Summary of paper : A Calculus for Cryptographic Protocols - The Spi Calculus

Chen Peng
Yoshikawa & Ma Laboratory
Student ID: 6930-30-2948
Email: chenpeng.acmer@yahoo.com

Jan.31 2019

1 The π-calculus and the spi calculus

The spi calculus was designed by Martin Abadi and Andrew D. Gordon. It is an extension of the π-calculus with cryptographic primitives for **formalized reasoning about cryptographic systems.**
The π-calculus a process calculus, which is simple, and has very few terms and so is a very small language, yet is very expressive. The scoping rules in the π-calculus becomes the basis of security, because they guarantee that the environment of a protocol (maybe the attacker) cannot access a channel that it is not explicitly given.
In general, the π-calculus appears to be a convenient calculus of protocols for secure communication. But a obvious shortcoming of the π-calculus is it cannot express the cryptographic operations that are commonly used for implementing channels in distributed systems, because there are no constructs for encryption and decryption.
Therefore, the spi calculus was created in order to permit an explicit representation of the use of cryptography in protocols. The difference between the spi calculus and other notations for describing security protocols is that the spi calculus provides a precise and solid basis for reasoning about protocols. Besides, the spi calculus is directly executable and it has a precise semanticcs. Specifically, security guarantees can be expressed as equivalences between spi calculus processes. Here, equivalence means equivalence under the circumstance of any arbitrary environment, and equivalence is not too hard to prove. In the spi calculus, the environment is supposed to be an arbitrary spi calculus process.
A possible improvement for spi calculus would be to introduce probabilistic interpretation and computational complexity into the spi calculus, so that it is capable of including notions of probability and complexity.

2 Authenticity and secrecy in π-calculus

2.1 Basic semantics

There are:

- An infinite set of *names*, to be used as channels names for communication. (e.g m,n,p,q,r)
- An infinite set of *variables*. (e.g. x,y,z)
- The set of *terms*:

$L, M, N ::=$	terms
n	name
(M, N)	pair
0	zero
$\mathrm{suc}(M)$	successor
x	variable

- The set *processes*:

$P, Q, R ::=$	processes
$\overline{M} < N > .P$	output
$M(x).P$	input
$P \vert Q$	parallel composition
$(vn)P$	restriction
$!P$	replication
$[MisN]P$	match
0	nil
$let(x, y) = MinP$	pair splitting
$caseMof0 : Psuc(x) : Q$	integer case

- Notice1:
  A restriction ($vn)P$ is a process that makes a new, private name n, and then behaves as P .
- Notice2:
  A pair splitting process let $(x, y) = M$ in P behaves as $P[N/x][L/y]$ if term M is the pair (N, L). Otherwise, the process is stuck.
- Notice3:
  An integer case process case M of $0 : P\,\mathrm{suc}(x) : Q$ behaves as P if term M is 0, as $Q[N/x]$ if M is $\mathrm{suc}(N)$. Otherwise, the process is stuck.
- Notice4:
  $P \simeq Q$ means that the behaviours of the processes P and Q are indistinguishable. In other words, the processes P and Q mave different internal

2.2 Abstract security protocols in π-calculus

Quick example of using restricted channels:
There are two principals A and B that share a channel, c_{AB}; only A and B can

send data or listen on this channel. The protocol is simply that A uses c_{AB} for sending a single message M to B.

Innformally:

$$Message1 \quad A \to B : M \qquad \text{on } c_{AB}$$

In π-calculus:

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}}\langle M\rangle \\ B &\triangleq c_{AB}(x).0 \\ \mathrm{Inst}(M) &\triangleq (\upsilon c_{AB})\,(A(M)|B) \end{aligned}$$

Informally, $F(x)$ is simply the result of applying F to x . More formally, F is an abstraction, and $F(x)$ is an instantiation of the abstraction.

