On the Symbolic Analysis of Low-Level Cryptographic Primitives: Modular Exponentiation and the Diffie-Hellman Protocol

MARZIA BUSCEMI, University of Pisa
buscemi@di.unipi.it

MICHELE BOREALE, University of Florence
boreale@dsi.unifi.it

Ottawa, FCS'03

Security Protocol Analysis

- Formal methods can be used for the verification of security protocols: agents executing the protocols are modelled as concurrent processes, described by means of an appropriate language, i.e., the spi-calculus.

- Model checking and reachability analysis rely on exploration of the space of this system. The sequences of actions (traces) are analysed to check whether they lead to an insecure state.

The Attacker Model

Formal methods for protocol analysis usually follow the Dolev-Yao attacker model: the (hostile) environment can manipulate any message travelling over the network, synthesize new messages by pairing, encryption, and decryption of previously known messages, etc.

State Explosion Problem: protocols typically generate an infinite number of traces, as an agent may expect any of the infinitely many messages the environment can send on the network at any time.

Symbolic Analysis

- Replace the infinitely many transitions arising from an input action by a single symbolic transition;
- Represent the received message as a variable;
- Add constraints on this variable as the computation proceeds.

Bibliography:

- [Amadio, Lugiez00], [Boreale01], [Millen, Shmatikov01], etc.

Example

Let $P = \mathsf{a}(x).\,\mathsf{let}\,y = \mathsf{dec}_k(x)\,\mathsf{in}\,P'$.

$$\begin{aligned} \langle \varepsilon,\ P\rangle &\longrightarrow \langle \mathsf{a}\langle x\rangle,\ \mathsf{let}\,y = \mathsf{dec}_k(x)\,\mathsf{in}\,P'\rangle \\ &\longrightarrow \langle \mathsf{a}\langle \{x'\}_k\rangle,\ P'[\{x'\}_k/x][x'/y]\rangle \end{aligned}$$

Note: The decryption of x using k is resolved by unifying x and $\{x'\}_k$ for some fresh x' (mgu $\theta = [\{x'\}_k/x]$); y is replaced by the result of the decryption, i.e., x'.

Problem

The Dolev-Yao model assumes black-box enciphering and perfect cryptography. In some cases, this is an over-simplification.

For instance, the analysis of a protocol based on modular exponentiation should take into account the operations involved in the protocol (exponentiation, product) and their ‘relevant’ algebraic laws; even operations not explicitly mentioned, provided they are considered feasible, must be accounted for, since an adversary might take advantage of them.

Diffie-Hellman Protocol

The protocol has two public parameters: a large prime p and a generator α for $\mathcal{Z}_p^*$. A and B generates n_A and B generates $n_B \in \mathcal{Z}_p^*$. Next, A and B exchange their public values:

$$1. \quad A \longrightarrow B: \quad \exp(\alpha, n_A)$$
$$2. \quad B \longrightarrow A: \quad \exp(\alpha, n_B).$$

Finally, A computes $K = \exp(\exp(\alpha, n_B), n_A) = \exp(\alpha, n_A \times n_B)$, and B computes $K = \exp(\exp(\alpha, n_A), n_B) = \exp(\alpha, n_A \times n_B)$.

$$3. \quad A \longrightarrow B: \quad \{d\}_K.$$

Contribution

We extend a symbolic method for automatic analysis of security protocols to handle modular exponentiation and, in particular, the Diffie-Hellman key exchange.

Related Work. Recently, some protocol analysis techniques ([MS03, CLS03, Her02]) have been developed which take into account an attacker model with some equational theory.

Our model is less precise than others (notably [MS03]), its main limitation being a bound on the number of factors that may appear in any exponent. However, for such restricted model, we offer a decision method.

Symbolic Framework

Our approach relies on a symbolic framework for protocol analysis in presence of a variety of crypto-functions and low-level operations.

Main Idea

- To formalise a protocol by means of a language akin to the applied pi.
- To give a symbolic operational semantics based on unification and such that provides finite and effective protocol models.
- To provide a method to carry out trace analysis directly on the symbolic model. Under certain conditions on the primitives the symbolic method is proven complete.

General Framework

- For Σ a signature of function symbols $f, g, \ldots$, T_Σ is the algebra of terms given by the grammar:

$$\zeta, \eta \quad ::= \quad u \mid f(\widetilde{\zeta})$$

- A frame $\mathcal{F}$ is a triple $(\Sigma, \mathcal{M}, \downarrow)$, where:
  - Σ is a signature;
  - $\mathcal{M} \subseteq \mathcal{E}_\Sigma$ is a set of messages $M, N, \ldots$;
  - $\downarrow \subseteq \mathcal{E}_\Sigma \times \mathcal{M}$ is an evaluation relation.

