More about message authentication and signature schemes

The RSA digital signature scheme

– Use the public key (n, e) and private key d as for RSA encryption and decryption.

– Signature of message m is $m^d \bmod n$.

– To verify that s is signature of m, check that $s^e = m \pmod{n}$.
Verification succeeds iff s is signature of m.

More about message authentication and signature schemes

The RSA digital signature scheme

– Use the public key (n, e) and private key d as for RSA encryption and decryption.

– Signature of message m is $m^d \bmod n$.

– To verify that s is signature of m, check that $s^e = m \pmod{n}$.

Verification succeeds iff s is signature of m.

An attack: Suppose an attacker knows

$$m_1 \text{ and } s_1 = m_1^d \bmod n$$
$$m_1 \text{ and } s_1 = m_1^d \bmod n$$

He can then forge the signature of $m_1 m_2$:

$$(m_1 m_2)^d \bmod n = (m_1^d \bmod n)(m_2^d \bmod n) \bmod n$$

An example message authentication scheme

Encrypt the message using CBC (Cipher Block Chaining) with zero IV.
The tag is the last block of ciphertext.

Given key K and message containing n blocks: $m = B_1 \ldots B_n$

$C_0 = 0 \ldots 0$
$C_i = \{C_{i-1} \oplus B_i\}_K \quad (1 \leq i \leq n)$
Tagging: $\mathcal{T}_K(m) = C_n$

An example message authentication scheme

Encrypt the message using CBC (Cipher Block Chaining) with zero IV.

The tag is the last block of ciphertext.

Given key K and message containing n blocks: $m = B_1 \ldots B_n$

$C_0 = 0 \ldots 0$

$C_i = \{C_{i-1} \oplus B_i\}_K \quad (1 \leq i \leq n)$

Tagging: $\mathcal{T}_K(m) = C_n$

Verification $\mathcal{V}_K(m, t)$: just check whether $t = \mathcal{T}_K(m)$!

Verification is trivial if the tagging algorithm is deterministic.

(Note that the algorithm for encrypting a block is deterministic.)

An example message authentication scheme

Encrypt the message using CBC (Cipher Block Chaining) with zero IV.
The tag is the last block of ciphertext.

Given key K and message containing n blocks: $m = B_1 \ldots B_n$

$C_0 = 0 \ldots 0$
$C_i = \{C_{i-1} \oplus B_i\}_K \quad (1 \leq i \leq n)$
Tagging: $\mathcal{T}_K(m) = C_n$

Verification $\mathcal{V}_K(m, t)$: just check whether $t = \mathcal{T}_K(m)$!

Verification is trivial if the tagging algorithm is deterministic.
(Note that the algorithm for encrypting a block is deterministic.)

Security ??

Modeling in the Dolev-Yao style

If the intruder knows K and m then he knows $\mathcal{T}_K(m)$.

If the intruder knows K and m then he knows $\mathcal{S}(K, m)$.

For describing the AKEP1 protocol, choose keys $shre(x, y)$ and $shrm(x, y)$ to be used in sessions started by x involving y.

We have rule:

$$\begin{aligned}Agent(x), Da(y) \rightsquigarrow\ & I(shre(x, y)), I(shrm(x, y)),\\ & I(shre(y, x)), I(shrm(y, x)), Agent(x), Da(y)\end{aligned}$$

$$\begin{aligned}
A \rightarrow B: &\quad A, N_a \\
B \rightarrow A: &\quad N_b, \{k\}_{K^e_{ab}}, \mathcal{T}_{K^m_{ab}}(\langle B, A, N_a, N_b, \{k\}_{K^e_{ab}}\rangle) \\
A \rightarrow B: &\quad \mathcal{T}_{K^m_{ab}}(\langle A, N_b\rangle)
\end{aligned}$$

The rules describing the protocol are as usual. Some example properties for reachable protocol states S containing $Ha(x), Ha(y)$:

– S does not contain $shre(x, y)$ or $shrm(x, y)$.

– If S contains $B_1(x, y, u, v, w), I(m)$ and w occurs in m, then it occurs as $\{w\}_{shre(x,y)}$.

– If S contains $\{w\}_{shre(x,y)}$ then S contains $B_1(x, y, u, v, w)$ or $B_2(x, y, u, v, w)$.

. . .

Forward secrecy

A and B create different session keys k_1, k_2, ... in different sessions.

Consider that at some point of time the long-lived keys of A gets exposed, i.e. the attacker finds out the keys $K^{e^{-1}}, K^{d^{-1}}$.

Then all future session keys of A are compromised. To prevent this A can revoke the public keys K_a^e, K_a^d as soon as possible.

But what about past session keys?

Forward secrecy: past sessions should not be compromised
even if long-lived keys are exposed.

