Spi Calculus

Gokhan Gokoz
Chad R. Meiners

What Spi Calculus Is

- Spi calculus is a form of pi calculus extended to support cryptography.
- Pi calculus is a language for describing and implementing concurrent processes over communication channels.
- Pi calculus is designed to have a concise description when compared to CSP.
- Spi calculus adds operators to perform symmetric cryptography.

How Pi and Spi Calculus is used

to verify security properties of protocols.
- Authenticity
  - Is the implementation equivalent to the specification?
- Secrecy
  - Can an external process distinguish one instance from another?
Basic Facilities of Pi Calculus

- **Process**: A system is constructed out of a set of concurrent processes.
- **Scope**: Variables and channels may be restricted to certain processes or they may be global.
- **Channel**: Processes communicate and synchronize with each other via channels.

Scope Extrusion

- Channels may be placed as messages on channels.
- Allows for scope restricted channel to be used outside of its original scope.
- Allows dataflow analysis.
- Spi calculus adds encryption operators.

Pi Grammar

Pi calculus has four types of objects:
- **Names**: channels.
  - represented as \( m, n, p, q \), and \( r \).
- **Variables**: 
  - represented as \( x, y, \) and \( z \).
- **Terms**: objects in Pi calculus.
  - represented as \( L, M, N \).
- **Processes**: 
  - represented as \( P, Q, \) and \( R \).
Terms

A term can be one of the following five forms:

• $n$: the name of a channel
• $(M, N)$: a pair of terms.
• $0$: the number zero.
• $\text{su}(M)$: the successor of $M$.
• $x$: a variable.

Process Primitives

• $0$: Is the nil process.
• $P | Q$: Is the process composition operator.
• $!P$: Is the process replication operator

Examples:

$A := 0$ : $A$ is the nil process
$B := !A | C$ : $B$ is an infinite number of $A$ in parallel with $C$.

Process communication

• $M < N > P$
  – communicate message $N$ on channel $M$
  – becomes $P$.
• $M(x).P$
  – block until it receives a message $N$ from channel $M$
  – $P$ where all occurrences of $x$ in $P$ are replaced by $N$
  – (We abbreviate such replacements with $P[N/x]$)

Examples:

$A := c<\emptyset >.0$ : $A$ sends nil on $c$ and becomes nil.
$B := c(x).0$ : $B$ received $x$ on $c$ and becomes nil.
Process Decisions

- \( [M \text{ is } N]P \)
  - \( P \) if \( M=N \)
  - else 0

- let \((x,y) = M \text{ in } P\)
  - \( P[N;x][Ly] \) when \( M=(N,L) \)
  - otherwise 0

- case \( M \) is 0 : \( P \) suc(\( x \)) : \( Q \)
  - \( P \) when \( M=0 \)
  - \( Q[N;x] \) if \( M=\text{suc}(N) \)
  - 0 if \( M \) is not an integer.

Process Decisions

Examples:
- \( A(M,N) := [M,N]B \)
  - \( A \) is \( B \) if \( M=N \); otherwise, \( A \) is 0.

- \( B(M) := \text{let } (x,y) = M \text{ in } A(x,y) \)
  - \( B \) is \( A(x,y) \) if \( M \) is a pair; otherwise, \( B \) is 0.

- \( C(M) := \text{case } M \text{ is 0 : } 0 \text{ suc}(\( x \)) : \text{C}(\( x \)) \)
  - \( C \) is 0 when \( M=0 \)
  - \( C \) is \( C(M-1) \) when \( M > 0 \)
  - \( C \) is 0 if \( M \notin \mathbb{N} \)

Process Scope and Extrusion

- \((n)P\)
  - \( P \) with the name \( n \) bound to \( P \)'s scope.

Example:
\( A(M) := (v \epsilon^0) <e <^c [\epsilon^0] <M >.0 \)
  - Send private channel \( e^c \) on \( c^e \) then become nil.

\( B := e^c(x).x(y).0 \)
  - Receive channel \( x \) on \( c^e \) then receive \( y \) on \( x \) then become 0.

