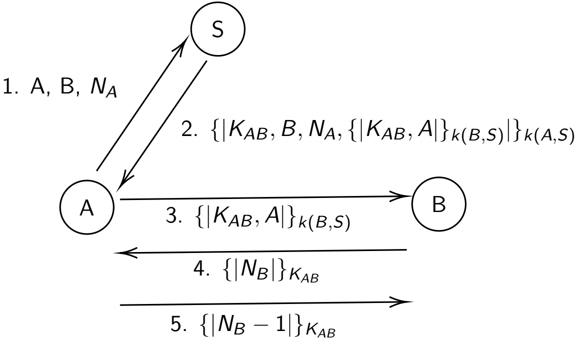
- A sends her nonce N_A to S with the request for a new key.
- If the same value is received from S with the session key, then A can deduce that the key has not been replayed.
- This is true if the session key and nonce are bound together so that only S can compose such a message.

Protocol 4


S
1. A, B, N_A
2. $\{|K_{AB}, B, N_A, \{|K_{AB}, A|\}_{k(B,S)}|\}_{k(A,S)}$
A
B
3. $\{|K_{AB}, A|\}_{k(B,S)}$
4. $\{|N_B|\}_{K_{AB}}$
5. $\{|N_B - 1|\}_{K_{AB}}$

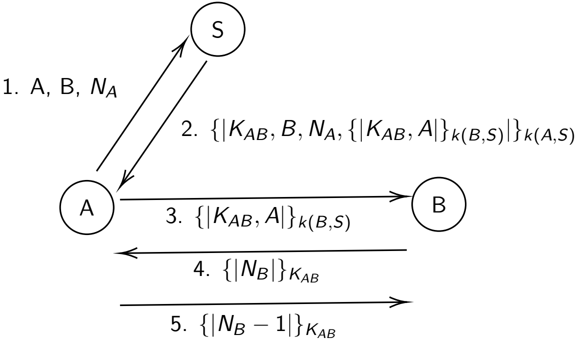
- If the encrypted key for B is included in the encrypted part of A's message, then A assures that it is fresh.
- A passes such assurance to B with an extra handshake:
    - B generates nonce N_B, encrypts it with K_{AB}, and sends it to A as message 4.
    - A uses K_{AB} to encrypt $N_B - 1$ (for avoid replaying message 4), and sends it to B as message 5.

Protocol 4 (Needham-Schroeder)


S
1. A, B, N_A
2. $\{|K_{AB}, B, N_A, \{|K_{AB}, A|\}_{k(B,S)}|\}_{k(A,S)}$
A
B
3. $\{|K_{AB}, A|\}_{k(B,S)}$
4. $\{|N_B|\}_{K_{AB}}$
5. $\{|N_B - 1|\}_{K_{AB}}$

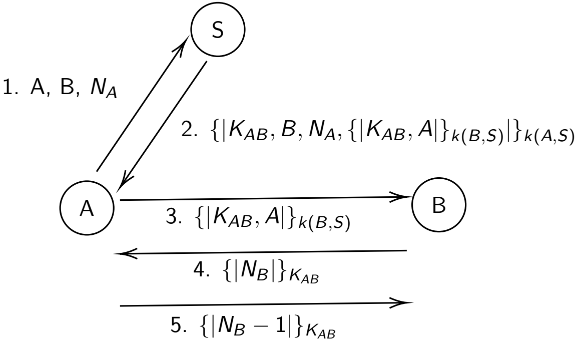
- Protocol 4 is actually a well-known security protocol: Needham-Schroeder with Conventional Keys
- Published by Needham and Schroeder in 1978, basis for several other protocols.

Protocol 4: security issues

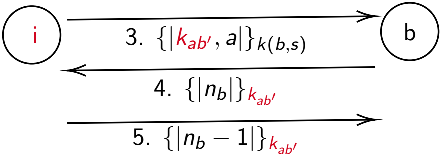

S
1. A, B, N_A
2. $\{|K_{AB}, B, N_A, \{|K_{AB}, A|\}_{k(B,S)}|\}_{k(A,S)}$
A
B
3. $\{|K_{AB}, A|\}_{k(B,S)}$
4. $\{|N_B|\}_{K_{AB}}$
5. $\{|N_B - 1|\}_{K_{AB}}$

- It is vulnerable to an attack due to Denning and Sacco in 1981.
- Problem: assumption that only A can form a correct reply to message 4 from B.
- Since A is any user (including the adversary that could know the value of old session keys), this assumption is unrealistic!

