Secrecy-Oriented, Computationally Sound First-Order Logical Analysis of Cryptographic Protocols

Gergei Bana (ENS Cachan)
with Koji Hasebe (University of Tsukuba),
Mitsuhiro Okada (Keio University)

Computational and Symbolic Analysis

- **Traditionally, two ways of looking at and analyzing cryptographic protocols**
  - **Symbolic**
    - Uses high level formal language
    - Amenable to automatization
    - Treated encryption as perfect, black box operation
    - Accuracy is unclear as neglects probability, complexity
  - **Computational**
    - More detailed description using probabilities and complexity
    - Proofs by hand (reductions)
    - More difficult
    - More accurate
- **Trying to link since 2000 (Abadi Rogaway)**

Success of Symbolic Analysis

- **Needham Schroeder protocol**

$$1.\ A \rightarrow B : \{\!|A\ N_1|\!\}_{K_B}$$

$$2.\ B \rightarrow A : \{\!|N_1\ N_2|\!\}_{K_A}$$

$$3.\ A \rightarrow B : \{\!|N_2|\!\}_{K_B}$$

- **Man in the middle attack:**
  - Possible to find an attack purely symbolically
  - Treating encryption as perfect black box
  - B thinks he is communicating with A (authentication fails)
  - M learns N_2 which was meant to be a secret of A and B (secrecy fails)

$$1.\ A \rightarrow M : \{\!|A, N_1|\!\}_{K_M}$$

$$2.\ M(A) \rightarrow B : \{\!|A, N_1|\!\}_{K_B}$$

$$3.\ B \rightarrow M(A) : \{\!|N_1, N_2|\!\}_{K_A}$$

$$4.\ M \rightarrow A : \{\!|N_1, N_2|\!\}_{K_A}$$

$$5.\ A \rightarrow M : \{\!|N_2|\!\}_{K_M}$$

$$6.\ M(A) \rightarrow B : \{\!|N_2|\!\}_{K_B}$$

- **Searching for and eliminating the possibility of symbolic attacks is a good thing**

Computational View

- **Computational execution is probabilistic polinomial time.**
  - Sequence of random processes indexed by security parameter η.
- **Negligible function:**
  - f(η): for any n natural, there is an η_0 such that whenever $\eta > \eta_0$, f(η) < 1/ η^n.
- **Adversary:**
  - An adversary is successful if it can do bad things with non-negligible probability:
- **Security proofs rely on the assumption that certain operations (e.g. computing logarithm) are infeasible**
  - Proof by reduction
- **Encryption is not perfect**
  - CCA2 security: the function $$\Pr[\,(e,d) \longleftarrow \mathcal{K}_\eta);\, i \longleftarrow \{0,1\};\; m_0, m_1 \longleftarrow \mathsf{A}_\eta^{\mathcal{D}_1(\cdot)}(e);\; c \longleftarrow \mathcal{E}_\eta(e, m_i);\, j \longleftarrow \mathsf{A}_\eta^{\mathcal{D}_2(\cdot)}(e,c) :\; i = j\,] - \tfrac{1}{2}$$ is neglibible in η

Ways of Soundness

- Aim: Symbolic analysis provide computational guarantees
- By now long history, active adversaries in two groups:
  - Two-world view
    - Symbolic and computational executions are formalized separately as well as security properties
    - Explicitly formalized what the adversary can do (eg. decrypt if he has the key)
    - Soundness: Try to prove that no successful symbolic (Dolev-Yao) attacker implies no successful computational attacker.
    - Such are
      - Reactive Simulatability of M. Backes, B. Pfitzmann, M. Waidner
      - D. Micciancio, B. Warinschi, V. Cortier (mapping lemma)
      - V. Cortier, H. Comon-Lundh (soundness of observational equivalence)
  - Logical view
    - Only computational execution, no explicit formal reasoning about adversary.
    - Logical theory axiomatizes the relevant properties of cryptographic primitives.
    - Security properties are directly proven from computationally sound axioms and derivation rules (I.e. works directly in the computational model)
      - Computational Protocol Compositional Logic of Stanford (John Mitchell's group)
      - The proof system presented here