- Given an initial set of messages S, the set $\mathcal{H}(S)$ consists of all the expressions inductively built by applying functions of Σ to elements of S. A message M is deducible from S, $S \vdash M$, if and only if $\exists \zeta \in \mathcal{H}(S) : \zeta \downarrow M$.

A Frame for Diffie-Hellman Protocol (1/3)

SIGNATURE $\quad \Sigma = \{\, \alpha,\ \mathsf{unit},\ 1,\ \{\cdot\}_{(\cdot)},\ \mathsf{dec}_{(\cdot)}(\cdot),\ \mathsf{exp}(\cdot,\cdot),\ \mathsf{root}(\cdot,\cdot),$

$\cdot \times \cdot,\ \mathsf{mult}(\cdot,\cdot),\ \mathsf{inv}(\cdot),\ \mathsf{inv}'(\cdot),\ (\cdot)^{-1} \,\}$

FACTORS $\quad f ::= u \mid u^{-1}$

PRODUCTS $\quad F ::= 1 \mid f_1 \times \cdots \times f_k$

KEYS $\quad K, H ::= f \mid \mathsf{exp}(\alpha, F)$

MESSAGES $\quad M, N ::= F \mid K \mid \{M\}_K$

A Frame for Diffie-Hellman Protocol (2/3)

$$(\textsc{Mult}) \quad \mathsf{mult}(\zeta_1 \times \cdots \zeta_k, \zeta_{k+1} \times \cdots \times \zeta_n) \rightsquigarrow \zeta_{i_1} \times \cdots \times \zeta_{i_n} \qquad 1 \leq k < n \leq l$$

$$(\textsc{Inv}_1) \quad \mathsf{inv}(\zeta_1 \times \cdots \times \zeta_n) \rightsquigarrow \mathsf{inv}'(\zeta_1) \times \cdots \times \mathsf{inv}'(\zeta_n) \qquad n \leq l$$

$$(\textsc{Inv}_2) \quad \mathsf{inv}'(\zeta^{-1}) \rightsquigarrow \zeta$$

$$(\textsc{Inv}_3) \quad \mathsf{inv}'(\zeta) \rightsquigarrow \zeta^{-1}$$

$$(\textsc{Inv}_4) \quad \mathsf{inv}'(\zeta) \times \zeta \rightsquigarrow \mathsf{unit}$$

$$(\textsc{Unit}_1) \quad \mathsf{unit} \times \zeta \rightsquigarrow \zeta$$

$$(\textsc{Unit}_2) \quad \mathsf{unit} \rightsquigarrow 1$$

A Frame for Diffie-Hellman Protocol (3/3)

(DEC) $\mathsf{dec}_\eta(\{\zeta\}_\eta) \rightsquigarrow \zeta$

(EXP) $\mathsf{exp}(\mathsf{exp}(\xi,\eta),\zeta) \rightsquigarrow \mathsf{exp}(\xi,\mathsf{mult}(\eta,\zeta))$

(ROOT) $\mathsf{root}(\mathsf{exp}(\xi,\eta),\zeta) \rightsquigarrow \mathsf{exp}(\xi,\mathsf{mult}(\eta,\mathsf{inv}(\zeta)))$

(CTX) $$\frac{\zeta \rightsquigarrow \zeta'}{C[\zeta] \rightsquigarrow C[\zeta']}$$

EVALUATION $\zeta \downarrow \eta$ iff $\zeta \rightsquigarrow^* \eta$

Discussion

The relation $\rightsquigarrow$ is terminating but not confluent. In fact, the non-determinism of $\rightsquigarrow$ is intended to model the commutativity and the associativity of the product operation, as reflected in the rule (MULT).

Model Restrictions:

1. a fixed upper bound (l) on the number of factors;
2. product and inverse operations cannot be applied to exponentials and to encrypted terms;
3. exponentiation starts from the basis α, and exponents can only be products.

Note. Starting from a term obeying the above conditions, an attacker is capable of 'deducing' all AC variants of the message represented by that term.

