On the Decidability of Cryptographic Protocols with Open-ended Data Structures

Ralf Küsters

Institute of Computer Science and Applied Mathematics
University of Kiel, Germany

kuesters@ti.informatik.uni-kiel.de

Overview

1. Closed-ended protocols
2. Open-ended protocols
3. A generic protocol model
4. A classical instance of this model: the rewriting model
5. New instances designed to capture open-ended data structures:
   (a) A generalization of the rewriting model: undecidable
   (b) The transducer-based model: decidable
6. Conclusion and future work

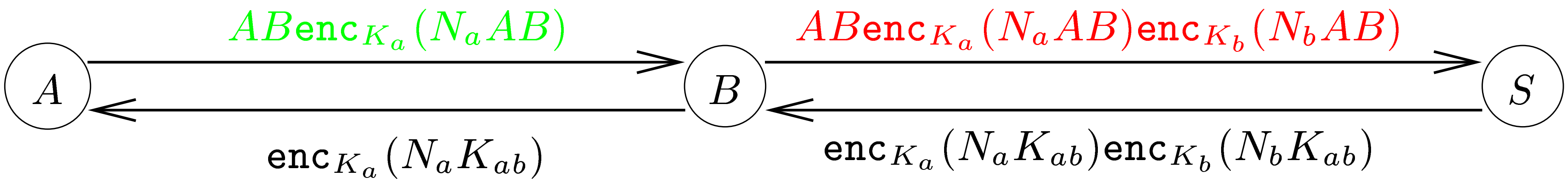
Closed-ended Protocols

$\boxed{\text{Example:}}$ Otway-Rees

$$\begin{tikzcd}[column sep=huge] A \arrow[r, shift left, "AB\mathtt{enc}_{K_a}(N_a AB)"] & B \arrow[l, shift left, "\mathtt{enc}_{K_a}(N_a K_{ab})"] \arrow[r, shift left, "AB\mathtt{enc}_{K_a}(N_a AB)\mathtt{enc}_{K_b}(N_b AB)"] & S \arrow[l, shift left, "\mathtt{enc}_{K_a}(N_a K_{ab})\mathtt{enc}_{K_b}(N_b K_{ab})"] \end{tikzcd}$$

Closed-ended Protocols

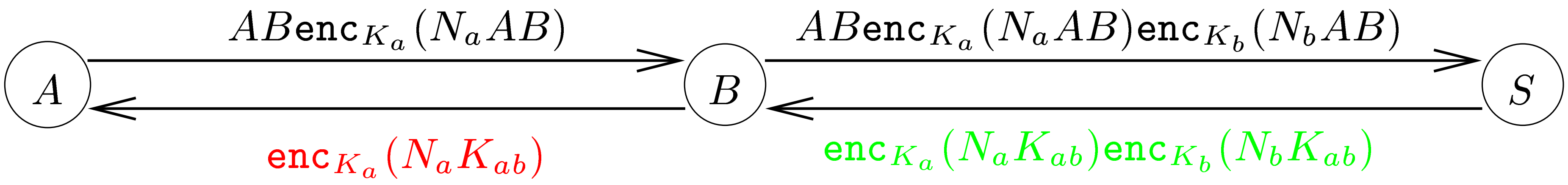
Example: Otway-Rees



receive-send action

Closed-ended Protocols

Example: Otway-Rees



receive-send action

Closed-ended Protocols

Example: Otway-Rees

$$\begin{tikzcd}[column sep=huge]
A \arrow[r, shift left, "AB\mathtt{enc}_{K_a}(N_a AB)"] & B \arrow[l, shift left, "\mathtt{enc}_{K_a}(N_a K_{ab})"] \arrow[r, shift left, "AB\mathtt{enc}_{K_a}(N_a AB)\mathtt{enc}_{K_b}(N_b AB)"] & S \arrow[l, shift left, "\mathtt{enc}_{K_a}(N_a K_{ab})\mathtt{enc}_{K_b}(N_b K_{ab})"]
\end{tikzcd}$$

In one receive-send action only a fixed number of data fields have to be processed.

Closed-ended Protocols

Example: Otway-Rees

$$\begin{tikzcd}[column sep=huge]
A \arrow[r, shift left, "AB\mathtt{enc}_{K_a}(N_aAB)"] & B \arrow[l, shift left, "\mathtt{enc}_{K_a}(N_aK_{ab})"] \arrow[r, shift left, "AB\mathtt{enc}_{K_a}(N_aAB)\mathtt{enc}_{K_b}(N_bAB)"] & S \arrow[l, shift left, "\mathtt{enc}_{K_a}(N_aK_{ab})\mathtt{enc}_{K_b}(N_bK_{ab})"]
\end{tikzcd}$$

In one receive-send action only a fixed number of data fields have to be processed.

Open-ended Protocols

The number of data fields processed in one receive-send action is unbounded.

Example IKE Protocol

$$\xrightarrow{\quad \mathsf{SA}_1\ \mathsf{SA}_2\ \cdots\ \mathsf{SA}_n \quad} A \xrightarrow{\quad \mathsf{SA}_i \quad}$$

Open-ended Protocols

The number of data fields processed in one receive-send action is unbounded.

$\boxed{\text{Example}}$ IKE Protocol

$$\xrightarrow{\quad \mathsf{SA}_1\ \mathsf{SA}_2\ \cdots\ \mathsf{SA}_n \quad} \; A \; \xrightarrow{\quad \mathsf{SA}_i \quad}$$

A needs to perform in one receive-send action
an unbounded number of internal actions: **Iteration**.

