

A formal implementation of value commitment

Cédric Fournet

Nataliya Guts

Francesco Zappa
Nardelli

“Pessimistic” approach to security



- Presumably bad participants
- Mandatory checks
- Strong security invariant

Example: airport security

“Optimistic” approach to security (this talk)



- Presumably honest participants
- Logs rather than checks
- *A posteriori* detection
- Efficiency

Example: traders' transactions at Société Générale

Logs

Optimistic protocols rely on logs to detect misbehaviors

- Audit logs are widely used in practice
- Secure logs implementations are well understood.

But what should the log contain?

What logs should contain

According to NIST,
“sufficient information to establish what events occurred
and who (or what) caused them”.

We want an a priori guarantee of logging enough data.

A sample optimistic protocol

To study logs, we focus on **value commitment**:

- ① A blinded value can be committed to.
- ② It can be revealed the later.

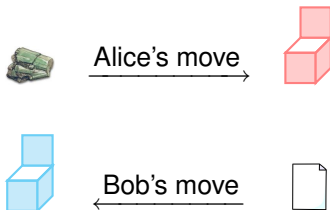
The commitment of the value is trusted.

The value cannot change after commitment.

Alice and Bob play Rock-Paper-Scissors

Players should simultaneously reveal their moves.

In a distributed setting, Bob can cheat to always win!



Rock-Paper-Scissors with commitment



Alice's committed move



Bob's committed move



Alice's opening



Bob's opening



Other optimistic protocols

Value commitment is a building block of:

- sealed bid auctions
- e-voting
- mental poker
- online games (Networked Virtual Environments)

Our formal case study

How to validate the usage of logs
in an optimistic implementation
of value commitment?

What we do

- We formalize commitment semantics in an ideal world
- We implement the semantics using secure logs.
- We establish correctness and reliable against potential cheaters.

Idealized commitment datatype

Source language

A concurrent language with committable cells
(extended applied π -calculus [Abadi, Fournet, '01])

A source system is a parallel composition of

- threads run by principals: $p[P]$
- global state of committable cells $l.(p)$ $l.(p V)$

Source systems: threads

A thread P may include

- ✿ parallel composition of threads $P_1 | P_2$
- ✿ name restriction (static scoping) $\nu c . P$
- ✿ reception on channel c $x = c?.P$
- ✿ sending of term V on channel c $c!\langle V \rangle.P$
- ✿ creating a location $x = \text{newloc} . P$
- ✿ committing a location $x = \text{commit } V \ell . P$

Source systems: cells

$\ell.(p V)$ denotes a cell ℓ owned by p and containing V

$\ell.(p)$ denotes an empty cell committable once by p

The owner can distribute partial read *capabilities*



$\ell.\text{Idc}(p)$: commitment + owner



$\ell.\text{Rd}(p V)$: commitment + owner + value

Operations on capabilities

(implemented using applied π -calculus equations)

Reading

$$\text{read}(\ell.\mathbf{Rd}(p\ V)) = V \qquad \text{get_prin}(\ell.\mathbf{Idc}(p)) = p$$

Note: *the value cannot be extracted from an Idc.*

Downgrading

$$\text{get_idc}(\ell.\mathbf{Rd}(p\ V)) = \ell.\mathbf{Idc}(p)$$

Commitment semantics

Creating a new cell yields its fresh name.

$$p[x = \text{newloc} . P] \longrightarrow v \ell . (\ell . (p) \mid p[P\{\ell/x\}])$$

Committing a cell yields an rd capability.

$$\ell . (p) \mid p[x = \text{commit } V \ell . P] \longrightarrow \ell . (p V) \mid p[P\{\ell . \text{Rd}(p V)/x\}]$$


Capabilities can be **communicated** as ordinary values.


$$p_1[c! \langle V \rangle . P_1] \mid p_2[x = c? . P_2] \longrightarrow p_1[P_1] \mid p_2[P_2\{V/x\}]$$

Idealized commitment guarantee


A cell can only be committed once by its owner


Rock-Paper-Scissors

\mathcal{A} [
 $\ell^{\mathcal{A}} = \text{newloc}.$
 $\ell_{rd}^{\mathcal{A}} = \text{commit}$  $\ell^{\mathcal{A}}.$
 $c!\langle \text{get_idc}(\ell^{\mathcal{A}}) \rangle.$