Plan

1. **Parsing and limitations of two-world view**
2. **Build a first-order system from scratch**
3. **Summary of what we built**
4. **How it works**

1. Parsing and limitations of two-world view

Parsing and adversarial capabilities

- **Computational soundness and parsing**
  - Two-world view: No successful symbolic adversary implies no successful computational
    - In the soundness proof, to each computational trace, you have to create a symbolic (trace lifting)
    - Symbolic expressions that are different but have the same computational interpretation cause problems
    - Limited soundness theorems
- **E.g.**
  - E.g. $\{N\}_K^R = \{N'\}_{K'}^{R'}$ may happen computationally; problems with key forging
  - Or, type-flaw attacks depend on things like $\langle n,Q\rangle = N$
  - In such cases, trace-lifting fails.
- **Such equalities must be included in adversarial capabilities**
  - Noone has been able to do this

Two type-flaw attacks on NSL

- Reminder:

  1. $A \to B$: $\{N_1, A\}_B$ 2. $B \to A$: $\{N_1, N_2, B\}_A$ 3. $A \to B$: $\{N_2\}_B$

- Using $\langle n,Q \rangle = N$ with two parallel sessions
  - Take a Q agent name such that for any value N, there is an n with $\langle n,Q \rangle = N$

3. $A \xrightarrow{\{N_2\}_B} B$

Q catches $\{N_2\}_B$

1. $Q \xrightarrow{\{N_2\}_B = \{n,Q\}_B} B$

$\downarrow$ B generates N

2. $Q \xleftarrow{\{n,N,B\}_Q} B$

Q computes n and $N_2 = \langle n,Q \rangle$

- Q is not honest agent
- Assigned name attack: Honest name may also be bad.
- In Dolev-Yao framework, such possibilities have to be explicitely included in adversarial capabilities

Unforgeability and symmetric NS

- **Symmetric Needham Schroeder protocol**

$$
\begin{aligned}
&1.\ A \to B : A\\
&2.\ B \to A : \{A, N_1\}_{BT}\\
&3.\ A \to T : \langle A, B, N_2, \{A, N_1\}_{BT}\rangle\\
&4.\ T \to A : \{N_2, B, K, \{K, N_1, A\}_{BT}\}_{AT}\\
&5.\ A \to B : \{K, N_1, A\}_{BT}\\
&6.\ B \to A : \{N_3\}_K\\
&7.\ A \to B : \{N_3 - 1\}_K
\end{aligned}
$$

- Clearly, if encryption can be forged, for example, using another key, **A** has no way to know that the message in step 6 came from **B**.

- In A symbolic execution, if the adversary sends a message of the form $\{N_4\}_{K'}$, **A** will halt, but computationally not necessarily.

- $\mathbf{Q} \xrightarrow{\{n\}_{K'} = \{m\}_K} \mathbf{A} \xrightarrow{\{m-1\}_K} \mathbf{B}$

Key Cycles

- **Two-world soundness theorems only work when for protocols that do not admit key cycles.**
  - So, first by symbolic analysis, show:
    - There is no successful symbolic attacker
    - There cannot be key cycles.
  - If symbolically there are no key cycles, does that mean there aren't computationally?
    - Probably, when you have unarbitrary parsing.
    - Otherwise?
- **Also: a protocol that admits key cycle, may still be secure.**

2. Build a first-order system from scratch

Aims

- For agreement, authentication and secrecy for protocols using long term public keys, long term shared keys and session keys.
- Derive security properties with a first order theory from what we surely know in the computational world
  - I.e. instead of listing what the adversary may do, list what he surely cannot spoil.
  - If e.g. the pairing is such that $\langle n,Q \rangle = N$ can be ruled out, then we can use $\langle n,Q \rangle \neq N$
- No need for additional assumption on key cycles
- Intuitive (proof should follow intuition)
- Keep it as simple as possible
- But, can prove protocols (done: NSL, symmetric NS, Otway-Rees all for unbounded sessions)

What we need 1

We need be able to talk about traces, what agents sent, received, and when honestly created new things in some session

$$A\ generates^i\ N_1 \qquad A\ sends^i \{N_1, A\}_Q^{r_1} \qquad A\ receives^i \{N_1, N_2, Q\}_A^{s_2}$$

Here we want to mean that the agent parsed the received and sent messages according to how the term indicates.

