Modeling and verification of security protocols

Part I: Basics of cryptography and introduction to security protocols

Dresden University of Technology
Martin Pitt
`martin@piware.de`

Paper and slides available at `http://www.piware.de/docs.shtml`

Role of security protocols

- critical element of the infrastructure of a distributed system
- simple, short and easy to express
- extremely subtle and hard to evaluate
- 'three-line programs that people still manage to get wrong'

→ excellent candidates for rigorous formal analysis

Structure

Aspects of security:

security properties, attacker models, limits of cryptography and security protocols

Principles of cryptographic algorithms:

keys, symmetric and asymmetric systems, DH key exchange

Security protocols:

notation, examples, vulnerabilities and attacks

Part:

Aspects of security

Security properties

What do we want to protect?

precise notions to formally talk about cryptography and protocols

Secrecy

Strongest interpretation:

> An intruder is not able to learn *anything* about *any* communication between two participants.

can be approximated quite closely, but major overhead

→ Design decision: trade off parts of secrecy against efficiency

Authentication

Strong authentication:

> If recipient R receives a message claiming to be from sender S then S sent exactly this message to R.

Weak authentication:

> If recipient R receives a message claiming to be from sender S then *either* S sent exactly this message to R *or* R unconditionally notices that this is not the case.

$\longrightarrow$ Authentication $=$ validation of origin $+$ integrity

non-repudiation: used for digital signature systems

Availability

If a certain service is requested, it must actually be available.

vital applications: distress signals, emergency telephones, remote surgery

Cryptography and protocols can do only little to achieve this!

Solutions: redundancy, reverse logic on alarms

Intruder models

Who do we want to protect data from?

> Every kind of security needs a physical support which is ultimately trusted.

→ impossible to defend against an almighty or omnipotent attacker

Limits of cryptography and security protocols

Many secure algorithms and protocols available (proved or stood the test of time)

→ only at *mathematical* level!

Real-world implementations: refinement → new aspects, properties and side effects:

- power consumption
- execution time
- radiation
- covert channels

Part:

Principles of cryptographic algorithms

Keys and why they are needed

> In every distributed system there must be something that distinguishes the legitimate recipient from all other participants.

In cryptography: knowledge of a specific secret → key

Vital properties of key generation

- based on a truly random number
- very big key space → prevent identical keys and right guesses
- verification of relationship key ↔ owner

The whole system is at most as good and trustworthy as the initial key generation.

Symmetric cryptography

- encryption and decryption / signing and testing is done with equal keys
- several thousand years old
- examples: Vernam chiffre (one time pad), DES, AES

Symmetric concealment

$encrypt : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{C}$
$decrypt : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{X}$

$\forall k \in \mathcal{K}, x \in \mathcal{X}.\ decrypt\big(encrypt(x, k), k\big) = x$

Sending an encrypted message from A to B:

- encryption: A chooses a message $x \in \mathcal{X}$ and calculates: $c = crypt(x, k_{AB})$
- transfer: c is now sent to the recipient (and possibly to observers and attackers)
- decryption: B calculates $x = decrypt(c, k_{AB})$

Symmetric authentication

$sign : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{S}$

Sending a signed message from A to B:

- signing: A chooses a message $x \in \mathcal{X}$ and calculates $s = sign(x, k_{AB})$
- transfer: $x; s$ is now sent to the recipient (and possibly to attackers)
- receiving: B receives a message $x'; s'$ (either the original or modified by attackers)
- test: B calculates $s'' = sign(x', k_{AB})$; if $s'' = s'$, the message is valid.

Symmetric key distribution

To use algorithms, participants have to agree to a common key → easy if they can meet

if not → trusted third party; exchange must be secret and authentic

Problems:

- verification of equality
- key explosion
- dynamic set of participants

solved by Needham-Schroeder Secret Key (NSSK) protocol

Asymmetric cryptography

- different keys for encryption and decryption / signing and testing
- first paper: 1976 (Diffie and Hellmann) → key exchange
- 1978: Rivest, Shamir, Adleman: RSA algorithm
- based on one-way function
- used conjectures: factorization, discrete logarithm
- breakthrough of “crypto for the masses” → PGP, GPG

Asymmetric concealment

$$encrypt : \mathcal{X} \times \mathcal{PUB} \to \mathcal{C}$$
$$decrypt : \mathcal{C} \times \mathcal{SEC} \to \mathcal{X}$$

$$\forall x \in \mathcal{X}.\ decrypt\big(encrypt(x, pub_A), sec_A\big) = x$$

Sending an encrypted message from A to B:

- encryption: A chooses a message $x \in \mathcal{X}$ and calculates $c = encrypt(x, pub_B)$
- transfer: c is now sent to the recipient (and possibly to observers and attackers)
- decryption: B calculates $x = decrypt(c, sec_B)$

Asymmetric authentication

$sign : \mathcal{X} \times \mathcal{SEC} \to \mathcal{S}$
$test : \mathcal{X} \times \mathcal{S} \times \mathcal{PUB} \to \{\text{correct}, \text{wrong}\}$

Creating a signed message by A:

- signing: A chooses a message $x \in \mathcal{X}$ and calculates $s = sign(x, sec_A)$
- transfer: $x; s$ is now sent to all desired recipients (and possibly to attackers)
- receiving: a participant B receives a message $x'; s'$ (either the original or modified by attackers)
- test: B now checks if $test(x', s', pub_A) = \text{correct}$

→ provides non-repudiation → digital signature system

Part:

Security protocols

Security protocols

Protocol: a prescribed sequence of interactions between entities designed to achieve a certain goal and end.