\( C := (v \epsilon^0)(A(M) \mid B) \)
  - \( C \) is \( A \) in parallel with \( B \). Channel \( c^e \) is only in \( A \)'s and \( B \)'s scope.
Spi Calculus Extensions

- \( [M]_N \): term representing the message “\( M \) encrypted with the key \( N \).”
- **case** \( L \) of \( \{x\}_N \) in \( P : P[M/x] \) provided that \( L = [M]_N \) otherwise it is \( 0 \).

Examples:

- \( A(M) := c^e[M] \) : \( 0 \)
  - \( A \) sends \( M \) encrypted with \( k \) on \( c^e \) and then becomes nil.
- \( B := c^e(y).\text{case } x \text{ of } \{x\}_k \text{ in } F(x) \)
  - \( B \) receives \( y \) on \( c^e \) and decrypts \( y \) into \( x \) using \( k \). \( B \) then become \( F(x) \).
- \( C(M) := (\forall e^d)k(A(M) | B) \)
  - \( C \) is \( A \) and \( B \) with channel \( c^e \) and key \( k \).

Process Equivalence

- In Pi Calculus, we write \( P = Q \) iff \( P \) and \( Q \) are indistinguishable to a separate process \( R \).
- In Spi Calculus we write \( P(M) = P(M') \) iff given the two process instances a separate process \( R \) cannot tell which instance is the instance of \( M \) and which the instance of \( M' \).

Pi Calculus Example

**Message 1**: \( A \to B : M \) on \( e^{ch} \)

\[
\begin{align*}
A(M) & := e^{ch} < M > \\
B & := e^{ch}(x).F(x) \\
Inst(M) & := (\forall e^{ch})(A(M) | B)
\end{align*}
\]

- Principal \( A \) sends message \( M \) on channel \( e^{ch} \) to principal \( B \).
- \( e^{ch} \) is restricted, only \( A \) and \( B \) have access to \( e^{ch} \).
- \( Inst(M) \) is one instance of the protocol.
Pi Calculus Example (cont.)

- Specification:
  \[ A(M) := c(ab) < M > \]
  \[ B_{spe}(M) := c(ab)(x).F(M) \]
  \[ Inst_{spe}(M) := (ve^{ab})(A(M) \parallel B_{spe}(M)) \]
- Difference between protocol and specification:
  \( B_{spe}(M) \) is a variant, which receives input from \( A \) and acts like \( B \) when \( B \) receives \( M \).

Security Properties

- Authenticity property:
  \( Inst(M) = Inst_{spe}(M) \), for all \( M \).
  The protocol with message \( M \) is indistinguishable from the specification with message \( M \), for all messages \( M \).
- Secrecy property:
  \( Inst(M) = Inst(M') \) if \( F(M) = F(M') \), for all \( M, M' \).
  If \( F(M) \) is indistinguishable from \( F(M') \), then the protocol with message \( M \) is indistinguishable from the protocol with message \( M' \).
  - These security properties hold because of the restriction on the channel \( c^{ab} \).

Channel Establishment Example

- Abstract and simplified version of the Wide Mouthed Frog protocol
- proposed by Michael Burrows in 1989
- passes a restricted channel from \( A \) to \( B \) via restricted channels to \( S \).
Channel Establishment Example (cont.)

- channels instead of the keys
- channel establishment and data communication happen only once

Message 1: $A \rightarrow S$: \( c^{ab} \) on \( c^{as} \)
Message 2: $S \rightarrow B$: \( c^{ab} \) on \( c^{sb} \)
Message 3: $A \rightarrow B$: $M$ on \( c^{ab} \)

Protocol Implementation

- $A(M) := (v_{c^{ab}})^{cs} (c_{as}^{ab} M)$
  - $A$ sends channel \( c^{ab} \) over \( c^{as} \) then sends $M$ over \( c^{sb} \).
- $S := c^{as}(x) (c_{sb}^{ab} x)$
  - $S$ forwards $x$ from $c^{as}$ to $c^{sb}$.
- $B := c^{sb}(x) x (y) F(x)$
  - $B$ receives channel $x$ on \( c^{sb} \) and receives $y$ on $x$.
- $Inst(M) := (v_{c^{as}})(v_{c^{sb}}) (A(M) \mid S \mid B)$
  - $Inst$ is the composition of $A, S$ and $B$.