 Alice's commitment \longrightarrow


$y_1^{\mathcal{B}} = c?.$

\longleftarrow Bob's commitment 

\mathcal{B} [
 $\ell^{\mathcal{B}} = \text{newloc}.$
 $\ell_{rd}^{\mathcal{B}} = \text{commit}$  $\ell^{\mathcal{B}}.$
 $y_1^{\mathcal{A}} = c?.$


$c!\langle \text{get_idc}(\ell_{rd}^{\mathcal{B}}) \rangle.$


$c!\langle \ell_{rd}^{\mathcal{A}} \rangle.$


 Alice's opening \longrightarrow

$y_2^{\mathcal{A}} = c?.$
 $c!\langle \ell_{rd}^{\mathcal{B}} \rangle.$


$y_2^{\mathcal{B}} = c?.$


\longleftarrow Bob's opening 

if $\text{read}(y_2^{\mathcal{A}}) =$ 
then "I won!" else ...]


if $\text{read}(y_2^{\mathcal{B}}) =$ 
then "I won!" else ...]

... + authenticating principals


\mathcal{A} [
 $\ell^{\mathcal{A}} = \text{newloc}.$
 $\ell_{rd}^{\mathcal{A}} = \text{commit}$  $\ell^{\mathcal{A}}.$
 $c!\langle \text{get_idc}(\ell^{\mathcal{A}}) \rangle.$

 Alice's commitment \longrightarrow


$y_1^{\mathcal{B}} = c?.$
if $\text{get_prin}(y_1^{\mathcal{B}}) = \mathcal{B}$ then


\longleftarrow Bob's commitment 


$c!\langle \ell_{rd}^{\mathcal{A}} \rangle.$

 Alice's opening \longrightarrow

$y_2^{\mathcal{B}} = c?.$


if $\text{read}(y_2^{\mathcal{B}}) =$ 
then "I won!" else ...]

\longleftarrow Bob's opening 


\mathcal{B} [
 $\ell^{\mathcal{B}} = \text{newloc}.$
 $\ell_{rd}^{\mathcal{B}} = \text{commit}$  $\ell^{\mathcal{B}}.$
 $y_1^{\mathcal{A}} = c?.$
 if $\text{get_prin}(y_1^{\mathcal{A}}) = \mathcal{A}$ then


$c!\langle \text{get_idc}(\ell_{rd}^{\mathcal{B}}) \rangle.$

$y_2^{\mathcal{A}} = c?.$
 $c!\langle \ell_{rd}^{\mathcal{B}} \rangle.$


if $\text{read}(y_2^{\mathcal{A}}) =$ 
then "I won!" else ...]

... + authenticating cells


\mathcal{A} [
 $\ell^{\mathcal{A}} = \text{newloc}.$
 $\ell_{rd}^{\mathcal{A}} = \text{commit}$  $\ell^{\mathcal{A}}.$
 $c!\langle \text{get_idc}(\ell^{\mathcal{A}}) \rangle.$


 Alice's commitment \longrightarrow


$y_1^{\mathcal{B}} = c?.$
 if $\text{get_prin}(y_1^{\mathcal{B}}) = \mathcal{B}$ then


\longleftarrow Bob's commitment 

$c!\langle \ell_{rd}^{\mathcal{A}} \rangle.$

 Alice's opening \longrightarrow


$y_2^{\mathcal{B}} = c?.$
 if $\text{get_idc}(y_2^{\mathcal{B}}) = y_1^{\mathcal{B}}$ then
 if $\text{read}(y_2^{\mathcal{B}}) =$ 
 then "I won!" else ...]

\longleftarrow Bob's opening 

\mathcal{B} [
 $\ell^{\mathcal{B}} = \text{newloc}.$
 $\ell_{rd}^{\mathcal{B}} = \text{commit}$  $\ell^{\mathcal{B}}.$
 $y_1^{\mathcal{A}} = c?.$
 if $\text{get_prin}(y_1^{\mathcal{A}}) = \mathcal{A}$ then

$c!\langle \text{get_idc}(\ell_{rd}^{\mathcal{B}}) \rangle.$

$y_2^{\mathcal{A}} = c?.$
 $c!\langle \ell_{rd}^{\mathcal{B}} \rangle.$
 if $\text{get_idc}(y_2^{\mathcal{A}}) = y_1^{\mathcal{A}}$ then

if $\text{read}(y_2^{\mathcal{A}}) =$ 
 then "I won!" else ...]

Implementation using logs

Target language

Standard applied π -calculus with

- no memory cells
- cryptographic hashes and public key signing

$$\text{verify}(v, \text{sign}(v, \text{sk}(m)), \text{pk}(m)) = \text{ok}$$

- an observer can decompose constructed terms

$$\text{tag}_i(\text{tag}(x_1, \dots, x_n)) = x_i$$

Compiling . . .

