TOWARDS UNCONDITIONAL SOUNDNESS

Gergei Bana and Hubert Comon-Lundh

comon@lsv.ens-cachan.fr
THE CONTEXT

- Can we trust attacks on protocols?
THE CONTEXT

- Can we trust attacks on protocols?
- Can we trust security proofs?
Consider the protocol:

\[
\begin{align*}
A &: \nu N, r, \quad \{A, N\}^r_{pk(B)} \rightarrow \\
&\quad \{B, N\}^r_{pk(A)} \leftarrow \\
B &: \nu r', \quad \rightarrow \{x, y\}^-_{pk(B)} \\
&\quad \leftarrow \{B, y\}^{r'}_{pk(x)}
\end{align*}
\]

security property: \(N\) is a shared secret between \(A\) and \(B\) (when the protocol is completed).
Consider the protocol:

\[
A : \nu N, r, \quad \{A, N\}^r_{pk(B)} \rightarrow \{B, N\}^r_{pk(A)} \leftarrow \\
B : \nu r', \quad \rightarrow \{x, y\}^{-}_{pk(B)} \leftarrow \{B, y\}^{r'}_{pk(x)}
\]

security property: \( N \) is a shared secret between \( A \) and \( B \) (when the protocol is completed).

**True** in the symbolic model
Consider the protocol:

\[
\begin{align*}
A & : \nu N, r, \quad \{A, N\}^{r}_{pk(B)} \rightarrow \{B, N\}^{r}_{pk(A)} \leftarrow \\
B & : \nu r', \quad \rightarrow \{x, y\}^{r'}_{pk(B)} \leftarrow \{B, y\}^{r'}_{pk(x)}
\end{align*}
\]

security property: \( N \) is a shared secret between \( A \) and \( B \) (when the protocol is completed).

**True** in the symbolic model

**False** for some malleable encryption schemes
SECOND STEP: SOUNDNESS RESULTS

**Theorem:** Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.
**SECOND STEP: SOUNDNESS RESULTS**

**Theorem:** Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

**Typical $H$:**

---

06/16/2011
**SECOND STEP: SOUNDNESS RESULTS**

**Theorem:** Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical $H$:

- the encryption scheme is IND-CCA
**SECOND STEP: SOUNDNESS RESULTS**

**Theorem**: Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical $H$:

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
SECOND STEP: SOUNDNESS RESULTS

**Theorem**: Assuming \( H \) then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical \( H \):

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
Theorem: Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical $H$:

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
- There is no dynamic corruption
SECOND STEP: SOUNDNESS RESULTS

**Theorem**: Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical $H$:

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
- There is no dynamic corruption
- There is no “bad key”: keys are certified
**SECOND STEP: SOUNDNESS RESULTS**

**Theorem**: Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical $H$:

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
- There is no dynamic corruption
- There is no “bad key”: keys are certified
- ...

06/16/2011
SECOND STEP: SOUNDNESS RESULTS

**Theorem**: Assuming \( H \) then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical \( H \):

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
- There is no dynamic corruption
- There is no “bad key”: keys are certified
- ...

necessary ?
Theorem: Assuming H then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical H:

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
- There is no dynamic corruption
- There is no “bad key” : keys are certified
- ...

necessary ?
necessary ?
**SECOND STEP: SOUNDNESS RESULTS**

**Theorem**: Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

**Typical $H$:**

- the encryption scheme is IND-CCA  necessary ?
- bitstrings can be efficiently parsed into terms necessary ?
- There is no key-cycle necessary ?
- There is no dynamic corruption necessary ?
- There is no “bad key”: keys are certified
- ...
**SECOND STEP: SOUNDNESS RESULTS**

**Theorem**: Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical $H$:

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
- There is no dynamic corruption
- There is no “bad key”: keys are certified
- ...

necessary ?
**SECOND STEP: SOUNDNESS RESULTS**

**Theorem:** Assuming $H$ then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

Typical $H$:

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
- There is no dynamic corruption
- There is no “bad key” : keys are certified
- ...

**SECOND STEP: SOUNDNESS RESULTS**

**Theorem:** Assuming \( \mathbf{H} \) then any symbolically secure protocol is also computationally secure.

Proof: Hard and (very) long.