Specification

In the specification, $A(M)$ and $S$ are same as above,

- $B_{spec}(M) := c^{sb}(x) x (y) F(M)$
  - Here $B_{spec}$ is similar to $B$ except it knows what $M$ is already for authenticity checking.
- $Inst(M)_{spec} := (v_{c^{as}})(v_{c^{sb}}) (A(M) \mid S \mid B_{spec}(M))$

The authenticity and secrecy properties hold.
Spi Calculus Example

Same as the first Pi example except that a key is used to insure secrecy.

• Message 1: A→B: \{M\}_{ab} on c^{ab}
  - A(M) := c^{ab} \langle {M} \rangle_{ab} >
    • A sends a shared key encrypted message M on c^{ab}.
  - B := c^{ab}(x).\text{case } x \text{ of } y |_{ab} \text{ in } F(y)
    • B decrypts x into y.
  - \text{Inst}(M) := (v k_{ab})(A(M) \mid B)
    • The key k^{ab} is restricted to only A and B.

Spi Example Specification

Specification:
• A(M) := c^{ab} \langle {M} \rangle_{ab} >
• B_{spec}(M) := c^{ab}(x).\text{case } x \text{ of } y |_{ab} \text{ in } F(M)
• \text{Inst}_{spec}(M) := (v k^{ab})(A(M) \mid B_{spec}(M))

Authenticity and secrecy properties are confirmed under a coarse-grained equivalence since an observer can definitely distinguish between P(M) and P(M').

Key establishment in Spi Calculus

• Same as the Pi frog protocol with key used instead of restricted channels.

1. new key k^{as} under k^{a}
2. new key k^{sb} under k^{b}
3. Data under new key k^{as}

Message 1: A→S: \{k^{as}\}_{as} on c^{as}
Message 2: S→B: \{k^{sb}\}_{sb} on c^{sb}
Message 3: A→B: \{M\}_{ab} on c^{ab}
Protocol

- $A(M) := (vk)_{\{k\}} \cdot (vk)_{\{M\}}$
  - $A$ sends a key $k^a$ to the server $S$ and uses $k^a$ to encrypt $M$ to send to $B$.
- $S := (vk)_{\{x\}} \cdot \text{case } x \text{ of } \{y\} \cdot \text{in } (vk)_{\{y\}}$
  - $S$ forwards the key contained in $x$ via the shared key $k^a$.
- $B := (vk)_{\{x\}} \cdot \text{case } x \text{ of } \{y\} \cdot \text{in } (vk)_{\{y\}} \cdot \text{case } z \text{ of } \{w\} \cdot \text{in } F(w)$
  - $B$ receives and decrypts the key in $x$ then uses that key to get the message $w$.
- $\text{Inst}(M) := (vk)_{\{k\}} \cdot (vk)_{\{A(M) \mid S \mid B\}}$

Specification

- Principals $A(M)$ and $S$ are the same as in the protocol;
  - $B_{\text{spec}}(M) := (vk)_{\{x\}} \cdot \text{case } x \text{ of } \{y\} \cdot \text{in } (vk)_{\{y\}} \cdot \text{case } z \text{ of } \{w\} \cdot \text{in } F(M)$
  - $\text{Inst}_{\text{spec}}(M) := (vk)_{\{k\}} \cdot (vk)_{\{A(M) \mid S \mid B_{\text{spec}}(M)\}}$
- The specification is more complex than the protocol but $B_{\text{spec}}(M)$ applies $F$ only to the data from $A$ and not to a message resulting from an attack or error.