Process Syntax

$$
\begin{array}{lcll}
A, B & ::= & \mathbf{0} & \text{(null)} \\
 & | & \mathsf{a}(x).A & \text{(input)} \\
 & | & \overline{\mathsf{a}}\langle\zeta\rangle.A & \text{(output)} \\
 & | & \mathsf{let}\, y = \zeta\, \mathsf{in}\, A & \text{(evaluation)} \\
 & | & [\zeta = \eta]A & \text{(matching)} \\
 & | & A \parallel B & \text{(parallel composition)} \\
 & | & (\mathsf{new}\, a)A & \text{(restriction)}
\end{array}
$$

Diffie-Hellman Specification

$$
\begin{aligned}
A &= (\mathsf{new}\, n_A)\, \overline{\mathsf{a1}}\langle \exp(\alpha, n_A)\rangle.\, \mathsf{a2}(x).\, \mathsf{let}\, z = \exp(x, n_A)\, \mathsf{in}\, \overline{\mathsf{a3}}\langle \{d\}_z\rangle.\, 0 \\
B &= (\mathsf{new}\, n_B)\, \mathsf{b1}(y).\, \overline{\mathsf{b2}}\langle \exp(\alpha, n_B)\rangle.\, \mathsf{let}\, w = \exp(y, n_B)\, \mathsf{in}\, \mathsf{b3}(t). \\
&\quad\ \mathsf{let}\, t' = \mathsf{dec}_w(t)\, \mathsf{in}\, B' \\
P &= A \,||\, B.
\end{aligned}
$$

Concrete Operational Semantics

- An action is a term of the form $\mathsf{a}\langle M\rangle$ (input action) or $\overline{\mathsf{a}}\langle M\rangle$ (output action).
- A trace is a sequence of actions such that any input message can be synthesized from the knowledge the environment has previously acquired. Formally, it is a closed string s such that

$$\forall s_1, s_2, \mathsf{a}\langle M\rangle, \quad \text{if } s = s_1 \cdot \mathsf{a}\langle M\rangle \cdot s_2 \text{ then } s_1 \vdash M.$$

- A configuration $\langle s, P\rangle$ is a pair consisting of a trace s and a process P.

Note: $s \vdash M$ abbreviates $\mathrm{msg}(s) \vdash M$.

Transition System

Basic reduction rules:

$$(\textsc{Inp}) \quad \langle s,\ \mathsf{a}(x).P\rangle \longrightarrow \langle s\cdot \mathsf{a}\langle M\rangle,\ P[{}^{M}/_{x}]\rangle \qquad s \vdash M,\ M \text{ closed}$$

$$(\textsc{Out}) \quad \langle s,\ \overline{\mathsf{a}}\langle\zeta\rangle.P\rangle \longrightarrow \langle s\cdot \overline{\mathsf{a}}\langle M\rangle,\ P\rangle \qquad \zeta \downarrow M,\ M \text{ closed}$$

Note: Rule (Inp) makes the transition relation infinite-branching, since M ranges over the infinite set of deducible messages.

Transition System (Cont.)

Other rules:

$$(\textsc{Let}) \quad \langle s,\ \mathsf{let}\, y = \zeta\, \mathsf{in}\, P\rangle \longrightarrow \langle s,\ P[^{M}/_{y}]\rangle \qquad \zeta \downarrow M,\ M \text{ closed}$$

$$(\textsc{Match}) \quad \langle s,\ [\zeta = \eta]P\rangle \longrightarrow \langle s,\ P\rangle \qquad \zeta \downarrow M, \eta \downarrow N, M = N$$

$$(\textsc{Par}) \quad \frac{\langle s,\ P\rangle \longrightarrow \langle s',\ P'\rangle}{\langle s,\ P \parallel Q\rangle \longrightarrow \langle s',\ P' \parallel Q\rangle}$$

Security Properties: Specification

Properties are expressed in terms of correspondence assertions:

- $\alpha \hookleftarrow \beta$ in s if action α always occurs prior to action β in s;
- $s \models \alpha \hookleftarrow \beta$ if, for each ground substitution ρ, $\alpha\rho \hookleftarrow \beta\rho$ in s.
- $\mathcal{C} \models \alpha \hookleftarrow \beta$ if, for all traces s generated by $\mathcal{C}$, $s \models \alpha \hookleftarrow \beta$.

Correspondence assertions are suited for security properties, such as secrecy and authentication.

An Example: Secrecy

The property that the protocol P in the above example should not leak the datum d can be expressed also by saying that the adversary will never be capable of synthesising d, without prior knowledge of it.