**Typical \( \mathbf{H} \):**

- the encryption scheme is IND-CCA
- bitstrings can be efficiently parsed into terms
- There is no key-cycle
- There is no dynamic corruption
- There is no “bad key”: keys are certified
- ...

necessary ?
ALTERNATIVE SOLUTIONS

Formal proofs in a computational model

- CRYPTOVERIF [Bruno Blanchet]
- CERTICRYPT [G. Barthe, S. Zanella et al]
Formal proofs in a computational model

- **CRYPTOVERIF** [Bruno Blanchet]
- **CERTICRYPT** [G. Barthe, S. Zanella et al]

Drawbacks:
**ALTERNATIVE SOLUTIONS**

Formal proofs in a computational model

- **CRYPTOVERIF** [Bruno Blanchet]
- **CERTICRYPT** [G. Barthe, S. Zanella et al]

**Drawbacks:**

- Takes time to develop
ALTERNATIVE SOLUTIONS

Formal proofs in a computational model

- CRYPTOVERIF [Bruno Blanchet]
- CERTICRYPT [G. Barthe, S. Zanella et al]

Drawbacks:

- Takes time to develop
- Minimal assumptions? Small modifications, experiments,...
ALTERNATIVE SOLUTIONS

Formal proofs in a computational model

- CRYPTOVERIF [Bruno Blanchet]
- CERTICRYPT [G. Barthe, S. Zanella et al]

Drawbacks:

- Takes time to develop
- Minimal assumptions? Small modifications, experiments,...
- Full automation?
ALTERNATIVE SOLUTIONS

Formal proofs in a computational model

- CRYPTOVERIF [Bruno Blanchet]
- CERTICRYPT [G. Barthe, S. Zanella et al]

Drawbacks:

- Takes time to develop
- Minimal assumptions ? Small modifications, experiments,...
- Full automation ?
- What if the proof fails ?
Can we trust Soundness theorems?
Can we trust Soundness theorems?

Is the list of assumptions exhaustive?
Can we trust Soundness theorems?

Is the list of assumptions exhaustive?

Why are the soundness proofs so complicated?
The symbolic model specifies **What is allowed**

The computational assumptions specify **What is forbidden**
The symbolic model specifies **What is allowed**

The computational assumptions specify **What is forbidden**

Idea: design a symbolic model that specifies **What is forbidden**
A PERMISSIVE SYMBOLIC MODEL

Anything that is not explicitly forbidden is possible:

A transition is possible as long as the required equalities/deductions are consistent with the current assumptions
A PERMISSIVE SYMBOLIC MODEL

Anything that is not explicitly forbidden is possible:

A transition is possible as long as the required equalities/deductions are consistent with the current assumptions

Advantages:

- All assumptions are necessarily formally stated
- Any model that (also) satisfies the negation of the security assumption is a potential attack
- We may (in principle) use any first-order consistency checker
- Arbitrary primitives, modularity,....
A PERMISSIVE SYMBOLIC MODEL

Anything that is not explicitly forbidden is possible:

A transition is possible as long as the required equalities/deductions are consistent with the current assumptions

Advantages:

- All assumptions are necessarily formally stated
- Any model that (also) satisfies the negation of the security assumption is a potential attack
- We may (in principle) use any first-order consistency checker
- Arbitrary primitives, modularity,....

Difficulties/questions:

- Design (in FO) the appropriate assumptions
- What about the computational attacks ?
- Is automation so easy ?
1. The (symbolic) execution model
2. The main result
3. The computational validity
1. The execution model
Atomic formulas:

- Terms over an arbitrary signature (encryption, pairs and names in the examples) including handles
- Equalities $s = t$ between terms
- Deducibility:

$$\phi, t_1, \ldots, t_n \vdash t$$

where $t_1, \ldots, t_n$ are terms and $\phi$ is interpreted, in any state, as a sequence of ground terms.
- Possibly, Interpreted predicates...

Formulas:
For the transition system: only Boolean combinations of ground atomic formulas.

Interpretation:
Any FO structure.