For comparing just bit strings corresponding to the terms, we will include the following notation:

So $\{N_1, A\}_Q^{r_1}$ means a parsed term, where as $\overline{\{N_1, A\}_Q^{r_1}}$ means the bit string (more precisely, the sequence of random variables of bit strings)

So we surely know e.g.:

$$\langle t_1, t_2 \rangle \neq N$$

$$\{t\}_{Q_1 Q_2}^s \neq \langle t_1, t_2 \rangle$$

$$\langle t_1, t_2 \rangle = \langle t_1', t_2' \rangle \rightarrow t_1 = t_1' \wedge t_2 = t_2'$$

$$\{t\}_Q^s = \{t'\}_{Q'}^{s'} \rightarrow t = t' \wedge Q = Q'$$

For N_1, N_2, A_1, A_2, r_1, r_2 honest:

$$N_1 \neq N_2 \rightarrow \overline{\langle N_1, t \rangle} \neq N_2$$

$$\overline{\langle t_1, t_2 \rangle} = \overline{\langle t_1', t_2' \rangle} \rightarrow \overline{t_1} = \overline{t_1'} \wedge \overline{t_2} = \overline{t_2'}$$

$$\overline{\{t\}_Q^s} = \overline{\{t'\}_Q^{s'}} \rightarrow \overline{t} = \overline{t'}$$

$$\overline{\{N_1\}_{A_1}^{r_1}} \neq \overline{\{N_2\}_{A_2}^{r_2}}$$

Honest items behave differently, so we need different types for honest things.

We also need subterm relation to talk about sent and receive information.

What we need 2

So far: types for honest things, pairing, encryption, parsed and not parsed terms, generate, send, receive, equality, subterm predicates

We also want secrecy and key usability:

$$Sec(\langle A_1, ..., A_n \rangle, N) \qquad \text{and} \qquad KeySec(\langle A_1, ..., A_n \rangle, K)$$

N is indistinguishable from another nonce generated independently of the protocol for anyone outside A_1, A_2, ..., A_n

K can be securely used by A_1, A_2,..., A_n.

We want to show secrecy inductively: that no honest send action corrupts important nonces and keys. That is, something like:

$Sec(\langle A_1, ..., A_n \rangle, N)$ Send action by honest guy following the protocol $Sec(\langle A_1, ..., A_n \rangle, N)$

But, instead:

$$Sec_\tau(\langle A_1, ..., A_n \rangle, N) \wedge \text{Honest } A \; sends^i_\tau t \text{ according to protocol role } \rightarrow Sec_\tau(\langle A_1, ..., A_n \rangle, t, N)$$

Where $Sec_\tau(\langle A_1, ..., A_n \rangle, t, N)$ means that N is indistinguishable (from independent nonce) for others until τ even if t is revealed to them. Clearly, we will need axioms that specify how we can build up t in the second argument. E.g:

$$(Q \, sends^i_\tau t \vee Q \, receives^i_\tau t) \wedge \tau < \tau' \wedge Sec_{\tau'}(\mathbf{A}, t', N) \rightarrow Sec_{\tau'}(\mathbf{A}, \langle t, t' \rangle, N)$$

What we need 3

So far: types for honest things, pairing, encryption, parsed and not parsed terms, generate, send, receive, equality, subterm, secrecy and key secrecy (usuability), time sections

We will need to talk about traces for roles and security properties:

$$Resp^{B}_{\mathrm{NSL}}[B, i', Q', n_1, N_2, s_1, r_2, s_3] \equiv B\ receives^{i'}\{n_1, Q'\}^{s_1}_{B};\ B\ generates^{i'} N_2;$$
$$B\ sends^{i'}\{n_1, N_2, B\}^{r_2}_{Q'};\ B\ receives^{i'}\{N_2\}^{s_3}_{B}$$