There are two properties in this protocol: authenticity and secrecy.
1. **Authenticity**: B always applies F to the message M that A sends; an attacker cannot cause B to apply F to some other message.
2. **Secrecy**: The message M cannot be read in transit from A to B: if F does not reveal M, then the whole protocol does not reveal M.

Formally:
Authenticity:

$$\bullet\, \mathrm{Inst}(M) \simeq \mathrm{Inst}_{spec}(M), \text{ for all } M$$

Secrecy:

$$\bullet\, \mathrm{Inst}(M) \simeq \mathrm{Inst}\,(M') \text{ if } F(M) \simeq F\,(M')\,, \text{ for all } M \text{ and } M'$$

where $B_{spec}(M)$ is a varient of B. This variant receives an input from A and then acts like B when B receives M. We may say that $B_{spec}(M)$ is a "*magical*" version of B that knows the message M sent by A, and similarly $Inst_{spec}$ is a "*magical*" version of Inst.

3 Authenticity and secrecy in spi-calculus

3.1 The Spi Calculus with Shared-Key Cryptography

The set of terms in spi calculus:

$$\begin{aligned} L, M, N ::= &\ \text{terms} \\ \cdots &\ \text{the same as } \pi\text{-calculus} \\ \{M\}_N &\ \text{shared-key encryption} \end{aligned}$$

The set of processes in spi calculus:

$$\begin{array}{cl} P, Q ::= & \text{processes} \\ \cdots & \text{the same as } \pi\text{-calculus} \\ \text{case } L \text{ of } \{x\}_N \text{ in } P & \text{shared-key decryption} \end{array}$$

where The term $\{M\}_N$ represents the ciphertext obtained by encrypting the term M under the key N using a shared-key cryptosystem such as DES [DES77]. Besides, the process case L of $\{x\}_N$ in P attempts to decrypt the term L with the key N. If L is a ciphertext of the form $\{M\}_N$, then the process behaves as P[M/x]. Otherwise, the process is stuck.

3.2 A cryptographic example

There are two principals A and B that share a key K_{AB}; in addition, we assume there is a public channel c_{AB} that A and B can use for communication, but which is in no way secure. The protocol is simply that A sends a message M under K_{AB} to B, on c_{AB}.

Innformally:

$$Message1 \quad A \rightarrow B : \{M\}_{K_{AB}} \quad \text{ on } c_{AB}$$

In spi-calculus:

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}} \langle \{M\}_{K_{AB}} \rangle \\ B &\triangleq c_{AB}(x) \cdot \text{case x of} \{y\}_{K_{AB}} \text{ in } F(y) \\ \text{In}\, st(M) &\triangleq (vK_{AB})\, (A(M)|B) \end{aligned}$$

Use specifications:

$$\begin{aligned} A(M) &\triangleq \overline{c_{AB}} \langle \{M\}_{K_{AB}} \rangle \\ B_{spec} &\triangleq c_{AB}(x) \cdot \text{case x of} \{y\}_{K_{AB}} \text{ in } F(M) \\ \text{Inst}_{spec}(M) &\triangleq (vK_{AB})\, (A(M)|B_{spec}(M)) \end{aligned}$$

1. **Authenticity**:

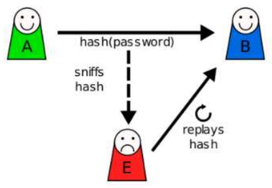
$$\text{Inst}(M) \simeq \text{Inst}_{spec}(M), \text{ for all } M$$

2. **Secrecy**:

$$\text{Inst}(M) \simeq \text{Inst}\,(M') \text{ if } F(M) \simeq F\,(M')\,, \text{ for all } M \text{ and } M'$$

3.3 The solution to reply attacks in spi calculus

The definition of Replay Attacks from Wikipedia:
A replay attack (also known as playback attack) is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed.
Suppose Alice wants to prove her identity to Bob. Bob requests her password as proof of identity, which Alice dutifully provides (possibly after some transformation like a hash function); meanwhile, Eve is eavesdropping on the conversation and keeps the password (or the hash). After the interchange is over, Eve (posing as Alice) connects to Bob; **when asked for a proof of identity, Eve sends Alice's password (or hash) read from the last session which Bob accepts, thus granting Eve access.**