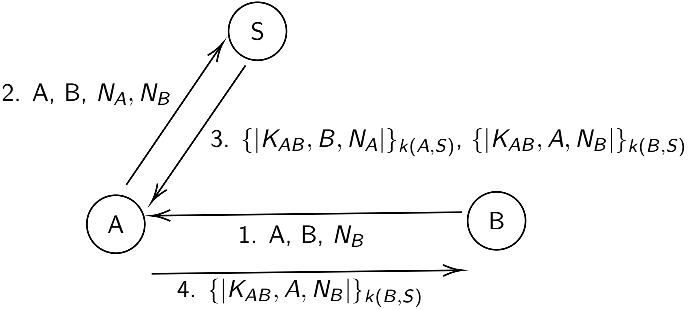
Denning–Sacco attack on Protocol 4


S
1. A, B, N_A
2. $\{|K_{AB}, B, N_A, \{|K_{AB}, A|\}_{k(B,S)}|\}_{k(A,S)}$
A
B
3. $\{|K_{AB}, A|\}_{k(B,S)}$
4. $\{|N_B|\}_{K_{AB}}$
5. $\{|N_B - 1|\}_{K_{AB}}$

Adversary masquerades as a and convinces b to use old key ($k_{ab'}$):

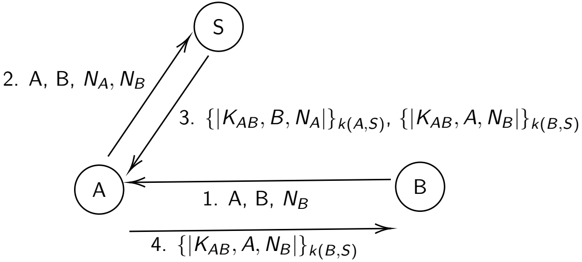


Fifth refinement: Protocol 5


S
2. A, B, N_A, N_B
3. $\{|K_{AB}, B, N_A|\}_{k(A,S)}, \{|K_{AB}, A, N_B|\}_{k(B,S)}$
A
B
1. A, B, N_B
4. $\{|K_{AB}, A, N_B|\}_{k(B,S)}$

- Refinement: Both B and A send their challenges to S.
- Protocol 5 is initiated by B who sends his nonce N_B to A.
- A adds her nonce N_A and sends both to S (message 2).
- S sends K_{AB} in separate messages for A and B, which can be verified as fresh by the corresponding recipients.

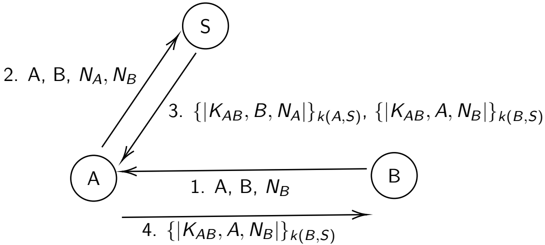
Key confirmation


S
2. A, B, N_A, N_B
3. $\{|K_{AB}, B, N_A|\}_{k(A,S)}$, $\{|K_{AB}, A, N_B|\}_{k(B,S)}$
A
B
1. A, B, N_B
4. $\{|K_{AB}, A, N_B|\}_{k(B,S)}$

Do we achieve more than protocol 4 using fewer messages?

- In protocol 4, A could verify that B has actually received the key (key confirmation). This property was achieved since B uses the key in message 4, assuming that $\{|N_B|\}_{K_{AB}}$ cannot be formed without knowledge of K_{AB}.
- In protocol 5, A or B can not finally deduce that its partner has received K_{AB}.

Is protocol 5 secure?


S
2. A, B, N_A, N_B
3. $\{|K_{AB}, B, N_A|\}_{k(A,S)}, \{|K_{AB}, A, N_B|\}_{k(B,S)}$
A
B
1. A, B, N_B
4. $\{|K_{AB}, A, N_B|\}_{k(B,S)}$

- It prevents all the attacks shown so far (assuming that the deployed cryptographic algorithms and S work correctly).
- But, the security of a protocol must always be considered according to its goals and assumptions.
- We must formalize (and prove) the security!
- Good news: A variant of protocol 5 (3PKD protocol) can be proven secure.

3PKD Protocol

1. $A \rightarrow B : A, B, N_A$
2. $B \rightarrow S : A, B, N_A, N_B$
3. $S \rightarrow A : N_B, E_{k(A,S)}(K_{AB}), M_{k'(A,S)}(A, B, N_A, N_B, E_{k(A,S)}(K_{AB}))$
4. $S \rightarrow B : E_{k(B,S)}(K_{AB}), M_{k'(B,S)}(A, B, N_A, N_B, E_{k(B,S)}(K_{AB}))$

- Keys $k(A, S)$ and $k'(A, S)$ are independent.
- (E(·),D(·)) is an IND-CPA symmetric-key encryption scheme.
- M(·) is a MAC unforgeable under adaptive chosen-message attacks (UF-CMA).