But this is just an abbreviation:

$$Q_1\, acts^{i_1}_{1} t_1; \ldots; Q_k\, acts^{i_k}_{k} t_k \equiv$$

$$\exists \tau_1 \ldots \tau_k (\mathbf{0} < \tau_1 < \ldots < \tau_k \wedge Q_1\, acts^{i_1}_{1,\tau_1} t_1 \wedge \ldots \wedge Q_k\, acts^{i_k}_{k,\tau_k} t_k)$$

We need to link secrecy and authentication: We need will need something that implies this in the NSL protocol:

$$Sec_{\tau}(\langle A, B\rangle, N_1) \wedge\ A\ receives^{i}_{\tau}\{N_1, n_2, B\}^{s}_{A}\ \rightarrow \{N_1, n_2, B\}^{s}_{A} \text{ came from B}$$

Eg:

$$\{t\}^{s}_{A} \sqsubseteq t_2\ \wedge\ A\ receives^{i_2}_{\tau_2} t_2 \wedge\ Sec_{\tau_2}(\mathbf{A}, \{t\}^{s}_{A}, \nu) \wedge \neg Sec_{\tau_2}(\mathbf{A}, t, \nu)$$
$$\rightarrow\ \exists A' t_1 t' i_1 r \tau_1 \Big(A' \sqsubseteq \mathbf{A} \wedge \{t'\}^{r}_{A} \sqsubseteq t_1 \wedge \overline{\{t\}^{s}_{A}} = \overline{\{t'\}^{r}_{A}} \wedge A'\ sends^{i_1}_{\tau_1} t_1;\ A\ receives^{i_2}_{\tau_2} t_2 \Big)$$

3. Summary of what we built

Sorts, Function Symbols

Sorts

$$\left.\begin{array}{r}\texttt{hseed}\\ \texttt{hname} \subseteq \texttt{name}\\ \texttt{hnonce}\\ \texttt{hkey}\\ \texttt{sessionid}\end{array}\right\} \subseteq \texttt{bitstring} \subseteq \texttt{bittree}$$

$$\texttt{timesection} \qquad \texttt{event}$$

Infinitely many variables of each sort

Message terms

$$T \ ::= t \;\Big|\; \overline{T} \;\Big|\; \langle T, T\rangle \;\Big|\; \{T\}^{s}_{Q} \;\Big|\; \{T\}^{s}_{QQ'} \;\Big|\; \{T\}^{s}_{k}$$

t: variable of sort bittree, s: bitstring, Q,Q': names

T: in general sort bittree, but T overline: bitstring

Message terms represent parsed messages, but T overline is the corresponding bitstring

They are in fact infinite sequences of random variables

Computational Model

The usual probabilistic polynomial time execution, the adversary controlling the network. For each fixed security parameter, the execution is a stochastic process with an underlying probability space

Computational structure: Execution together with a non-negligible sequence of subsets D of the underlying sequence of probability spaces.

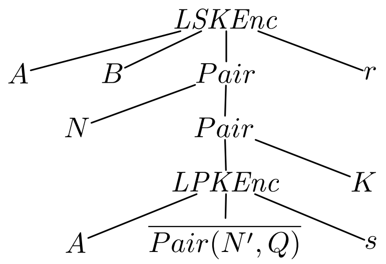
Interpretation of elements of sort bitstring: sequences (in the security par) of random variables on D

Interpretation of elements of sort bittree: Ordered trees with sequences of random var's (over D) on the leaves and encryption or paring on the internal nodes.

Interpretation of elements of sort timesection: Infinite sequence of stopping times on D

Interpretation of elements of sort event: Non-negligible sequence of subsets of D

Tree structure of
$\{N, \langle\{\overline{\langle N', Q\rangle}\}^s_A, K\rangle\}^r_{AB}$



Atomic Formulas

$$\varphi_0 ::= Q\ acts_\tau^i t \ \Big|\ Sec_\tau(\boldsymbol{A}, t, \nu) \ \Big|\ KeySec_\tau(\boldsymbol{A}, t, K) \ \Big|\ t = t' \ \Big|\ t \sqsubseteq t' \ \Big|\ \tau < \tau'$$

acts

is either generates or receives or sends

Q does the corresponding action on section τ in session i doing as much parsing as indicated by t

Sec, KeySec

A is a list of honest names ⟨A, B, C...⟩ for Sec and KeySec to be not false.