Open-ended Protocols

The number of data fields processed in one receive-send action is unbounded.

$\boxed{\text{Example}}$ Recursive Authentication Protocol [BullOtway '97] (a group protocol)

$$\downarrow \ \mathtt{hash}(\cdots \mathtt{hash}(K_c C D N_c\ \mathtt{hash}(K_b B C N_b\ \mathtt{hash}(K_a A B N_a -)))\cdots)$$

S

$$\downarrow \ \cdots \mathtt{enc}_{K_c}(K_{cd} D N_c)\mathtt{enc}_{K_c}(K_{bc} B N_c)\mathtt{enc}_{K_b}(K_{bc} C N_b)\mathtt{enc}_{K_b}(K_{ab} A N_b)\mathtt{enc}_{K_a}(K_{ab} B N_a)$$

Open-ended Protocols

The number of data fields processed in one receive-send action is unbounded.

Example Recursive Authentication Protocol [BullOtway '97]
(a group protocol)

$$\downarrow \; \mathtt{hash}(\cdots \mathtt{hash}(K_c C D N_c \; \mathtt{hash}(K_b B C N_b \; \mathtt{hash}(K_a A B N_a -)))\cdots)$$

$$S$$

$$\downarrow \; \cdots \mathtt{enc}_{K_c}(K_{cd} D N_c)\mathtt{enc}_{K_c}(K_{bc} B N_c)\mathtt{enc}_{K_b}(K_{bc} C N_b)\mathtt{enc}_{K_b}(K_{ab} A N_b)\mathtt{enc}_{K_a}(K_{ab} B N_a)$$

S needs to perform in one receive-send action
an unbounded number of internal actions: **Iteration**.

Formal Analysis of Open-ended Protocols

There exist only a few contributions so far.

$\boxed{\text{The recursive authentication protocol}}$ has been analyzed by

- Paulson (Isabelle theorem prover), 1997, and
- Bryans and Schneider (PVS theorem prover), 1997.

$\boxed{\text{The A-GDH.2 protocol}}$ has been analyzed by

- Meadows (NRL Analyzer), 2000, and
- Pereira and Quisquater (manually, based on a modal similar to the strand spaces model), 2001.

Formal Analysis of Open-ended Protocols

There exist only a few contributions so far.

The recursive authentication protocol has been analyzed by

- Paulson (Isabelle theorem prover), 1997, and
- Bryans and Schneider (PVS theorem prover), 1997.

The A-GDH.2 protocol has been analyzed by

- Meadows (NRL Analyzer), 2000, and
- Pereira and Quisquater (manually, based on a modal similar to the strand spaces model), 2001.

Decidability issues have not been studied so far.

A Generic Protocol Model

Similar to other protocol models [Cervesato et al., Rusinowitch et al., Amadio et al., Huima, ...] except that

receive-send actions may be arbitrary computations.

Assumptions

- Principals and the intruder cannot generate nonces.
- Keys are atomic.
- The number of sessions is bounded.
- The intruder is the standard Dolev-Yao intruder; messages of arbitrary size can be generated.

A Generic Protocol Model: Messages

Consider messages as terms.

Finite set of atomic messages $\mathcal{N}$: keys, prinicipal names, nonces, ...
secret $\in \mathcal{N}$

Set of messages over $\mathcal{N}$: least set $\mathcal{M}$ with

- atomic messages: $\mathcal{N} \subseteq \mathcal{M}$;
- concatenation: if $m, m' \in \mathcal{M}$, then $mm' \in \mathcal{M}$;
- encryption: if $m \in \mathcal{M}$ and $a \in \mathcal{N}$, then $\mathtt{enc}_a(m) \in \mathcal{M}$ (atomic keys);
- hashing: if $m \in \mathcal{M}$, then $\mathtt{hash}(m) \in \mathcal{M}$.

$\mathcal{M}_\varepsilon := \mathcal{M} \cup \{\varepsilon\}$.

A Generic Protocol Model: Principals and Protocols

A generic principal Π: (Q, I, l, α)

- (possibly infinite) set of states: Q,
- set of initial states: $I \subseteq Q$,
- number of actions to be performed by Π: l,
- receive-send actions: α is a mapping assigning to every $j < l$, a relation

$$\alpha(j) \subseteq Q \times \mathcal{M}_\varepsilon \times \mathcal{M}_\varepsilon \times Q.$$

A generic protocol P: $(n, \{\Pi_i\}_{i<n}, \mathcal{K})$

- number of principals: n,
- family of n principals: $\{\Pi_i\}_{i<n}$, and
- finite initial intruder knowledge: $\mathcal{K} \subseteq \mathcal{M}_\varepsilon$.