06/16/2011
THE EXECUTION MODEL: AN EXAMPLE

A : $\nu N, r, \{A, N\}^{r}_{pk(B)} \rightarrow \{B, N\}^{r}_{pk(A)} \leftarrow$

B : $\nu r', \rightarrow \{x, y\}^{r'}_{pk(B)} \leftarrow \{B, y\}^{r'}_{pk(x)}$

Initial state: $q_0, \emptyset, \top$

A successor state: $q_1, \{A, N\}^{r}_{pk(B)}, \top$

A succsucc state: $q_3, \{A, N\}^{r}_{pk(B)}$

$$\{A, N\}^{r}_{pk(B)} \vdash h \land \text{dec}(h, sk(A)) = <B, N>$$
Axioms: Examples

\[ \phi \vdash A, \]
\[ \phi \vdash B, \]
\[ \phi \vdash x, \phi \vdash y \rightarrow \phi \vdash f(x, y), \ldots \]

Anything that the Dolev-Yao attacker can do
AXIOMS: EXAMPLES

\[ \phi \vdash A, \]
\[ \phi \vdash B, \]
\[ \phi \vdash x, \phi \vdash y \rightarrow \phi \vdash f(x, y), \ldots \]

Anything that the Dolev-Yao attacker can do

Secrecy:

\[ \forall x. \phi, \{x\}_{\text{pk}(A)}^{r} \vdash x \rightarrow \phi \vdash x \lor \phi, \{x\}_{\text{pk}(A)}^{r} \vdash \text{sk}(A) \]
Axioms: Examples

\[ \phi \vdash A, \]
\[ \phi \vdash B, \]
\[ \phi \vdash x, \phi \vdash y \rightarrow \phi \vdash f(x, y),... \]

Anything that the Dolev-Yao attacker can do

Secrecy:

\[ \forall x. \: \phi, \{ x \}_rk(A) \vdash x \rightarrow \phi \vdash x \lor \phi, \{ x \}_rk(A) \vdash sk(A) \]

Integrity:

\[ \forall y. \: \phi \vdash y \land \phi, \text{dec}(y, sk(K)) \vdash N \land y \not\in \phi \rightarrow \phi \vdash sk(K) \lor \phi \vdash N \]
**The Execution Model: An Example**

\[ A : \nu N, r, \{ A, N \}^{r}_{pk(B)} \rightarrow \{ B, N \}^{-}_{pk(A)} \quad \leftrightarrow \quad B : \nu r', \rightarrow \{ x, y \}^{-}_{pk(B)} \quad \leftarrow \{ B, y \}^{r'}_{pk(x)} \]

**Initial state:** \( q_0, \emptyset, \top \)

**A successor state:** \( q_1, \{ A, N \}^{r}_{pk(B)}, \top \)

**A succsucc state:** \( q_3, \{ A, N \}^{r}_{pk(B)}, \)

\[ \{ A, N \}^{r}_{pk(B)} \vdash h \quad \land \quad \text{dec}(h, sk(A)) = \langle B, N \rangle \]

This state is now discarded because the formula is inconsistent with the axioms.

The integrity axiom is necessary (otherwise the formula is consistent with the axioms).
2. **THE MAIN RESULT**
**Theorem:** Assume that the axioms are computationally valid. If there is a computational attack, then there is a symbolic attack.

**Note:** this is independent of the security primitives, independent of the properties...
The Computational Soundness Theorem: Assume that the axioms are computationally valid. If there is a computational attack, then there is a symbolic attack.

Note: this is independent of the security primitives, independent of the properties...

Computational validity of axioms, for instance:

Proposition: If the encryption scheme is IND-CCA, then the secrecy and integrity axioms are computationally valid.
3. THE COMPUTATIONAL VALIDITY
THE COMPUTATIONAL INTERPRETATION

- $\mathcal{A}$ is a PPT machine and $\tau$ is a sample (mapping names to bit-strings)
- Each function symbol is interpreted as a deterministic polynomial algorithm.
- For any term $t$, $[t]_\tau$ is the homomorphic extension of $\tau$ to terms
- $\mathcal{A}, \tau \models^c s = t$ iff $[t]_\tau = [s]_\tau$.
- $\mathcal{A}, \tau \models^c t_1, \ldots, t_n \vdash t$ iff $\mathcal{A}([t_1]_\tau, \ldots, [t_n]_\tau) = [t]_\tau$. 