Security protocols: provide security properties to distributed systems

Notation

Message n $a \rightarrow b : data$

$data$ consists of:

atoms: names, variables, literal constants.

nonces: n_A unpredictable, freshly generated unique number

encryption: $\{data\}_k$: encryption of $data$ with the key k.

authentication: $Sign_k(data)$: signature of $data$ using the key k.

concatenation: $a.b$

Challenge – Response

Purpose: verify that two parties A and B share a common secret key k without revealing it.

1. A→ B: n_A
2. B→ A: $\{n_A\}_k.n_B$
3. A→ B: $\{n_B\}_k$

Needham–Schroeder Secret Key

Purpose: establish a common secret key between A and B using only symmetric cryptography and a trusted third party S (server)

Preliminary: pairwise distinct keys with S

1. A→ S: $A.B.n_A$
2. S→ A: $\{n_A.B.k_{AB}.\{k_{AB}.A\}_{SB}\}_{SA}$
3. A→ B: $\{k_{AB}.A\}_{SB}$
4. B→ A: $\{n_B\}_{k_{AB}}$
5. A→ B: $\{n_B - 1\}_{k_{AB}}$

solves key explosion, dynamic participant set

NB: encryption must provide binding of concatenated parts!

Station–To–Station protocol

Purpose: establish a common secret key between A and B without trusted third party → uses DH key exchange

1. A→ B: a^x
2. B→ A: $a^y.\{Sign_B(a^y.a^x)\}_k$
3. A→ B: $\{Sign_A(a^x.a^y)\}_k$

Replay attack

Attacker monitors a (possibly partial) run of a protocol and later replays some messages. This can happen if the protocol does not have any mechanism for distinguishing between separate runs or cannot determine the freshness of messages.

Example: military ship that gets encrypted commands from base

Solutions: nonces, run identifiers, timestamps, indeterministic encryption

Mirror attack

Other participant is made to answer his own questions.

Vulnerability on challenge – response (A does not know k):

1. $A \rightarrow S$: n_A
2. $S \rightarrow A$: $\{n_A\}_k.n_S$
3. $A' \rightarrow S$: n_S
4. $S \rightarrow A'$: $\{n_S\}_k.n'_S$
5. $A \rightarrow S$: $\{n_S\}_k$

Man in the middle

The attacker imposes himself between the communications of A and B. This can happen if messages or keys are not properly authenticated.

“Academic” (stupid) example protocol for encrypted communication without knowing each other’s public key:

Use of a commutative asymmetric cipher (like RSA):

1. $A \rightarrow B$: $\{X\}_{p_A}$
2. $B \rightarrow A$: $\{\{X\}_{p_A}\}_{p_B}$ $\quad \{\{X\}_{p_A}\}_{p_B} = \{\{X\}_{p_B}\}_{p_A}$
3. $A \rightarrow B$: $\{X\}_{p_B}$

Man in the middle - attack

1. $A \rightarrow I(B)$: $\{X\}_{p_A}$
2. $I(B) \rightarrow A$: $\{\{X\}_{p_A}\}_{p_I}$
3. $A \rightarrow I(B)$: $\{X\}_{p_I}$
4. $I(A) \rightarrow B$: $\{X\}_{p_I}$
5. $B \rightarrow I(A)$: $\{\{X\}_{p_I}\}_{p_B}$
6. $I(A) \rightarrow B$: $\{X\}_{p_B}$

Practical applications: initial key exchange is most susceptible to this attack

→ key exchange plays the role of the physical support!

Interleave

The attacker uses several parallel runs of a protocol to exploit their interactions.

Needham–Schroeder Public Key:

1. $A \to B$: $\{A.n_A\}_{p_B}$
2. $B \to A$: $\{n_A.n_B\}_{p_A}$
3. $A \to B$: $\{n_B\}_{p_B}$

has been believed secure for many years; was even analyzed with BAN logic!

Interleave – attack

I is legitimate user, plays an active role, but does not obey to protocol:

a.1. A→ I: $\{A.n_A\}_{p_I}$
b.1. I(A)→ B: $\{A.n_A\}_{p_B}$
b.2. B→ I(A): $\{n_A.n_B\}_{p_A}$
a.2. I→ A: $\{n_A.n_B\}_{p_A}$
a.3. A→ I: $\{n_B\}_{p_I}$
b.3. I(A)→ B: $\{n_B\}_{p_B}$

→ I knows both nonces and caused mismatch in A's and B's perception:

A thinks: communication and secret share with I
B thinks: communication and secret share with A

Part:

Questions and criticism