A Generic Protocol Model: The Dolev-Yao Intruder

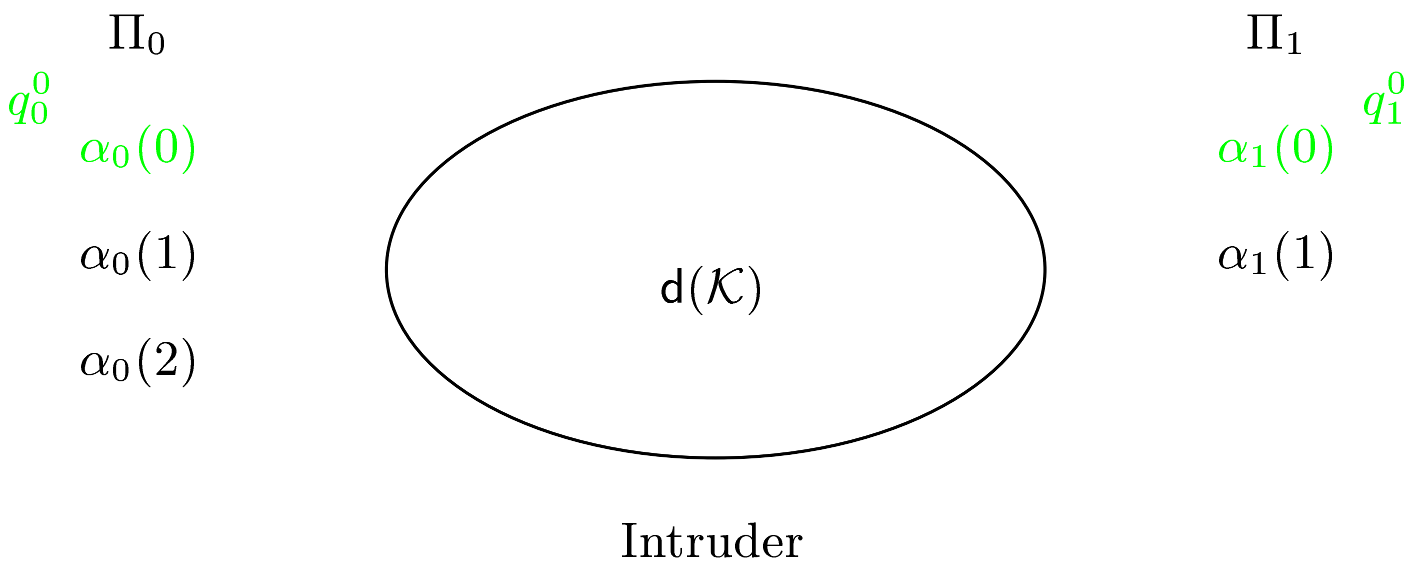
The intruder has complete control over the network and can derive new messages from his knowledge $\mathcal{K} \subseteq \mathcal{M}_\varepsilon$.

The set of messages $\mathsf{d}(\mathcal{K})$ the intruder can derive from $\mathcal{K}$:

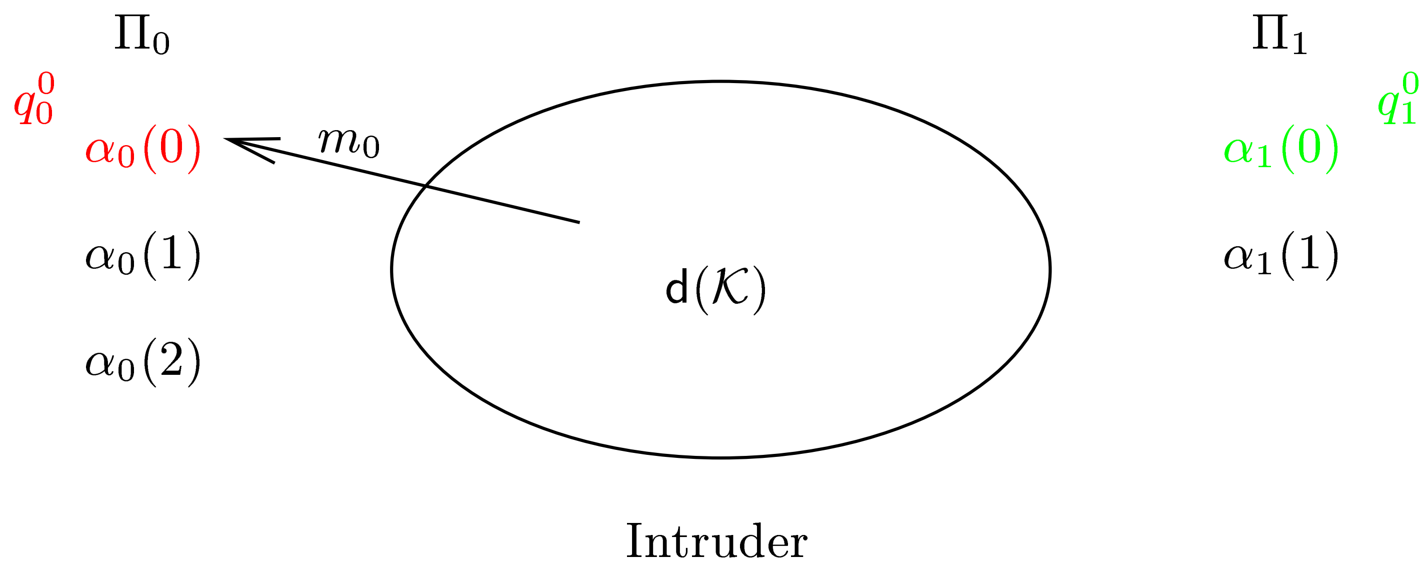
- $\mathcal{K} \subseteq \mathsf{d}(\mathcal{K})$;
- decomposition: if $mm' \in \mathsf{d}(\mathcal{K})$, then $m \in \mathsf{d}(\mathcal{K})$ and $m' \in \mathsf{d}(\mathcal{K})$;
- decryption: if $\mathtt{enc}_a(m) \in \mathsf{d}(\mathcal{K})$ and $a \in \mathsf{d}(\mathcal{K})$, then $m \in \mathsf{d}(\mathcal{K})$;
- composition: if $m \in \mathsf{d}(\mathcal{K})$ and $m' \in \mathsf{d}(\mathcal{K})$, then $mm' \in \mathsf{d}(\mathcal{K})$;
- encryption: if $m \in \mathsf{d}(\mathcal{K})$, $m \neq \varepsilon$, and $a \in \mathcal{N} \cap \mathsf{d}(\mathcal{K})$, then $\mathtt{enc}_a(m) \in \mathsf{d}(\mathcal{K})$;
- hashing: if $m \in \mathsf{d}(\mathcal{K})$ and $m \neq \varepsilon$, then $\mathtt{hash}(m) \in \mathsf{d}(\mathcal{K})$.