This can be formalised by considering an 'absurd' action $\perp$ (that is nowhere used in agent expression) and by extending P with a 'guardian' process: $\mathrm{g}(x).\mathbf{0}$. The secrecy property to check is then:

$$\langle \varepsilon,\ P \parallel \mathrm{g}(x).\mathbf{0} \rangle \models \perp \hookleftarrow \mathrm{g}\langle d \rangle$$

Symbolic Semantics

We equip the frame for modular exponentiation with a symbolic evaluation relation ($\downarrow_s$), which is in agreement with its concrete counterpart ($\downarrow$). Intuitively, $\zeta \downarrow_\theta \eta$ means that ζ evaluates to η under all instances ρ of θ. The relation $\downarrow_s$ is presented in the next slide.

A symbolic trace σ is a trace that may contain variables.

A symbolic configuration $\langle \sigma, A \rangle_s$ is a pair composed by a symbolic trace σ and an agent A.

Symbolic Evaluation Relation (1/2)

$$(\textsc{Mult}_s) \quad \mathsf{mult}(\zeta_1, \zeta_n) \overset{\theta}{\rightsquigarrow}_s (x_{i_1} \times \cdots \times x_{i_n})\,\theta \qquad \begin{cases} 1 \le k < n \le l, \\ \theta = \mathrm{mgu}(\zeta_1 = x_1 \times \cdots \times x_k, \\ \zeta_2 = x_{k+1} \times \cdots \times x_n) \end{cases}$$

$$(\textsc{Inv1}_s) \quad \mathsf{inv}(\zeta) \overset{\theta}{\rightsquigarrow}_s (\mathsf{inv}'(x_{i_1}) \times \cdots \times \mathsf{inv}'(x_{i_n}))\,\theta \qquad \begin{cases} 1 \le n \le l, \\ \theta = \mathrm{mgu}(\zeta = x_1 \times \cdots \times x_n) \end{cases}$$

$$(\textsc{Inv2}_s) \quad \mathsf{inv}'(\zeta) \overset{\theta}{\rightsquigarrow}_s x_1\,\theta \qquad \theta = \mathrm{mgu}(\zeta, {x_1}^{-1})$$

$$(\textsc{Inv3}_s) \quad \mathsf{inv}'(\zeta) \overset{\varepsilon}{\rightsquigarrow}_s \zeta^{-1}$$

$$(\textsc{Inv4}_s) \quad \mathsf{inv}'(\zeta) \times \eta \overset{\theta}{\rightsquigarrow}_s \mathsf{unit} \qquad \theta = \mathrm{mgu}(\zeta, \eta)$$

$$(\textsc{Unit1}_s) \quad \mathsf{unit} \times \zeta \overset{\varepsilon}{\rightsquigarrow}_s \zeta$$

$$(\textsc{Unit2}_s) \quad \mathsf{unit} \overset{\varepsilon}{\rightsquigarrow}_s 1$$

Symbolic Evaluation Relation (2/2)

$$(\textsc{Dec}_s)\quad \mathsf{dec}_\eta(\zeta) \overset{\theta}{\rightsquigarrow}_s x_1\,\theta \qquad \theta = \mathrm{mgu}(\zeta = \{x_1\}_{x_2}, \eta = x_2)$$

$$(\textsc{Exp1}_s)\quad \mathsf{exp}(x,\zeta) \overset{\theta}{\rightsquigarrow}_s \mathsf{exp}(\alpha, \mathsf{mult}(x_1,\zeta)) \qquad \theta = [\mathsf{exp}(\alpha,x_1)/x]$$

$$(\textsc{Exp2}_s)\quad \mathsf{exp}(\mathsf{exp}(\xi,\eta),\zeta) \overset{\varepsilon}{\rightsquigarrow}_s \mathsf{exp}(\xi, \mathsf{mult}(\eta,\zeta))$$

$$(\textsc{Root1}_s)\quad \mathsf{root}(x,\zeta) \overset{\theta}{\rightsquigarrow}_s \mathsf{exp}(\alpha, \mathsf{mult}(x_1, \mathsf{inv}(\zeta))) \qquad \theta = [\mathsf{exp}(\alpha,x_1)/x]$$

$$(\textsc{Root2}_s)\quad \mathsf{root}(\mathsf{exp}(\xi,\eta),\zeta) \overset{\varepsilon}{\rightsquigarrow}_s \mathsf{exp}(\xi, \mathsf{mult}(\eta, \mathsf{inv}(\zeta)))$$

$$(\textsc{Ctx}_s)\quad \frac{\zeta \overset{\theta}{\rightsquigarrow}_s \zeta'}{C[\zeta] \overset{\theta}{\rightsquigarrow}_s C\theta[\zeta']}$$

$$\textsc{Symbolic Evaluation} \qquad \zeta \downarrow_\theta \eta \quad \text{iff} \quad \zeta \overset{\theta_1}{\rightsquigarrow}_s \cdots \overset{\theta_n}{\rightsquigarrow}_s \eta \quad \text{and} \quad \theta = \theta_1 \cdots \theta_n$$