Complete Authentication

Example (with a flaw)

- A server and $n$ other principals
- Each principal’s input channels are public and are named as $c^1, c^2, \ldots, c^n$.
- Server shares a pair of keys with each other principal, $k^n$ and $k^a$.
- Message sequence:
  - Message 1: $A \rightarrow S: A, [B, k^{ab}]_{k^a}$ on $c^n$
  - Message 2: $S \rightarrow B: [A, k^{ab}]_{k^a}$ on $c^n$
  - Message 3: $A \rightarrow B: A, [M]_{k^b}$ on $c^n$
Instance of the protocol

- We have two principals (A and B) and the message sent after key establishment.
- Instance I is a triple (i,j,M) where
  i: source address, j: destination address
- Send(i,j,M) := (vk(ck < (i, j, k)ks > c < (i, M)ks >))
- Recv(j) := c(ycipher).case y_cipher of {xa, x_key}ks in c(z_cipher) x_a is za case z_cipher of {z_cipher}ks in F(xa, za, z_cipher)

Instance of the protocol

(Sending)

- Send(i,j,M) := (vk(ck < (i, j, k)ks > c < (i, M)ks >))
  - Creates a key k, sends to the server along with the names i and j of the principals of the instance.
  - Sends M under k with its name j.

Instance of the protocol

(Receiving)

- Recv(j) := c(y_cipher).case y_cipher of {xa, x_key}ks in c(z_cipher) x_a is za case z_cipher of {z_cipher}ks in F(xa, za, z_cipher)
  - Waits for a message y_cipher, from server, extracts x_key from this message
  - Then waits for a message z_cipher under this key
  - At the end applies F to the name x_a of the presumed sender, j and to z_cipher of the message.
Server

The server $S$ is the same for all instances:

\[ S := c(x_a, x_{cipher}) \]
\[ \Pi_{i \in 1..n} \{ x \text{ is } i \} \text{ case } x_{cipher} \text{ of } \{ x_i, x_{cipher} \} \text{ in } \]
\[ \Pi_{i \in 1..n} \{ x \text{ is } i \} \notin \{ x_i, x_{cipher} \} \text{ end } \]

$S$ receives a key that selects the correct branch to forward the key to the correct $j$.

\[ \Pi_{i \in 1..k} P_i \text{ is the k-way composition } P_1 | \ldots | P_k \]

Whole System

\[ \text{Sys}(I_1, \ldots, I_m) := (vk^S)(vk^S) \]
\[ (\text{Send}(I_1) \ldots \text{Send}(I_m)) \]
\[ !S \]
\[ !\text{Recv}(1) \ldots !\text{Recv}(n) \]

- Where $(vk^S)(vk^S)$ stands for $(vk^{i_1}) \ldots (vk^{i_m})$
- $(vk^{i_1}) \ldots (vk^{i_m})$ and $\text{Sys}(I_1, \ldots, I_m)$ represents a system with $m$ instances of the protocol.

The Flaw

- The protocol is vulnerable to a replay attack.
- System: $\text{Sys}(I, I')$ where $I = (i, j, M), I' = (i, j, M')$
- An attacker can replay messages of one instance and get them mistaken for messages of the other instance.
- So $M$ will be passed to $F$ twice and $\text{Sys}(I, I')$ could execute two copies of $F(I, M)$ although $\text{Sys}(I, I')$ can run $F$ for both instances $F(I, M)$ and $F(I', M')$ only once.
- Therefore the authenticity equation doesn’t hold.

$\text{Inst}(M) = \text{Inst}_{seq}(M)$, for all $M$. 
Complete Authentication Example (repaired)

• To protect previous protocol against replay attacks, nonce handshakes (tag in the message to authenticate the sender) are added.
• The new protocol, informally looks like:
  – Message 1: $A \rightarrow S: A$ on $c$
  – Message 2: $S \rightarrow A: N_s$ on $c$
  – Message 3: $A \rightarrow S: A|A.A.B.k^s.s_{N_s}$ on $c$
  – Message 4: $S \rightarrow B: *$ on $c$
  – Message 5: $B \rightarrow S: N_b$ on $c$
  – Message 6: $S \rightarrow B: \{S.A.B.k^s.s_{N_b}\}$ on $c$
  – Message 7: $A \rightarrow B: A.(M |s_{N_b})$ on $c$
• See Appendix for implementation.