In the above protocol, the attacker records $\{k\}_{K_a^e}$ for each session key k, so he obtains k.
Hence he obtains all messages encrypted with session keys in past sessions.

Forward secrecy using Diffie-Hellman key exchange

$$\begin{aligned} A \to B: &\quad g^{N_a} \\ B \to A: &\quad g^{N_b}, \mathcal{S}(K_b^{d^{-1}}, \langle B, A, g^{N_a}, g^{N_b}\rangle) \\ A \to B: &\quad \mathcal{S}(K_a^{d^{-1}}, \langle A, g^{N_b}\rangle) \end{aligned}$$

This protocol is meant to work in the presence of an active attacker.

If the long-lived keys are exposed after the session,
the session key $g^{N_a N_b}$ is still not exposed.

Protocol analysis:
some complexity results

We consider protocols described using multiset rewriting rules, and secrecy properties expressed as non-reachability of bad protocol states, as seen in our examples.

We consider unbounded number of sessions.

We consider protocols described using multiset rewriting rules, and secrecy properties expressed as non-reachability of bad protocol states, as seen in our examples.

We consider unbounded number of sessions.

- The secrecy problem is undecidable

We consider protocols described using multiset rewriting rules, and secrecy properties expressed as non-reachability of bad protocol states, as seen in our examples.

We consider unbounded number of sessions.

- The secrecy problem is undecidable
- The problem remains undecidable if protocols don't use nonces

We consider protocols described using multiset rewriting rules, and secrecy properties expressed as non-reachability of bad protocol states, as seen in our examples.

We consider unbounded number of sessions.

- The secrecy problem is undecidable
- The problem remains undecidable if protocols don't use nonces
- In case of arbitrarily many nonces, the problem remains undecidable even if the depth and width of terms (messages) involved in the protocols are bounded by constants.

A positive result for unbounded number of sessions, no nonces, bounded message widths and depths.

In this case we can decide secrecy in EXPTIME.

- There is an exponential bound on the number of distinct facts that can be derived from the rules during protocol execution.
- Each role can be rerun as many times as we want.
- If we can reach a state S containing fact $P(\overrightarrow{t})$ then we can reach another state $S' \supseteq S$ which contains at least one more copy of $P(\overrightarrow{t})$.
- Any rule from any role can be applied any number of times.
- All facts can be treated as persistent facts.

Decision procedure:

1. R := the set of ground instances of rules from protocol theory and intruder theory.
2. $F := \emptyset$
3. repeat:
   (a) Select a rule $l \longrightarrow r$ from R.
   (b) If $l \subset F$ then $F := F \cup r$.
4. until no new facts can be added.

Check if some ground instance of the security property is included in F.

The algorithm terminates in exponential time.
(Actually we cannot do better :-))

If further, we only allow bounded number of sessions, then the non-secrecy problem is in NP.

(This is again optimal, i.e. the problem is NP-complete :-))

We may consider protocols with nonces, but because of finitely many sessions they can just be considered as finitely many constants.

An NP decision procedure for bounded number of sessions

We take as input a candidate attack and check that this actually represents an attack.

– We show that the length of candidate attacks is of polynomial length.

– We present a polynomial time decision procedure for checking that a given candidate attack is actually an attack.

Showing that candidate attack runs are of polynomial length:

Let r be the bound on the number of sessions.

Let R be the maximum number of steps in a session.

Hence any attack consists of at most rR protocol steps.

Let k be the bound on message size.

The maximum number of distinct nonces generated by the intruder is $B = krR$.

The maximum number of distinct nonces used by the participants is also B.

These can be assumed to be generated beforehand.

Guessing a polynomial length candidate attack:

1. Guess a set of at most krR nonces to be used by the intruder in the attack. These are included in the intruder's initial knowledge.

2. Generate upto r roles using the rules for role generation. These initial role states also contain all the nonces that are going to be used by that role.

3. Guess a candidate sequence of upto rR protocol steps for the attack. The variables occurring in these rules are fully instantiated by actual messages, nonces, keys. Let the sequence of steps be $s_1, \ldots, s_N$.

Checking that the candidate attack sequence is a valid protocol run

Let S_i be the multiset of facts after step s_i. S_0 contains the initial knowledge of the intruder, initial role states, etc. For each rule

$$s_i : A_j(\overrightarrow{m}), I(n) \rightarrow A_k(\overrightarrow{m'}), I(n'), I(n)$$

1. Check that $A_i(\overrightarrow{m}) \in S_{i-1}$. If not, REJECT.
2. Check that the message n can be can be deduced by the intruder from the messages in his memory in S_{i-1}. If not, REJECT.
3. The new state S_i has all the facts of S_{i-1} except a fact $A_j(\overrightarrow{m})$, as well as new facts $A_k(\overrightarrow{m'})$ and $I(n')$.