Symbolic Transition System

$$(\textsc{Inp}_s) \quad \langle \sigma,\ a(x).A\rangle_s \longrightarrow_s \langle \sigma \cdot a\langle x\rangle,\ A\rangle_s$$

$$(\textsc{Out}_s) \quad \langle \sigma,\ \overline{a}\langle\zeta\rangle.A\rangle_s \longrightarrow_s \langle \sigma\theta \cdot \overline{a}\langle M\rangle,\ A\theta\rangle_s \qquad \zeta \downarrow_\theta M$$

$$(\textsc{Let}_s) \quad \langle \sigma,\ \mathsf{let}\, y = \zeta\, \mathsf{in}\, A\rangle_s \longrightarrow_s \langle \sigma\theta,\ A\theta[{}^{M}/_{y}]\rangle_s \qquad \zeta \downarrow_\theta M$$

$$(\textsc{Match}_s) \quad \langle \sigma,\ [\zeta = \eta]A\rangle_s \longrightarrow_s \langle \sigma\theta,\ A\theta\rangle_s \qquad \zeta \downarrow_{\theta_1} M,\ \eta\,\theta_1 \downarrow_{\theta_2} N,$$

$\theta_3 = \mathrm{mgu}(M\theta_2, N)$,

$N\theta_3 \in \mathcal{M}$, $\theta = \theta_1\theta_2\theta_3$

$$(\textsc{Par}_s) \quad \frac{\langle \sigma,\ A\rangle_s \longrightarrow_s \langle \sigma',\ A'\rangle_s}{\langle \sigma,\ A \parallel B\rangle_s \longrightarrow_s \langle \sigma',\ A' \parallel B'\rangle_s}$$

(side condition on B')

Symbolic Semantics: Example

Let $P = \overline{\mathsf{a}}\langle k\rangle.\,\mathsf{a}(x).\,\mathsf{let}\,z = \mathsf{root}\,(x,k)\,\mathsf{in}\,P'$. Then,

$$\langle \varepsilon,\, P\rangle_{\mathrm{s}} \longrightarrow_{\mathrm{s}} \longrightarrow_{\mathrm{s}} \longrightarrow_{\mathrm{s}} \langle (\overline{\mathsf{a}}\langle k\rangle \cdot \mathsf{a}\langle x\rangle)\,\theta,\, P'\theta\,\theta'\rangle_{\mathrm{s}},$$

with $\theta = [^{\mathsf{exp}\,(\alpha, x_1)}/_{x}]$ (x_1 fresh) and $\theta' = [^{\mathsf{exp}\,(\alpha, x_1 \times k^{-1})}/_{z}]$.

Symbolic vs. Concrete Semantics

Theorem (soundness and completeness). Let $\mathcal{C}$ be an initial configuration and s a trace. Then $\mathcal{C}$ generates s if and only if there is σ s.t. $\mathcal{C}$ symbolically generates σ and s is an instance of σ.

By this result, the task of checking correspondence assertions is reduced to the analysis of the finitely many symbolic traces the initial configuration can generate.

Trace Consistency

Many symbolic traces are inconsistent, i.e., they cannot be instantiated to any concrete trace. For example, $\mathsf{a}\langle\{x\}_k\rangle \cdot \overline{\mathsf{a}}\langle x\rangle$ is inconsistent.

Proposition Let σ be a symbolic trace. Then there exists a finite set of traces $\mathbf{Refinement}(\sigma)$, which are instances of σ and have the following property: for any s, s is a solution of σ if and only if s is a solution of some $\sigma' \in \mathbf{Refinement}(\sigma)$.

Roughly, the set $\mathbf{Refinement}(\sigma)$ is computed by repeatedly unifying each input message in σ to terms that can be synthetized out of previous messages in σ.