For example, the Wide Mouthed Frog protocol can be expressed as follows:

$$\begin{array}{lll} \text{Message 1} & A \to S : \{K_{AB}\}_{K_{AS}} & \text{on } c_{AS} \\ \text{Message 2} & S \to B : \{K_{AB}\}_{K_{SB}} & \text{on } c_{SB} \\ \text{Message 3} & A \to B : \{M\}_{K_{AB}} & \text{on } c_{AB} \end{array}$$

where the principals A and B share keys K_{AS} and K_{SB} respectively with a server S.

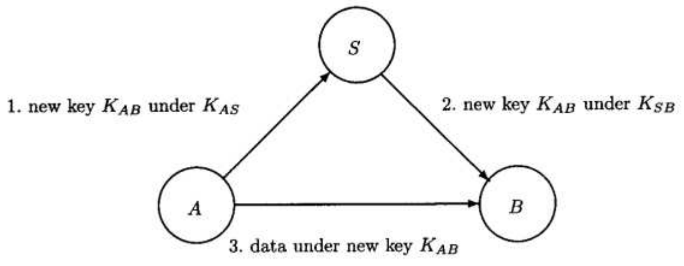

S
1. new key K_{AB} under K_{AS}
2. new key K_{AB} under K_{SB}
A
B
3. data under new key K_{AB}

FIG. 2. Sketch of the Wide Mouthed Frog protocol.

After extending this standard example to a system of $n + 1$ principals:

$$\begin{array}{lll} \text{Message 1} & A \to S : A, \{B, K_{AB}\}_{K_{AS}} & \text{on } c_S \\ \text{Message 2} & S \to B : \{A, K_{AB}\}_{K_{SB}} & \text{on } c_B \\ \text{Message 3} & A \to B : A, \{M\}_{K_{AB}} & \text{on } c_B \end{array}$$

This protocol is vulnerable to a replay attack that invalidates the authenticity equation. Consider the system $Sys(I, I')$ where $I = (i, j, M)$ and $I = (i, j, M')$. An attacker can replay messages of one instance and get them mistaken for messages of the other instance, causing M to be passed twice to F.

Furthermore, we can improve the original protocol (multiple principals) by adding **nonce handshakes** as protection against replay attacks.

Message 1	$A \rightarrow S : A$	on c_S
Message 2	$S \rightarrow A : N_S$	on c_A
Message 3	$A \rightarrow S : A, \{A, A, B, K_{AB}, N_S\}_{K_{AS}}$	on c_S
Message 4	$S \rightarrow B : *$	on c_B
Message 5	$B \rightarrow S : N_B$	on c_S
Message 6	$S \rightarrow B : \{S, A, B, K_{AB}, N_B\}_{K_{SB}}$	on c_B
Message 7	$A \rightarrow B : A, \{M\}_{K_{AB}}$	on c_B

The nonce must not have been used before for this purpose. Obviously the nonce is not secret, but it must be unpredictable. We represent the nonces of this protocol as newly created names.
In this protocol, the authenticity property can be obtained:

$$Sys\,(I_1, \ldots, I_m) \simeq Sys_{spec}\,(I_1, \ldots, I_m)$$
$$\text{for any instances } I_1, \ldots, I_m$$

By introducing nonces, the attack can no longer distinguish Sys $(I_1, \ldots, I_m)$ from $Sys_{spec}\,(I_1, \ldots, I_m)$.

Besides, we have the secrecy property:

$$\text{Sys}\,(I_1, \ldots, I_m) \simeq \text{Sys}\,(J_1, \ldots, J_m)$$
$$\text{if each pair } \left(I_1, J_1\right), \ldots, \left(I_m, J_m\right) \text{ is indistinguishable.}$$