The above procedure is repeated for each of the steps $s_1, \ldots, s_N$.

Guess a suitable instance of the security property. If it is included in s_N then ACCEPT otherwise REJECT.

Protocols as processes: spi calculus [AbadiGordon97]

Extends pi calculus [MilnerParrowWalker92] by adding cryptographic operations.

Pi calculus programs consist of independent parallel processes that synchronize by passing messages on named channels.

Channels may be restricted, so that only certain processes can communicate on them. Scope of channels may change during execution: channels can be sent as messages.

Security is expressed as equivalence between processes in the eyes of an arbitrary environment. Environment is modeled as an arbitrary spi calculus process.

We write $\overline{c}\langle M\rangle.P$ to denote a process that sends the message M on channel c after which it executes the process P.

$c(x).Q$ denotes a process that is listening on the channel c. If it receives some message M on this channel then it will execute the process $Q[M/x]$.

We may compose these two processes in parallel to get a bigger process, denoted as $\overline{c}\langle M\rangle.P \mid c(x).Q$. Now the two smaller processes may communicate on the channel c after which they will execute the process $P \mid Q[M/x]$.

We use $\mathbf{0}$ to denote the nil process which does nothing.

A one message protocol:

$$A \longrightarrow B : M \qquad \text{on } c_{AB}$$

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}}\langle M\rangle.\mathbf{0} \\ B &\triangleq c_{AB}(x).\mathbf{0} \\ Inst(M) &\triangleq (\nu c_{AB})(A(M) \mid B) \end{aligned}$$

$(\nu c)P$ denotes a process that creates a new channel c which is then used by the process P. This channel is not accessible outside the process P.

$Inst(M)$ is an abstraction. For any M it denotes an exchange of message M between A and B.

For example $Inst(0)$ denotes exchange of the data '0' between A and B.

$$\begin{aligned} A(0) &= \overline{c_{AB}}\langle 0\rangle.\mathbf{0} \\ B &= c_{AB}(x).\mathbf{0} \\ Inst(0) &= (\nu c_{AB})(A(0) \mid B) \end{aligned}$$

Here B runs the process F on the message M that it gets.

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}}\langle M\rangle.\mathbf{0} \\ B &\triangleq c_{AB}(x).F(x) \\ Inst(M) &\triangleq (\nu c_{AB})(A(M) \mid B) \end{aligned}$$

Authenticity : B always applies F to the message M that A sends.

Secrecy : If F does not reveal M then the whole process does not reveal M. Stated in terms of equivalence:

If $F(M) \simeq F(M')$ for all M, M' then $Inst(M) \simeq Inst(M')$ for all M, M'.

For authenticity, we take the following protocol as specification:

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}}\langle M\rangle.\mathbf{0} \\ B_{spec}(M) &\triangleq c_{AB}(x).F(M) \\ Inst_{spec}(M) &\triangleq (\nu c_{AB})(A(M) \mid B_{spec}(M)) \end{aligned}$$

Authenticity: $Inst(M) \simeq Inst_{spec}(M)$ for all M.

Terms

We assume an infinite set $m, n, p, q, r, \ldots$ of names and an infinite set $x, y, z, \ldots$ of variables.

Terms $L, M, N ::=$

n	name
(M, N)	pair
0	zero
$suc(M)$	successor
x	variable

Pi calculus processes

P,Q,R ::=

$\overline{M}\langle N\rangle.P$	output
$M(x).P$	input
$P \mid Q$	parallel composition
$(\nu n)P$	restriction
$!P$	replication
$[M\ is\ N]P$	match
$\mathbf{0}$	nil
$let\ (x,y) = M\ in\ P$	pair splitting
$case\ M\ of\ 0 : P\ suc(x) : Q$	integer case

Channel establishment:

$$
\begin{aligned}
A &\longrightarrow S : c_{AB} \text{ on } c_{AS} \\
S &\longrightarrow B : c_{AB} \text{ on } c_{SB} \\
A &\longrightarrow B : M \text{ on } c_{AB}
\end{aligned}
$$

$$
\begin{aligned}
A(M) &\triangleq (\nu c_{AB})\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle.\mathbf{0} \\
S &\triangleq c_{AS}(x).\overline{c_{SB}}\langle x\rangle.\mathbf{0} \\
B &\triangleq c_{SB}(x).x(y).F(y) \\
Inst(M) &\triangleq (\nu c_{AS})(\nu c_{SB})(A(M) \mid S \mid B)
\end{aligned}
$$

The scope of the channel c_{AB} extrudes out of A to B.

$$
\begin{aligned}
A(M) &\triangleq (\nu c_{AB})\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle.\mathbf{0} \\
S &\triangleq c_{AS}(x).\overline{c_{SB}}\langle x\rangle.\mathbf{0} \\
B_{spec}(M) &\triangleq c_{SB}(x).x(y).F(M) \\
Inst(M) &\triangleq (\nu c_{AS})(\nu c_{SB})(A(M) \mid S \mid B_{spec}(M))
\end{aligned}
$$

Authenticity: $Inst(M) \simeq Inst_{spec}(M)$, for all M

Secrecy: $Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for all M, M'.