What we get with Spi Calculus?

• Protocols in Spi Calculus are tedious.
• Good for proofing authenticity and secrecy.
• The scope of errors that it can find are limited though
• Spi Calculus is not as general as other logics, but this lack of generality allows us more confidence in the properties we can prove.

Tool Support

• Spi Calculus does not have any direct tool support.
• Security proprieties must be proven by humans.
• There is however a protocol language Cryptc that is based on Spi Calculus.
Cryptc

- Redefines Spi calculus’s grammar
- Adds protocol beginnings and endings
- Protocols are considered secure if every protocol ending has a distinct beginning.
- Cryptc performs an exhaustive search for paths that generate an end without a begin.

Online References

- A Calculus for Cryptographic Protocols: The Spi Calculus
  Martin Abadi and Andrew D. Gordon
  Digital SRC Research Report 149
  January 25, 1998
  http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-149.html

- Authenticity by Typing for Security Protocols
  A.D. Gordon and A.S.A. Jeffrey
- The Cryptc webpage:
  http://cryptc.cs.depaul.edu/intro.html
Appendix : Corrected Protocol

\[ \text{Send}(i,j,M) := \langle i \rangle | \]
\[ c(s_{\text{nonce}}) \]
\[ (v_k) \langle j \rangle (l_k k_v x_{\text{nonce}_{k_v}}) \cong \langle i \rangle \langle M \rangle | \]
\[ S := c(s_{\text{nonce}}) P_{\text{init}} \left[ x_{\text{is}} \cong \langle i \rangle \right] \]
\[ c(x_{\text{nonce}}) \left[ x_{\text{is}} \cong \langle i \rangle \right] \]
\[ \text{case } \text{cipher of } \left[ y_{\text{is}}, x_k, y_{\text{nonce}} \right] \text{ in } \]
\[ P_{\text{init}}, x_{\text{is}} \cong \langle i \rangle \left[ x_{\text{is}} \cong \langle i \rangle \right] \left[ y_{\text{nonce}} \cong N_j \right] \]
\[ \langle i \rangle \cong \langle N_j \rangle \left[ c_j (y_{\text{nonce}}) \cong \langle i \rangle \langle M \rangle \rangle \right] \]

\[ \text{Recv}(j) := c(w) \left[ v_{N_j} \langle N_j \rangle \cong \langle i \rangle \right] \]
\[ c(y_{\text{nonce}}) \left[ x_{\text{is}} \cong \langle i \rangle \right] \]
\[ \text{case } \text{cipher of } \left[ y_{\text{is}}, x_k, y_{\text{nonce}} \right] \text{ in } \]
\[ P_{\text{init}}, x_{\text{is}} \cong \langle i \rangle \left[ x_{\text{is}} \cong \langle i \rangle \right] \left[ y_{\text{nonce}} \cong N_j \right] \]
\[ c(x_{\text{nonce}}) \left[ x_{\text{is}} \cong \langle i \rangle \right] \]
\[ \text{case } \text{cipher of } \left[ z_{\text{is}}, x_{\text{nonce}} \right] \text{ in } \]
\[ F(k_{\text{nonce}}, l_{\text{plain}}) \]

\[ \text{Sys}(I_1, \ldots, I_m) := (v_k) (v_{N_k}) \]
\[ (\text{Send}(I_1) \ldots \text{Send}(I_m) | \]
\[ \text{IS} | \]
\[ !\text{Recv}(I_1) \ldots !\text{Recv}(n) ) \]

• Authenticity:
\[ \text{Sys}(I_1, \ldots, I_m) = \text{Sys}_{\text{spec}}(I_1, \ldots, I_m) \text{ for any} \]
\[ \text{instances } I_1, \ldots, I_m. \]
\[ \text{This property holds because of the nonces.} \]

• Secrecy:
\[ \text{Sys}(I_1, \ldots, I_m) = \text{Sys}_{\text{spec}}(I_1, \ldots, I_m) \text{ if each pair} \]
\[ (I_j, I_j), \ldots, (I_m, I_m) \text{ is indistinguishable.} \]