ν is either honest nonce or honest key.

Sec (indistinguishability) means that agents other than those listed in *A* together with the adversary, based on their combined view until τ cannot distinguish ν from another nonce (or key) generated independently from the protocol, even if we give them the bit string corresponding to t

KeySec (key usability) means that agents other than those listed in *A* together with the adversary, based on their combined view until τ cannot break the security game (CCA-2) against key K even if we give them the bit string corresponding to t

Formulas

$$\varphi_0 ::= Q\ acts^i_\tau t \;\Big|\; Sec_\tau(\mathbf{A}, t, \nu) \;\Big|\; KeySec_\tau(\mathbf{A}, t, K) \;\Big|\; t = t' \;\Big|\; t \sqsubseteq t' \;\Big|\; \tau < \tau'$$

=

on bitstrings: equality of sequences of random variables up to negligible sets.
on bittrees: labels on internal nodes agree, the leaves are equal up to negl, except the random seed.
But: $\overline{\{t\}^s_{Q_1 Q_2}} = \overline{\{t\}^{s'}_{Q_1 Q_2}} \to \{t\}^s_{Q_1 Q_2} = \{t\}^{s'}_{Q_1 Q_2}$

$\sqsubseteq$

subterm
e.g.: $N \not\sqsubseteq \{\overline{\{N\}^r_B}, N'\}^s_{QQ'}$ $\qquad \overline{\{N\}^r_B} \sqsubseteq \{\overline{\{N\}^r_B}, N'\}^s_{QQ'}$

<

τ is earlier than τ' on all traces

Formulas:

$$\varphi \;::=\; \varphi_0 \Big| \neg\varphi \Big| \varphi \wedge \varphi \Big| \varphi \vee \varphi \Big| \varphi \to \varphi \Big| \forall v \varphi \Big| \exists v \varphi$$

Axioms

Axioms about the partial ordering of τ

Term axioms resulting e.g. $N \not\sqsubseteq \{\overline{\{N\}^r_B}, N'\}^s_{QQ'}$
expressing what we surely know about the behavior of bit strings: eg. $N_1 \neq N_2 \rightarrow \overline{\langle N_1, t\rangle} \neq N_2$

Axioms about secrecy: further examples:

$$\neg[Key]Sec_\tau(\boldsymbol{A}, \nu, \nu)$$

$$[Key]Sec_\tau(\boldsymbol{A}, \langle t, t'\rangle, \nu) \longrightarrow [Key]Sec_\tau(\boldsymbol{A}, t, \nu)$$

$$[Key]Sec_\tau(\boldsymbol{A}, \langle t, t'\rangle, \nu) \longrightarrow [Key]Sec_\tau(\boldsymbol{A}, \langle t', t\rangle, \nu)$$

$$A \sqsubseteq \boldsymbol{A} \wedge [Key]Sec_\tau(\boldsymbol{A}, t, \nu) \longrightarrow [Key]Sec_\tau(\boldsymbol{A}, \langle t, \{t'\}^r_A\rangle, \nu)$$

$$KeySec(\boldsymbol{A}, \langle t, t'\rangle, K) \wedge K \neq \nu \wedge [Key]Sec_\tau(\boldsymbol{A}, t, \nu) \longrightarrow [Key]Sec_\tau(\boldsymbol{A}, \langle t, \{t'\}^r_K\rangle, \nu)$$

Axioms about secrecy implying authentication

Induction axiom

4. How it works

Roles

E.g. NSL

1. $A \rightarrow B$: $\{N_1, A\}_B$ 2. $B \rightarrow A$: $\{N_1, N_2, B\}_A$ 3. $A \rightarrow B$: $\{N_2\}_B$

With our syntax:

$Init^{A}_{\mathrm{NSL}}[A, i, Q, N_1, n_2, r_1, s_2, r_3] \equiv A\ generates^{i}\ N_1;\ A\ sends^{i}\{N_1, A\}^{r_1}_{Q};$
$A\ receives^{i}\{N_1, n_2, Q\}^{s_2}_{A};\ A\ sends^{i}\{n_2\}^{r_3}_{Q}$