- **systems**

$\llbracket p[P] \rrbracket$ publishes p 's public key and compiles $\llbracket P \rrbracket_p$

- **threads**: compilation is mostly homomorphic

$$\llbracket P_1 \mid P_2 \rrbracket_p = \llbracket P_1 \rrbracket_p \mid \llbracket P_2 \rrbracket_p \quad \llbracket v c . P \rrbracket_p = v c . \llbracket P \rrbracket_p$$

$$\llbracket c! \langle V \rangle . P \rrbracket_p = c! \langle \llbracket V \rrbracket \rangle . \llbracket P \rrbracket_p$$

- **cells**: local state is stored on private channels

$$\llbracket x = \text{newloc} . P \rrbracket_p = v c . (c! \langle \text{None} \rangle \mid \llbracket P \rrbracket_p \{^c/x\})$$

- **capabilities**: simple cryptography is needed

Implementation of capabilities

We assign a unique identifier s to each cell $\ell.(p V)$.



read

$\text{rd}(\text{pk}(p), s, \llbracket V \rrbracket, w)$



committed id

$\text{idc}(\text{pk}(p), h(s) + h(s + \llbracket V \rrbracket), w)$


where

$w = \text{sign}(h(s) + h(s + \llbracket V \rrbracket), \text{sk}(p))$ (the seal).

Receiving an invalid capability

$\mathcal{A}[\dots]$
 $y = c?$

if Bob did not sign $v_1 + v_2$ in w ,
STOP

Bob's commitment 
 $y : \text{idc}(\text{pk}(\mathcal{B}), v_1 + v_2, w)$

Receiving double committed capabilities

\mathcal{A} [

...

$y = c?$.

Bob's commitment 

$\text{idc}(\text{pk}(\mathcal{B}), v_1 + v_2, \text{sign}(v_1 + v_2, \text{sk}(\mathcal{B})))$

$y_2 = c?$.

Bob's opening 

$\text{rd}(\text{pk}(\mathcal{B}), s, \text{sign}(h(s) + h(s + \text{scissors}), \text{sk}(\mathcal{B})))$

if $v_1 = h(s)$ but
not $v_2 = h(s + \text{scissors})$,
STOP

How to automate these checks

- Forward all the received seals on channel *log*
- Check that there is at most one seal per cell on *log*:

log?(*y*₁).*log*?(*y*₂).

if check_idc(*y*₁) and check_idc(*y*₂) then

if idc₂₁(*y*₁) = idc₂₁(*y*₂) and idc₂(*y*₁) ≠ idc₂(*y*₂)

then *bad!*⟨get_prin(*y*₁)⟩

check_idc(*x*) $\stackrel{\text{def}}{=} \text{verify}(\text{idc}_2(\mathbf{x}), \text{idc}_3(\mathbf{x}), \text{idc}_1(\mathbf{x})) = \text{ok}$

Operational correspondance

We study system **interactions with contexts** using LTS

Traces of source systems
(idealized source semantics)



Traces of source systems translations
(realistic semantics in applied π -calculus)

Optimistic security

Correctness

Let $\llbracket A \rrbracket$ be the translation of source system A .

Therem 1

Given a source system A , given a source trace $A \xrightarrow{\phi} A'$,
there exists a target trace $\llbracket A \rrbracket \xrightarrow{\psi} \llbracket A' \rrbracket$.

Completeness

Theorem 2

Given a source system A , given a trace $\llbracket A \rrbracket \xrightarrow{\psi} S$,

- either there exists a trace $A \xrightarrow{\phi} A'$ such that

$$S \xrightarrow*_D \llbracket A' \rrbracket,$$

- or S accuses some principal of cheating.

Theorem 3

Given a source system A , given a trace $\llbracket A \rrbracket \xrightarrow{\psi} S$,

if S blames a principal p , then p does not follow our implementation.

Conclusions

- sample optimistic construct and its implementation
- proved full abstraction for labelled traces
- OCaml prototype implementing committable cells
- a first work on the formalisation of audit logs

Questions?

- sample optimistic construct and its implementation
- proved full abstraction for labelled traces
- OCaml prototype implementing committable cells
- a first work on the formalisation of audit logs

<http://www.msr-inria.inria.fr/projects/sec/logs/>

So what should the log contain?

Protocols based on value commitment
should log **all the sealed Idc and Rd capabilities**
handled during the execution.