Formalizing an authentication protocol

Needham-Schroeder Public Key protocol (1978):

1. $A \rightarrow B : \{N_A, A\}_{pk(B)}$
2. $B \rightarrow A : \{N_A, N_B\}_{pk(A)}$
3. $A \rightarrow B : \{N_B\}_{pk(B)}$

Security goal: mutual authentication of A and B

Protocol execution

1. $A \rightarrow B : \{N_A, A\}_{pk(B)}$
2. $B \rightarrow A : \{N_A, N_B\}_{pk(A)}$
3. $A \rightarrow B : \{N_B\}_{pk(B)}$

Role A:

- Construct and send message 1:
  - Generate nonce N_A, concatenate it with identifier A, and encrypt with $pk(B)$. Send $\{N_A, A\}_{pk(B)}$ to B.
- Receive a message M and check that it is message 2.
  Q: how to do this when running multiple sessions (or protocols) in parallel?
  - Decrypt M with $sk(A)$, call it M'. If decryption fails, reject M.
    Q: how to detect wrong decryption?
    Q: what to do about rejected messages?
  - Split the message into two nonces N'_A and N_B. If not possible, reject M.
    Q: how to check this?
  - Check that $N'_A = N_A$. If not, reject M.
- Construct and send message 3.
  - Encrypt N_B with $pk(B)$
  - Send $\{N_B\}_{pk(B)}$ to B.

Informal correctness

1. $A \rightarrow B : \{N_A, A\}_{pk(B)}$

This is Alice and I have chosen a nonce N_A.

2. $B \rightarrow A : \{N_A, N_B\}_{pk(A)}$

Here is your nonce N_A. Since I could read your message, I must be Bob. I also have a challenge for you: N_B

3. $A \rightarrow B : \{N_B\}_{pk(B)}$

You sent me N_B. Since only Alice could read it and I sent it back to you, I must be Alice.

Protocol attacks

- Impersonation attack: pretend to be another principal.
- Man-in-the-middle attack: $a \leftrightarrow i \leftrightarrow b$.
- Replay attack: reuse parts of previous messages.
- Reflection attack: send transmitted information back to originator.
- Unknown key-share (UKS) attack: entities will have different views about their peers.
- Oracle attack: use protocol responses as encryption and decryption services.
- Type flaw attack: use absence of proper message type checking and substitute a different type of message field.
- Chosen protocol attack: A new protocol is made to interact with the main protocol and create a security hole.
- Internal action flaws: *Small-subgroup attack, Invalid-curve attack, etc.*
- Cryptanalysis: *Offline dictionary attack, Online dictionary attack, etc.*

Attacks on protocol implementations

- Side-channel attacks:
  - Timing attack
  - Power analysis attack
  - Electromagnetic analysis attack
  - Acoustic cryptanalysis
  - Padding oracle attacks
  - Memory attacks
  - Fault injection attacks
- Traffic analysis

Man-in-the-middle attack

Needham-Schroeder Public Key protocol (1978):

1. $A \rightarrow B : \{N_A, A\}_{pk(B)}$
2. $B \rightarrow A : \{N_A, N_B\}_{pk(A)}$
3. $A \rightarrow B : \{N_B\}_{pk(B)}$

Attack (Lowe 1996):

1. $A \rightarrow I : \{N_A, A\}_{pk(I)}$

1'. $I(A) \rightarrow B : \{N_A, A\}_{pk(B)}$
2'. $B \rightarrow I(A) : \{N_A, N_B\}_{pk(A)}$

2. $I \rightarrow A : \{N_A, N_B\}_{pk(A)}$
3. $A \rightarrow I : \{N_B\}_{pk(I)}$

3'. $I(A) \rightarrow B : \{N_B\}_{pk(B)}$

Property that B authenticates A is violated:
B believes that he is talking to A but he is talking to I.

What was the problem?

1. $A \rightarrow B : \{N_A, A\}_{pk(B)}$
2. $B \rightarrow A : \{N_A, N_B\}_{pk(A)}$
3. $A \rightarrow B : \{N_B\}_{pk(B)}$

- The problem exists in step 2: It does not say where the message comes from!
- Lowe's Fix: B should include his name: $\{N_A, N_B, B\}_{pk(A)}$.