Assumption: perfect cryptography

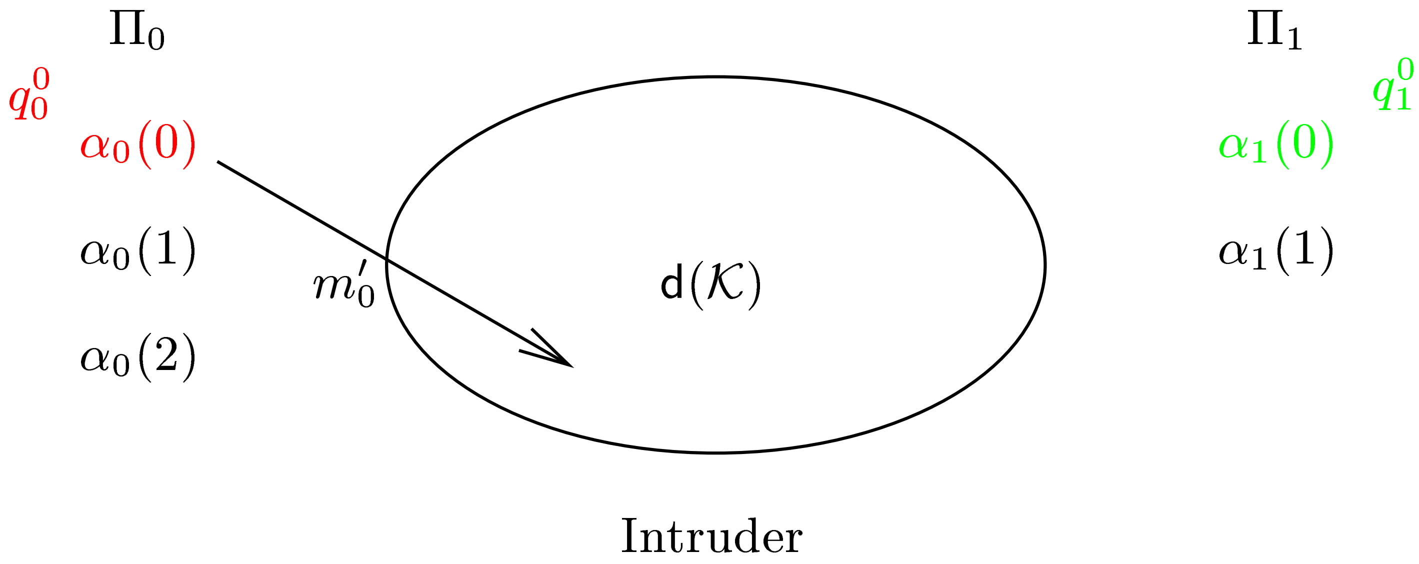
A Generic Protocol Model: Attacks



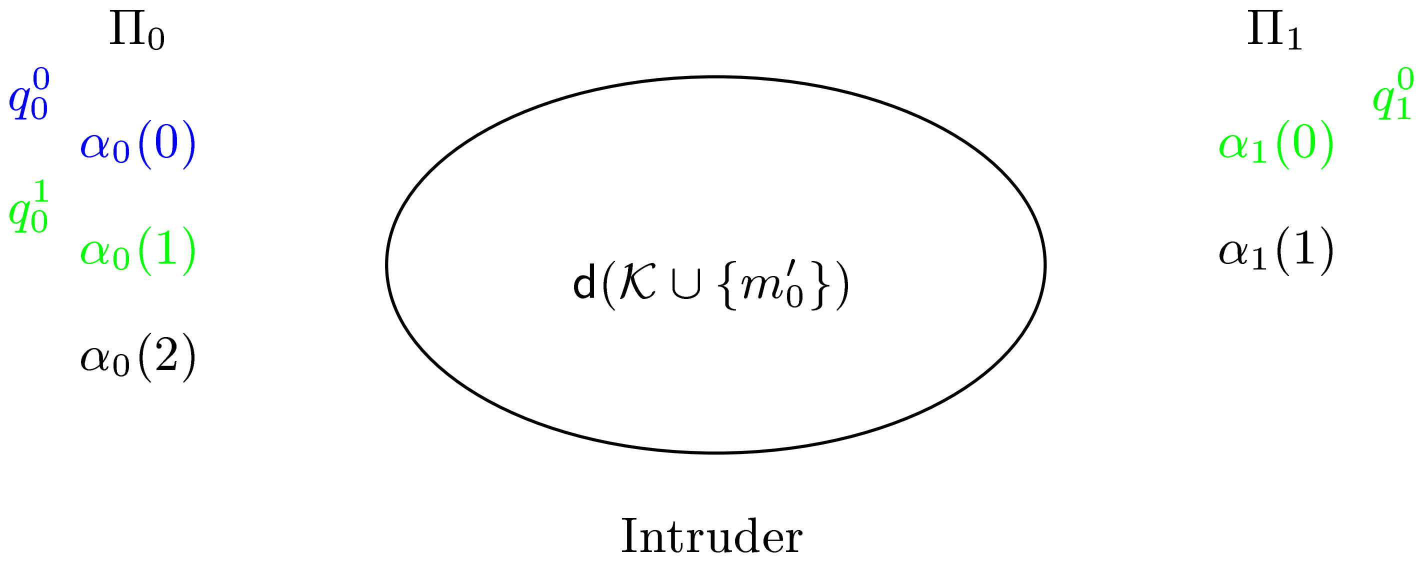
A Generic Protocol Model: Attacks



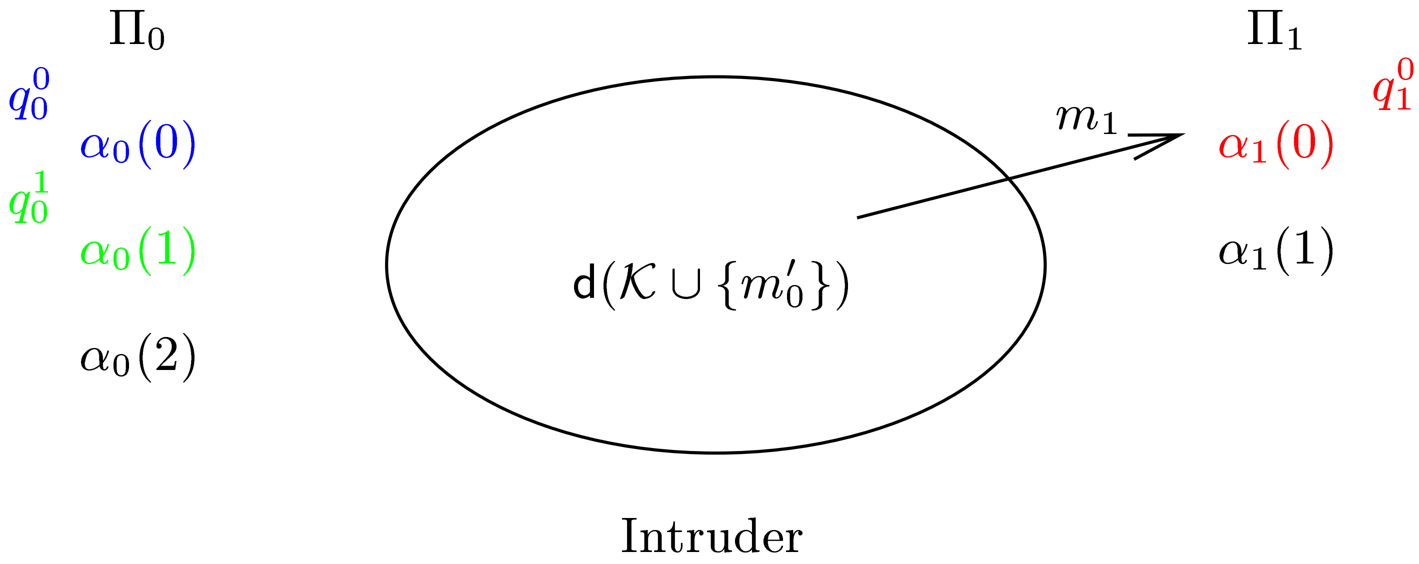
A Generic Protocol Model: Attacks



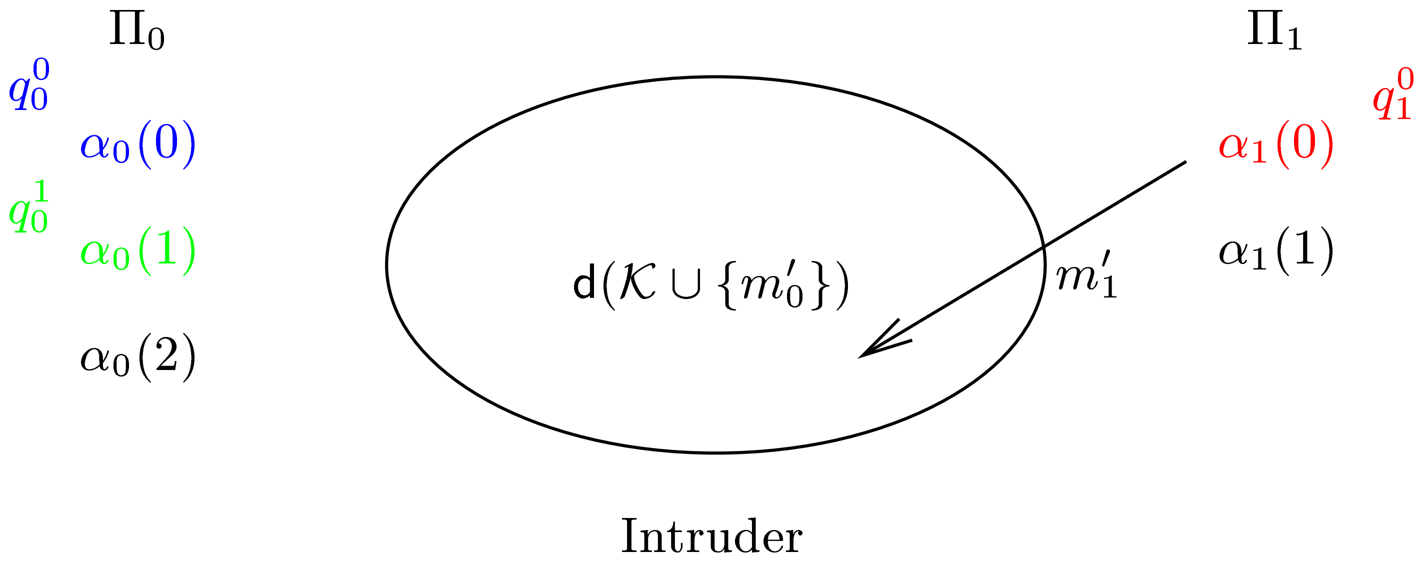
A Generic Protocol Model: Attacks



A Generic Protocol Model: Attacks



A Generic Protocol Model: Attacks



A Generic Protocol Model: Attacks

Π_0

q_0^0

$\alpha_0(0)$

q_0^1