$Resp^{B}_{\mathrm{NSL}}[B, i', Q', n_1, N_2, s_1, r_2, s_3] \equiv B\ receives^{i'}\{n_1, Q'\}^{s_1}_{B};\ B\ generates^{i'}\ N_2;$
$B\ sends^{i'}\{n_1, N_2, B\}^{r_2}_{Q'};\ B\ receives^{i'}\{N_2\}^{s_3}_{B}$

Agreement and Authentication

E.g. NSL from the responder's view:

This is what we want to prove

$$Resp^B_{NSL}[i', A, n_1, N_2, s_1, r_2, s_3] \wedge FOLL(Init^A_{NSL}) \wedge FOLL(Resp^B_{NSL})$$
$$\vdash \exists i r_1 s_2 r_3 Init^A_{NSL}[i, B, n_1, N_2, r_1, s_2, r_3]$$

Where

$$FOLL(Init^A_{NSL}) \equiv \forall i \exists Q N_1 n_2 r_1 s_2 r_3 Foll(Init^A_{NSL}[i, Q, N_1, n_2, r_1, s_2, r_3])$$

Foll is an abbreviation meaning that an initial section of the trace in question was executed (maybe to the end) with the given values.

Proof through secrecy 1

First we show that nonces (or keys) are not corrupted throughout the protocol. That is (reminder):

$$Sec_\tau(\langle A_1, ..., A_n\rangle, N) \wedge \text{Honest } A \; sends^i_\tau t \text{ according to protocol role } \rightarrow Sec_\tau(\langle A_1, ..., A_n\rangle, t, N)$$

Formally:

$$\forall A i t \nu \tau \Big(A \sqsubseteq \boldsymbol{A} \; \wedge \; C[\nu] \; \wedge \; A \; sends^i_\tau t \; \wedge \; [Key]Sec_\tau(\boldsymbol{A}, \nu) \longrightarrow [Key]Sec_\tau(\boldsymbol{A}, t, \nu) \Big)$$

C expresses that ν was generated by the agents in ***A*** and intended for communication among them

Notice that τ is the same for the send action and for the Secrecy predicates.

The formula expresses that ν remains a secret of the agents in A

Does not work.

We need:

$$Sec_\tau(\boldsymbol{A}, u, N) \wedge \; \text{Honest } A \; sends^i_\tau t \text{ following its role } \rightarrow Sec_\tau(\boldsymbol{A}, \langle u, t\rangle, N)$$

Proof through secrecy 2

Again:

$$Sec_\tau(\boldsymbol{A}, u, N) \wedge \text{ Honest } A\ sends^i_\tau t \text{ following its role } \rightarrow Sec_\tau(\boldsymbol{A}, \langle u, t\rangle, N)$$

Formally:

$$[Key]SecSend(\boldsymbol{A}, C, C') \equiv \forall Ait\nu u\tau \Big((A \sqsubseteq \boldsymbol{A} \wedge C[\nu] \wedge C'[u] \wedge \nu \not\sqsubseteq u \wedge A\ sends^i_\tau t)$$
$$\wedge \forall u' \big(C'[u'] \wedge \nu \not\sqsubseteq u' \rightarrow [Key]Sec_\tau(\boldsymbol{A}, u', \nu) \big)$$
$$\longrightarrow [Key]Sec_\tau(\boldsymbol{A}, \langle t, u\rangle, \nu) \Big)$$

In NSL: *A* = ⟨A,B⟩

$$C[N] \equiv \exists irn \big(A\ generates^i N; A\ sends^i \{N, A\}^r_B \vee B\ generates^i N; B\ sends^i \{n, N, B\}^r_A \big)$$

$$C'[u] \equiv \forall t(t \sqsubseteq u \rightarrow \exists m(t = m) \vee \exists t_1 t_2 (t = \langle t_1, t_2 \rangle)$$
$$\wedge \forall m \big(m \sqsubseteq u \rightarrow \exists i (A\ generates^i m \vee B\ generates^i m) \big)$$