Man-in-the-middle attack on Diffie-Hellman protocol

MITM attack on the basic Diffie-Hellman protocol:

1. $a \to i(b) : exp(g, x)$

1'. $i(a) \to b : exp(g, z)$
2'. $b \to i(a) : exp(g, y)$

2. $i(b) \to a : exp(g, z)$

- a believes to share key $exp(exp(g, z), x)$ with b.
- b believes to share key $exp(exp(g, z), y)$ with a.
- The adversary knows both keys.
- Countermeasure: authenticate half-keys (e.g. with digital signatures)

  1. $A \to B : \{exp(g, x)\}_{sk(A)}$
  2. $B \to A : \{exp(g, y)\}_{sk(B)}$

Reflection attack

Example: challenge-response authentication protocol

1. $A \rightarrow B : \{|N_A|\}_{k(A,B)}$
2. $B \rightarrow A : \{|N_A + 1|\}_{k(A,B)}$

is vulnerable to a reflection attack:

1. $a \rightarrow i(b) : \{|n_a|\}_{k(a,b)}$

1'. $i(b) \rightarrow a : \{|n_a|\}_{k(a,b)}$
2'. $a \rightarrow i(b) : \{|n_a + 1|\}_{k(a,b)}$

2. $i(b) \rightarrow a : \{|n_a + 1|\}_{k(a,b)}$

- a works on behalf of the adversary and acts as an *oracle*: a provides the correct answer to her own question.
- a believes that b is operational while b may no longer exist.
- Fix: add A's name to message 1 or use different keys for each direction $k(a, b) \neq k(b, a)$.

Parallel sessions

The previous attack requires that a executes several protocol runs in parallel.

Assumption 5

The adversary may start any number of parallel protocol runs between any principals, including different runs involving the same principals and with principals taking the same or different protocol roles.

Our formal model shall allow for an unbounded number of protocol runs of arbitrary roles and with arbitrary participating principals.

Basic properties for key establishment

- Basic goals:
  - key freshness
  - parties should accept the same key if adversary is benign;
  - adversary cannot obtain the session key (often called implicit key authentication[1]).
- Adversary capabilities:
  - controls the network;
  - reveal session keys from other (non-target) sessions;
  - corrupt parties not from other (non-target) sessions to obtain long-term keys.

[1] Explicit key authentication is provided when both *implicit key authentication* and *key confirmation* is provided. In key confirmation of A to B, B will be assured that A has possession of key K.

Advanced properties for key agreement

Some goals that concern the compromise of ephemeral or long-term keys/secrets:

- Forward secrecy: Compromise of long-term keys of both parties after session is complete should not compromise the session key.
- Key compromise impersonation (KCI) resistance: After compromise of A's long-term key, the adversary should not be able to masquerade to A as any different principal.
- Leakage of ephemeral secrets: Compromise of ephemeral keys of target session should not compromise the session key.
- Weak forward secrecy: A weaker form of forward secrecy is provided when the adversary is not allowed to be active in the target session.

Prudent Engineering of Security Protocols

Principles proposed by Abadi and Needham (1994, 1995):

- Every message should say what it means.
- Specify clear conditions for a message to be acted on.
- Mention names explicitly if they are essential to the meaning.
- Be clear as to why cryptography is being used: confidentiality, message authentication, binding of messages, ...
  For example: signing $\{X, Y\}_{sk(A)}$ versus $\{X\}_{sk(A)}$, $\{Y\}_{sk(A)}$
- Be clear on what properties you are assuming.
- Beware of clock variations (for timestamps).
- ...

Good advice but

- Is the protocol guaranteed to be secure?
- Have you considered all types of attacks?
- Is it optimal and/or minimal?

Formal modeling of security protocols

Syntax:

- Alice & Bob: protocol is a sequence of A & B events: $A \rightarrow B : M$
- Tamarin: multiset rewrite rules (more fine-grained).

Semantics:

- Semantics of rewrite rule specification is a transition system.
- Trace: records past events (send, receive, events, etc.)
- Send rule: thread passes message to adversary (i.e. network)
- Receive rule: thread obtains an adversary-derivable message

Security properties:

- Predicates on the traces for authentication, secrecy, etc.
- Equivalence properties: privacy, anonymity, etc.

Why is protocol analysis difficult?

Infinite state space:

- Messages: adversary can produce messages of arbitrary size
- Sessions: unbounded number of parallel sessions
- Nonces: unbounded number of nonces (if sessions unbounded)

Undecidability:

- Secrecy problem for security protocols is undecidable (Even & Goldreich, 1983)

  even if the number of nonces or the message size is bounded

Summary

- Security protocols are building blocks in secure communications. They provide us with some properties that other cryptographic primitives cannot offer alone: e.g. authentication, freshness, etc.
- A protocol without explicit goals and assumptions is useless.
- Even simple protocols show the difficulty of a correct design.
- A protocol without a proof of its properties has probably problems.
- Formal modeling and analysis of protocols is useful but not trivial (even with the assumption of perfect cryptography).