$\alpha_0(1)$

$\alpha_0(2)$

$\mathsf{d}(\mathcal{K} \cup \{m_0', m_1'\})$

Intruder

Π_1

q_1^0

$\alpha_1(0)$

q_1^1

$\alpha_1(1)$

A Generic Protocol Model: Attacks

Π_0

q_0^0

$\alpha_0(0)$

q_0^1

$\alpha_0(1)$

q_0^2

$\alpha_0(2)$

q_0^3

$\mathsf{d}(\mathcal{K} \cup \{m'_0, m'_1, m'_2, m'_3, m'_4\})$

Intruder

Π_1

q_1^0

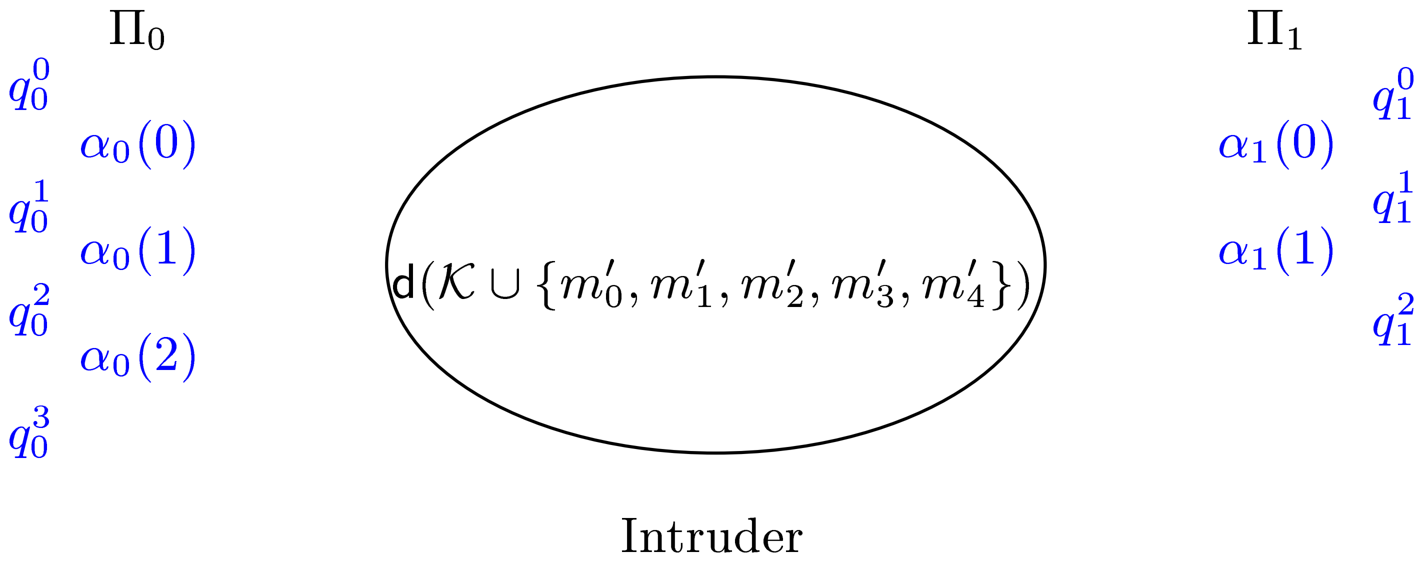
$\alpha_1(0)$

q_1^1

$\alpha_1(1)$

q_1^2

A Generic Protocol Model: Attacks



successful attack: secret $\in \mathsf{d}(\mathcal{K} \cup \{m_0', m_1', m_2', m_3', m_4'\})$

A Generic Protocol Model: Attacks

Π_0 Π_1

q_0^0 $\alpha_0(0)$ q_0^1 $\alpha_0(1)$ q_0^2 $\alpha_0(2)$ q_0^3

$\mathsf{d}(\mathcal{K} \cup \{m_0', m_1', m_2', m_3', m_4'\})$

Intruder

q_1^0 $\alpha_1(0)$ q_1^1 $\alpha_1(1)$ q_1^2

successful attack: secret $\in \mathsf{d}(\mathcal{K} \cup \{m_0', m_1', m_2', m_3', m_4'\})$

$\boxed{\text{Decision problem:}}$ ATTACK

Given a protocol P, decide whether there exists a succesful attack on P.