Refinement

Definition (Refinement) *Refinement* ($\succ$) is the least binary relation over symbolic traces given by the following rules, assuming that (1) σ' is the longest prefix of σ which is a trace, (2) $\sigma = \sigma' \cdot \mathsf{a}\langle M \rangle \cdot \sigma''$, for some σ'', (3) $N, N' \notin \mathcal{V} \cup \widehat{\mathcal{V}}$, (4) $\theta \neq \varepsilon$.

$$(\textsc{Ref}_1)\ \frac{N' \in \mathbf{b}(\sigma') \quad M = C[N] \quad \theta = \mathrm{mgu}(N, N')}{\sigma \ \succ\ \sigma\theta\theta_0}$$

$$(\textsc{Ref}_2)\ \frac{x \in \mathrm{v}(M)}{\sigma \ \succ\ \sigma[\hat{x}/x]}$$

where $\theta_0 = [x/\hat{x} \mid \hat{x} \in \mathrm{v}(\sigma)$ and $|(\sigma\theta) \setminus \hat{x}| < |\sigma \setminus \hat{x}|\,]$.

For any symbolic trace σ,

$$\mathbf{Refinement}(\sigma) = \{\, \sigma' \mid \sigma \succ {}^{*}\sigma' \text{ and } \sigma' \text{ is a trace} \,\}.$$

Verification Method

$\underline{\mathbf{M}(\mathcal{C}, \alpha \hookleftarrow \beta)}$

1. compute $\mathbf{Mod}_{\mathcal{C}} = \{\sigma \mid \mathcal{C} \searrow_{s} \sigma\}$;
2. **foreach** $\sigma \in \mathbf{Mod}_{\mathcal{C}}$ **do**
3. **foreach** action γ in σ **do**
4. **if** $\exists \theta = \mathrm{mgu}(\beta, \gamma)$ **and** $\exists \sigma' = (\sigma\theta\theta') \in \mathbf{SF}(\sigma\theta)$ s.t.
5. $\alpha\theta\theta'$ does not occur prior to $\beta\theta\theta'$ in σ'
6. **then return**(No, σ');
7. **return**(Yes);

Theorem.

- If $\mathbf{M}(\mathcal{C}, \alpha \hookleftarrow \beta)$ returns Yes then $\mathcal{C} \models \alpha \hookleftarrow \beta$.
- If $\mathbf{M}(\mathcal{C}, \alpha \hookleftarrow \beta)$ returns (No, σ) then $\mathcal{C} \not\models \alpha \hookleftarrow \beta$.

Diffie-Hellman Protocol Analysis

1. The symbolic model $\mathbf{Mod}_{C_{DH}}$ is computed (in practice, symbolic traces are generated and checked 'on-the-fly').

2. The following symbolic trace σ is considered:

$$s = \overline{\mathsf{a1}}\langle \mathsf{exp}(\alpha, n_A)\rangle \cdot \mathsf{a2}\langle \mathsf{exp}(\alpha, x_0)\rangle \cdot \overline{\mathsf{a3}}\langle \{d\}_K\rangle \cdot \mathsf{g}\langle t\rangle$$

3,4. Action $\gamma = \mathsf{g}\langle t\rangle$ is found that unifies with $\beta = \mathsf{g}\langle d\rangle$, via $\theta = [^{d}/_{t}]$.

5. The set $\mathbf{Refinement}(\sigma\theta) = \{\sigma'\}$ is computed where $\sigma' = \sigma\theta\theta'$, and $\theta' = [^{\hat{x}_0}/_{x_0}]$. By the above proposition, σ' is a consistent trace.

6. Action $\perp$ does not appear in σ', hence,

7. (No, σ') is returned.

An attack to the protocol, i.e. a ground trace s such that the $\langle \varepsilon, S\rangle \searrow s$ and $s \not\models \mathit{Secret}(d) = \perp \hookleftarrow \mathsf{g}\langle d\rangle$, is then retrieved as $s = \sigma[^{\underline{n}_I}/_{\hat{x}_0}]$.

Conclusion and Future Work

We have presented a model for the analysis of protocols built around shared-key encryption and modular exponentiation. The model is less precise than others. However, for such restricted model, we offer a decision method.

We believe the method is effective in practice, since the symbolic model is compact and the refinement procedure is invoked only on demand. We are in process of integrating our technique into the STA analysis tool.

`http://www.dsi.unifi.it/~boreale/tool.html`

Our technical development has been confined to multiplication and exponentiation, but we are confident that the approach can be extended to other low-level cryptographic primitives (RSA encryption).