06/16/2011
THE COMPUTATIONAL INTERPRETATION

- A is a PPT machine and \( \tau \) is a sample (mapping names to bit-strings)
- Each function symbol is interpreted as a deterministic polynomial algorithm.
- For any term \( t \), \([t]_\tau\) is the homomorphic extension of \( \tau \) to terms
- \( A, \tau \models^c s = t \iff [t]_\tau = [s]_\tau \).
- \( A, \tau \models^c t_1, \ldots, t_n \vdash t \iff A([t_1]_\tau, \ldots, [t_n]_\tau) = [t]_\tau \).

We wish however to reason on families of first-order structures interpreting the formulas. Otherwise, there is always an \( A \) breaking

\[
\forall x. \phi, \{x\}_{pk(A)}^r \vdash x \quad \rightarrow \quad \phi \vdash x \lor \phi, \{x\}_{pk(A)}^r \vdash sk(A)
\]
THE COMPUTATIONAL INTERPRETATION

- \( \mathcal{A} \) is a PPT machine and \( \tau \) is a sample (mapping names to bit-strings)
- Each function symbol is interpreted as a deterministic polynomial algorithm.
- For any term \( t \), \([t]_\tau\) is the homomorphic extension of \( \tau \) to terms
- \( \mathcal{A}, \tau \models^c s = t \iff [t]_\tau = [s]_\tau \).
- \( \mathcal{A}, \tau \models^c t_1, \ldots, t_n \vdash t \iff \mathcal{A}([t_1]_\tau, \ldots, [t_n]_\tau) = [t]_\tau \).

We wish however to reason on families of first-order structures interpreting the formulas. Otherwise, there is always an \( \mathcal{A} \) breaking

\[
\forall x. \quad \phi, \{x\}^r_{pk(A)} \vdash x \quad \rightarrow \quad \phi \vdash x \quad \lor \quad \phi, \{x\}^r_{pk(A)} \vdash sk(A)
\]

For any \( \tau \), \( \mathcal{A} \) returns

- \([n_1]_\tau\) on input \([n_1]_\tau, [n_2]_\tau, \{n_1\}^r_{pk(A)}]_\tau\)
- \([n_2]_\tau\) on input \([n_1]_\tau, [n_2]_\tau\).
$S, S_1, S_2, \ldots$ are sets of samples
The Computational Interpretation (cntd)

\(S, S_1, S_2, \ldots\) are sets of samples

\(\mathcal{A}, \Pi, S \models^c \exists x. \theta\) if there is a PPT \(\mathcal{A}_x\) such that \(\mathcal{A}, \Pi, S, \mathcal{A}_x \models \theta\).

In what follows: \(\sigma\) is an assignment of PPT machines to the free variables of the formula.
$S, S_1, S_2, \ldots$ are sets of samples

- $\mathcal{A}, \Pi, S \models^c \exists x. \theta$ if there is a PPT $A_x$ such that $\mathcal{A}, \Pi, S, A_x \models \theta$. In what follows: $\sigma$ is an assignment of PPT machines to the free variables of the formula.

- $\mathcal{A}, \Pi, S, \sigma \models^c \theta_1 \wedge \theta_2$ if $\mathcal{A}, \Pi, S, \sigma \models^c \theta_1$ and $\mathcal{A}, \Pi, S, \sigma \models^c \theta_2$
$S, S_1, S_2, \ldots$ are sets of samples

- $\mathcal{A}, \Pi, S \models^c \exists x. \theta$ if there is a PPT $\mathcal{A}_x$ such that $\mathcal{A}, \Pi, S, \mathcal{A}_x \models \theta$.

In what follows: $\sigma$ is an assignment of PPT machines to the free variables of the formula.