Questions

1. What instances of the generic protocol model are suitable to capture open-ended protocols?

Questions

1. What instances of the generic protocol model are suitable to capture open-ended protocols?

   What is a suitable computational model for receive-send actions?

Questions

1. What instances of the generic protocol model are suitable to capture open-ended protocols?

   What is a suitable computational model for receive-send actions?

2. Is ATTACK decidable, and if so, what is the complexity?

An Instance of the Generic Protocol Model

$\boxed{\text{Rewriting model:}}$ [Rusinowitch et al. 2001]

receive-send action α: $t \to t'$, where t and t' are terms.

$(m, m') \in \alpha$ iff there exists a substitution σ such that $\sigma(t) = m$ and $\sigma(t') = m'$.

The state of a principal is implicitly given by the substitutions of the variables.

An Instance of the Generic Protocol Model

Rewriting model: [Rusinowitch et al. 2001]

receive-send action α: $t \to t'$, where t and t' are terms.

$(m, m') \in \alpha$ iff there exists a substitution σ such that $\sigma(t) = m$ and $\sigma(t') = m'$.

The state of a principal is implicitly given by the substitutions of the variables.

Theorem: [Rusinowitch et al. 2001]

ATTACK is NP-complete.

An Instance of the Generic Protocol Model

$\boxed{\text{Rewriting model:}}$ [Rusinowitch et al. 2001]

receive-send action α: $t \to t'$, where t and t' are terms.

$(m, m') \in \alpha$ iff there exists a substitution σ such that $\sigma(t) = m$ and $\sigma(t') = m'$.

The state of a principal is implicitly given by the substitutions of the variables.

$\boxed{\text{Theorem:}}$ [Rusinowitch et al. 2001]

$\textsc{Attack}$ is NP-complete.

$\boxed{\text{Problem:}}$ No iteration.
Open-ended protocols cannot be modelled.

Generalizing the Rewriting Model

receive-send action α: set S of rewrite rules of the form $t \rightarrow t'$.

$(m, m') \in \alpha$ iff nondeterministic applications of rewrite rules in S turn m into m'.

Generalizing the Rewriting Model

receive-send action α: set S of rewrite rules of the form $t \rightarrow t'$.

$(m, m') \in \alpha$ iff nondeterministic applications of rewrite rules in S turn m into m'.

We have iteration, but

Theorem: ATTACK is undecidable.

Generalizing the Rewriting Model

receive-send action α: set S of rewrite rules of the form $t \to t'$.

$(m, m') \in \alpha$ iff nondeterministic applications of rewrite rules in S turn m into m'.

We have iteration, but

Theorem: ATTACK is undecidable.

Main reasons: Principals can

1. compare submessages of arbitrary size, or
2. store one message of arbitrary size.

Generalizing the Rewriting Model

receive-send action α: set S of rewrite rules of the form $t \to t'$.

$(m, m') \in \alpha$ iff nondeterministic applications of rewrite rules in S turn m into m'.

We have iteration, but

Theorem: $\textsc{Attack}$ is undecidable.

Main reasons: Principals can

1. compare submessages of arbitrary size, or
2. store one message of arbitrary size.

$\Rightarrow$ Model receive-send actions by transducers.

A Simple Computational Model: Message Transducers

A message transducer $\mathcal{A}$ is a finite automaton with output, which interprets messages as words over the alphabet

$$\Sigma_{\mathcal{N}} := \mathcal{N} \cup \{\text{“}\mathtt{enc}_a(\text{”} \mid a \in \mathcal{N}\} \cup \{\text{“}\mathtt{hash}(\text{”}, \text{“})\text{”}\}.$$

$\boxed{\text{Message transducer } \mathcal{A}\text{:}}$ $(Q, \Sigma_{\mathcal{N}}, I, \Delta, F)$

- finite set of states: Q,
- finite input and output alphabet: $\Sigma_{\mathcal{N}}$,
- set of initial states: $I \subseteq Q$,
- finite set of transitions: $\Delta \subseteq Q \times \Sigma_{\mathcal{N}}^* \times \Sigma_{\mathcal{N}}^* \times Q$, and
- set of final states: $F \subseteq Q$.

such that

“If $\mathcal{A}$ reads a message, $\mathcal{A}$ also returns a message.”

A Simple Computational Model: Message Transducers

A message transducer $\mathcal{A}$ is a finite automaton with output, which interprets messages as words over the alphabet

$$\Sigma_{\mathcal{N}} := \mathcal{N} \cup \{\text{“}\mathtt{enc}_a(\text{”} \mid a \in \mathcal{N}\} \cup \{\text{“}\mathtt{hash}(\text{”}, \text{“})\text{”}\}.$$

Message transducer $\mathcal{A}$: $(Q, \Sigma_{\mathcal{N}}, I, \Delta, F)$

receive-send action α: message transducer $\mathcal{A}$

$(q, m, m', q') \in \alpha$ iff $q \in I$, $q' \in F$, and there is a path in $\mathcal{A}$ from q to q' with input label m and output label m'.

A Simple Computational Model: Message Transducers

A message transducer $\mathcal{A}$ is a finite automaton with output, which interprets messages as words over the alphabet

$$\Sigma_{\mathcal{N}} := \mathcal{N} \cup \{\text{“}\mathtt{enc}_a(\text{”} \mid a \in \mathcal{N}\} \cup \{\text{“}\mathtt{hash}(\text{”}, \text{“})\text{”}\}.$$

$\boxed{\text{Message transducer } \mathcal{A}\text{:}}$ $(Q, \Sigma_{\mathcal{N}}, I, \Delta, F)$

receive-send action α: message transducer $\mathcal{A}$

$(q, m, m', q') \in \alpha$ iff $q \in I$, $q' \in F$, and there is a path in $\mathcal{A}$ from q to q' with input label m and output label m'.