Adding cryptography: the spi calculus

Terms $L, M, N ::=$

n	name
(M, N)	pair
0	zero
$suc(M)$	successor
x	variable
$\{M\}_N$	encryption

Processes P,Q,R ::=

$\overline{M}\langle N\rangle.P$	output
$M(x).P$	input
$P \mid Q$	parallel composition
$(\nu n)P$	restriction
$!P$	replication
$[M\ is\ N]P$	match
$\mathbf{0}$	nil
$let\ (x,y) = M\ in\ P$	pair splitting
$case\ M\ of\ 0 : P\ suc(x) : Q$	integer case
$case\ L\ of\ \{x\}_N\ in\ P$	decryption

A one message protocol using cryptography:

$$A \longrightarrow B : \{M\}_{K_{AB}} \text{ on } c_{AB}$$

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}}\langle \{M\}_{K_{AB}} \rangle.\mathbf{0} \\ B &\triangleq c_{AB}(x).case\ x\ of\ \{y\}_{K_{AB}}\ in\ F(y) \\ Inst(M) &\triangleq (\nu K_{AB})(A(M) \mid B) \end{aligned}$$

The key K_{AB} is restricted, only A and B can use it. On the other hand the channel c_{AB} is public. Other principals may send messages on it or listen on it.

The specification protocol for authentication:

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}}\langle\{M\}_{K_{AB}}\rangle.\mathbf{0} \\ B_{spec}(M) &\triangleq c_{AB}(x).case\ x\ of\ \{y\}_{K_{AB}}\ in\ F(M) \\ Inst_{spec}(M) &\triangleq (\nu K_{AB})(A(M) \mid B_{spec}(M)) \end{aligned}$$

Authenticity: $Inst(M) \simeq Inst_{spec}(M)$ for all M.

Secrecy : $Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for all M, M'.

The notion of process equivalence needs to be coarse-grained enough.

$$P(M) \triangleq (\nu K)\overline{c}\langle\{M\}_K\rangle.\mathbf{0}$$

Clearly $P(M)$ and $P(M')$ are different for different M and M'. Usual equivalence relations would distinguish them.

However we would like to consider $P(M)$ and $P(M')$ as equivalent processes because no outside observer can distinguish them because of the encryption using the restricted key K.

Key establishment:

$$A \longrightarrow S : \{K_{AB}\}_{K_{AS}} \text{ on } c_{AS}$$
$$S \longrightarrow B : \{K_{AB}\}_{K_{SB}} \text{ on } c_{SB}$$
$$A \longrightarrow B : \{M\}_{K_{AB}} \text{ on } c_{AB}$$

$$\begin{array}{rcl}
A(M) & \triangleq & (\nu K_{AB})(\overline{c_{AS}}\langle\{K_{AB}\}_{K_{AS}}\rangle \\
 & & .\overline{c_{AB}}\langle\{M\}_{K_{AB}}\rangle.\mathbf{0} \\
S & \triangleq & c_{AS}(x).case\ x\ of\ \{y\}_{K_{AS}}\ in\ \overline{c_{SB}}\langle\{y\}_{K_{SB}}\rangle.\mathbf{0} \\
B & \triangleq & c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in \\
 & & c_{AB}(z).case\ z\ of\ \{w\}_{y}\ in\ F(w) \\
Inst(M) & \triangleq & (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B)
\end{array}$$

$$
\begin{array}{rcl}
A(M) & \triangleq & (\nu K_{AB})(\overline{c_{AS}}\langle\{K_{AB}\}_{K_{AS}}\rangle. \\
 & & \overline{c_{AB}}\langle\{M\}_{K_{AB}}\rangle.\mathbf{0} \\
S & \triangleq & c_{AS}(x).case\ x\ of\ \{y\}_{K_{AS}}\ in \\
 & & \overline{c_{SB}}\langle\{y\}_{K_{SB}}\rangle.\mathbf{0} \\
B_{spec}(M) & \triangleq & c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}} in \\
 & & c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(M) \\
Inst_{spec}(M) & \triangleq & (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B_{spec}(M))
\end{array}
$$

Authenticity: $Inst(M) \simeq Inst_{spec}(M)$, for all M

Secrecy: $Inst(M) \simeq Inst(M')$ if $F(M) \simeq F(M')$, for all M, M'.