C expresses that **N** was generated by **A** or **B**

C' expresses that **u** is a list of nonces generated by **A** or **B**

Proof steps

First show

$$FOLL(Init^{A}_{NSL}) \wedge FOLL(Resp^{B}_{NSL}) \vdash SecSend(\langle A, B\rangle, C, C').$$

For each $A'\ sends^{i}_{\tau}t$ send actions of **A** and **B**, we have to show that $Sec_{\tau}(\langle A, B\rangle, u', N)$ for all u' implies $Sec_{\tau}(\langle A, B\rangle, \langle u, t\rangle, N)$ for all u.

Send actions of **Init**: $t = \{N_1, A\}^{r_1}_{Q} \vee t = \{n_2\}^{r_3}_{Q}$

Send actions of **Resp**: $t = \{n_1, N_2, B\}^{r_2}_{Q'}$

Once SecSend is proven, agreement and auth.

Good news

Works!

We can actually prove protocols

So far: NSL, symmetric NS, Otway-Rees

(by hand, not too difficult)

Bad news

We lied.

The axioms are not quite sound

Correcting soundness

Encryption implying authentication e.g. for public key

$$\{t\}^s_A \sqsubseteq t_2 \ \wedge\ A\ \mathit{receives}^{i_2}_{\tau_2} t_2 \wedge\ \mathit{Sec}_{\tau_2}(\boldsymbol{A}, \{t\}^s_A, \nu) \wedge \neg \mathit{Sec}_{\tau_2}(\boldsymbol{A}, t, \nu)$$
$$\rightarrow \exists A' t_1 t' i_1 r \tau_1 \Big(A' \sqsubseteq \boldsymbol{A} \wedge \{t'\}^r_A \sqsubseteq t_1 \wedge \overline{\{t\}^s_A} = \overline{\{t'\}^r_A} \wedge A'\ \mathit{sends}^{i_1}_{\tau_1} t_1; A\ \mathit{receives}^{i_2}_{\tau_2} t_2\Big)$$

Not sound.

Sound:

$$\Big(\{t\}^s_A \sqsubseteq t_2 \ \wedge\ A\ \mathit{receives}^{i_2}_{\tau_2} t_2 \wedge\ \mathit{Sec}_{\tau_2}(\boldsymbol{A}, \{t\}^s_A, \nu) \wedge \neg \mathit{Sec}_{\tau_2}(\boldsymbol{A}, t, \nu)\Big)\Big|_{\Delta}$$
$$\rightarrow\ \exists A' t_1 t' i_1 r \tau_1 \Delta' \Big(A' \sqsubseteq \boldsymbol{A}\ \wedge\ \Delta' \subseteq \Delta\ \wedge\ \Big(\{t'\}^r_A \sqsubseteq t_1 \wedge\ \overline{\{t\}^s_A} = \overline{\{t'\}^r_A}\ \wedge$$
$$A'\ \mathit{sends}^{i_1}_{\tau_1} t_1; A\ \mathit{receives}^{i_2}_{\tau_2} t_2\Big)\Big|_{\Delta'}\Big)$$

Because of this, all predicates have to be redefined on non-negligible subsets

More good news

About Soundness:

With the correction, soundness holds if

the encryption is CCA-2 (for sym key, we have unforgeable and non unforgeable versions) and

if length of pairing and encryption depend only on the length of the inputs

About Subsets:

For properties that are preserved under restriction to subsets and unions, you don't have to consider sets in the proof. That is, you can use unsound axioms to derive sound result.

Conclusions

We motivated and built a first-order system:

- Computationally sound
  - Relies only on facts that we can surely know
- Simple, but is capable of proving protocols
- Proof works by proving secrecy inductively
- Adjustable
  - E.g. $\overline{\langle n, Q\rangle} \neq N$ can be added if we know that the pairing and nonces are such
  - If we want to allow such a type-flaw attack, we don't include this in the axioms.
  - If we want to allow assigned name attacks (honest names can depend on other things), then we have to remove some axioms.

Further Work

Should be easy: allowing corrupted long-term keys

Dynamic corruption - should not be difficult

Composability conditions