A principal can be described by an extended message transducer of the form

$$(Q, \Sigma_{\mathcal{N}}, \Delta, I_1, F_1, \ldots, I_n, F_n)$$

Modelling the Server S of the Recursive Authentication Protocol

$\downarrow$ $\mathtt{hash}(K_c C S N_c\ \mathtt{hash}(K_b B C N_b\ \mathtt{hash}(K_a A B N_a -)))$

S

$\downarrow$ $\mathtt{enc}_{K_c}(K_{cs} S N_c)\mathtt{enc}_{K_c}(K_{bc} B N_c)\mathtt{enc}_{K_b}(K_{bc} C N_b)\mathtt{enc}_{K_b}(K_{ab} A N_b)\mathtt{enc}_{K_a}(K_{ab} B N_a)$

$\mathcal{A}_S$ works as follows:

1. Read $\mathtt{hash}(K_c C S N_c$ output $\mathtt{enc}_{K_c}(K_{cs} S N_c)$, and store (C, N_c).
2. Read $\mathtt{hash}(K_b B C N_b$ output $\mathtt{enc}_{K_c}(K_{bc} B N_c)\mathtt{enc}_{K_b}(K_{bc} C N_b)$, and store (B, N_b).
3. $\ldots$

Main Result

Theorem: In the transducer-based model,

ATTACK is decidable and PSPACE-hard.

Main Result

Theorem: In the transducer-based model,

ATTACK is decidable and PSPACE-hard.

Proof sketch:

Hardness result: Reduction from the finite automata intersection problem [Kozen].

Decidability: two sources of infinite behavior

1. The intruder can choose a message from an infinite message space $d(\mathcal{K})$.
2. The length of paths in message transducers is unbounded.

Pumping Argument

In an attack the length of paths can be bounded in the size of the problem instance:

$$q_i^I \xrightarrow[m_i']{m_i} \!\!>\!\!> q_i^F \quad \overset{\text{truncate}}{\Longrightarrow} \quad q_i^I \xrightarrow[\overline{m}_i']{\overline{m}_i} \!\!>\!\!> q_i^F$$

Paths must be truncated such that

1. $\overline{m}_i \in \mathsf{d}(\mathcal{K} \cup \{m_0', \ldots, m_{i-1}'\})$, and
2. there exists a successful attack if $\mathsf{d}(\mathcal{K} \cup \{m_0', \ldots, m_i'\})$ is replaced by $\mathsf{d}(\mathcal{K} \cup \{m_0', \ldots, \overline{m}_i'\})$.

Pumping Argument

In an attack the length of paths can be bounded in the size of the problem instance:

$$q_i^I \xrightarrow[m_i']{m_i}\!\!\!>\!\!> q_i^F \quad \overset{\text{truncate}}{\Longrightarrow} \quad q_i^I \xrightarrow[\overline{m}_i']{\overline{m}_i}\!\!\!>\!\!> q_i^F$$

Paths must be truncated such that

1. $\overline{m}_i \in \mathsf{d}(\mathcal{K} \cup \{m_0', \ldots, m_{i-1}'\})$, and
2. $\overline{m}_i'$ is equivalent to m_i' w.r.t. a so-called solvability preserving ordering.

Pumping Argument

In an attack the length of paths can be bounded in the size of the problem instance:

$$q_i^I \xrightarrow[\beta\gamma\delta]{}\!\!\!\!\twoheadrightarrow q_i^F \quad \overset{\text{truncate}}{\Longrightarrow} \quad q_i^I \xrightarrow[\beta\delta]{}\!\!\!\!\twoheadrightarrow q_i^F$$

Proposition:

If $\beta\gamma$ is equivalent to β w.r.t. a so-called path truncation ordering, then $\beta\gamma\delta$ is equivalent to $\beta\delta$ w.r.t. the solvability preserving ordering.

Pumping Argument

In an attack the length of paths can be bounded in the size of the problem instance:

$$q_i^I \xrightarrow[\beta\gamma\delta]{}\!\!\!\gg q_i^F \quad \overset{\text{truncate}}{\Longrightarrow} \quad q_i^I \xrightarrow[\beta\delta]{}\!\!\!\gg q_i^F$$

Proposition:

If $\beta\gamma$ is equivalent to β w.r.t. a so-called path truncation ordering, then $\beta\gamma\delta$ is equivalent to $\beta\delta$ w.r.t. the solvability preserving ordering.

Proposition:

The path truncation ordering has an index bounded in the size of the problem instance.

Conclusion

- Open-ended protocols require receive-send actions with iteration.
- Undecidability if principals have infinite memory or can compare messages of arbitrary size for equality.
- Message transducers: a simple computational model for receive-send actions; see the paper for a comparison with the rewriting approach.
- The transducer-based model is decidable and PSPACE-hard; new proof techniques were required. Complexity upper bound?
- This should carry over rather easily to public-key encryption and authentication.

Future work

- Tree transducers
- Extensions: Complex keys, infinite number of sessions and principals, restricted equality tests, ...
- Combining the rewriting and the transducer-based approach
- Other applications: modes of encryption