- $\mathcal{A}, \Pi, S, \sigma \models^c \theta_1 \land \theta_2$ if $\mathcal{A}, \Pi, S, \sigma \models^c \theta_1$ and $\mathcal{A}, \Pi, S, \sigma \models^c \theta_2$

- $\mathcal{A}, \Pi, S, \sigma \models^c \theta_1 \lor \theta_2$ if $S = S_1 \cup S_2$ and $\mathcal{A}, \Pi, S_1, \sigma \models^c \theta_1$ and $\mathcal{A}, \Pi, S_2, \sigma \models^c \theta_2$
THE COMPUTATIONAL INTERPRETATION (CNTD)

$S, S_1, S_2, \ldots$ are sets of samples

- $\mathcal{A}, \Pi, S \models^c \exists x. \theta$ if there is a PPT $A_x$ such that $\mathcal{A}, \Pi, S, A_x \models \theta$. In what follows: $\sigma$ is an assignment of PPT machines to the free variables of the formula.

- $\mathcal{A}, \Pi, S, \sigma \models^c \theta_1 \land \theta_2$ if $\mathcal{A}, \Pi, S, \sigma \models^c \theta_1$ and $\mathcal{A}, \Pi, S, \sigma \models^c \theta_2$

- $\mathcal{A}, \Pi, S, \sigma \models^c \theta_1 \lor \theta_2$ if $S = S_1 \cup S_2$ and $\mathcal{A}, \Pi, S_1, \sigma \models^c \theta_1$ and $\mathcal{A}, \Pi, S_2, \sigma \models^c \theta_2$

- $\mathcal{A}, \Pi, S, \sigma \models^c \neg \theta$ if $\mathcal{A}, \Pi, S', \sigma \models \theta$ implies that $S'$ is negligible.
THE COMPUTATIONAL INTERPRETATION (CNTD)

\( S, S_1, S_2, \ldots \) are sets of samples

- \( A, \Pi, S \models^c \exists x. \theta \) if there is a PPT \( A_x \) such that \( A, \Pi, S, A_x \models \theta \).

In what follows: \( \sigma \) is an assignment of PPT machines to the free variables of the formula.

- \( A, \Pi, S, \sigma \models^c \theta_1 \land \theta_2 \) if \( A, \Pi, S, \sigma \models^c \theta_1 \) and \( A, \Pi, S, \sigma \models^c \theta_2 \)

- \( A, \Pi, S, \sigma \models^c \theta_1 \lor \theta_2 \) if \( S = S_1 \cup S_2 \) and \( A, \Pi, S_1, \sigma \models^c \theta_1 \) and \( A, \Pi, S_2, \sigma \models^c \theta_2 \)

- \( A, \Pi, S, \sigma \models^c \neg \theta \) if \( A, \Pi, S', \sigma \models \theta \) implies that \( S' \) is negligible.

- \( A, \Pi, S, \sigma \models^c \phi, t_1, \ldots, t_n \vdash t \) if

  For every non negl. \( S' \subseteq S \), there is a non-negl. \( S'' \subseteq S' \) s.t.
  There is a PPT \( A_D \) such that, \( \forall \tau \in S'' \),
  The computation of \( \Pi, A \) yields a bitstring \( b \) s.t.

\[
A_D([\phi]_\tau, [t_1]_\tau^{\sigma(b)}, \ldots, [t_n]_\tau^{\sigma(b)}) = [t]_\tau^{\sigma(b)}
\]
CONCLUSIONS

What remains to do?
What remains to do?

- Automation: simulating the symbolic execution requires a consistency check. We conjecture that, for saturated sets of axioms, this consistency check is in PTIME (ongoing work with Véronique Cortier and Guillaume Scerri).
CONCLUSIONS

What remains to do?

- Automation: simulating the symbolic execution requires a consistency check. We conjecture that, for saturated sets of axioms, this consistency check is in PTIME (ongoing work with Véronique Cortier and Guillaume Scerri).

- Design (and prove the computational validity for classical cryptographic assumptions) axioms for several primitives. Note: this is modular.
What remains to do?

- Automation: simulating the symbolic execution requires a consistency check. We conjecture that, for saturated sets of axioms, this consistency check is in PTIME (ongoing work with Véronique Cortier and Guillaume Scerri).

- Design (and prove the computational validity for classical cryptographic assumptions) axioms for several primitives. Note: this is modular.

- Try several examples of protocols.