VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Smart cards: Representing Cryptographic protocols with Tree automata

Ondřej Klubal

LTA 2012 - 11th December

Table of contents

Smart cards - introduction

- plastic cards whith integrated chip
- can provide identification, authentication, data storage and application processing
- standardized size and interface (ISO/IEC 7810, ISO/IEC 7816)
  - Contact - 8 gold-plated pads
  - Contactless - comunication and power provided via RF signals
  - Hybrid - two chips or chip whith dual interface

- operating systems
  - static - Can read and wtrite data, data processing. Allmost all cryptographic cards (SIM etc.).
  - dynamic - Can load program code. (JavaCards)

Smart cards - Communication protocols

T=0

- asynchronous half duplex character transmission
- default protocol after "answer to reset", most used
- strictly separated request and reply
- limited error correction, master slave relationship
- ISO/OSI physical layer (1)

T=1

- asynchronous half duplex block transmission
- allow data transfer on both directions in a same command
- can provide flow control, block chaining, error correction
- ISO/OSI data link layer (2), physical (1) same as T=0

T=CL

- only used for Contactless cards

Smart cards - Communication security and attacks

Is communication with card safe?

- communication protocols does not provide confidentiality or autentication
- attacks to communication interface:
  - passive - listening
  - active - MITM (attacker changes communication)
  - side channel attacks - Simple/Differential power analysis
  - etc.

Solution?

- use of cryptographic protocols needed
- PKI (Public-key infrastructure)

Cryptographic protocols

Definition

A security protocol is an abstract or concrete protocol that performs a security-related function and applies cryptographic methods.

Uses

- subject authentication
- key distribution
- combination of previous

Channel types:

- secure channel - tamper resistant, overhearing resistant
- confidential channel - overhearing resistant
- authentic channel - tamper resistant

Is a cryptografic protocol secure? Verification is needed...

Formal verification of cryptographic protocols

- proving or disproving correctness on formal protocol specification

AVISPA - complex tool for cyptographic protocol verification

- HLPSL-High Level Protocol Specification Language - expressive language for modeling communication and security protocols
- CL-AtSe (Constraint-Logic-based Attack Searcher)
- OFMC (On-the-fly Model-Checker)
- SATMC - builds a propositional formula, SAT solver checking
- TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) - approximates the intruder knowledge by using regular tree languages and rewriting to produce under and over approximations

Cryptographic protocol example

Otway-Rees protocol

$Message1$	$A \rightarrow B$	$M; A; B; \{N_a; M; A; B\}_{K_{as}}$
$Message2$	$B \rightarrow S$	$M; A; B; \{N_a; M; A; B\}_{K_{as}}; \{N_b; M; A; B\}_{K_{bs}}$
$Message3$	$S \rightarrow B$	$M; \{N_a; K_{ab}\}_{K_{as}}; \{N_b; K_{ab}\}_{K_{bs}}$
$Message4$	$B \rightarrow A$	$M; \{N_a; K_{ab}\}_{K_{as}}$

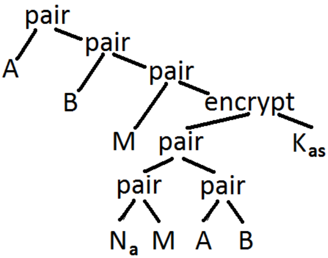
- $\{X\}_K$ is a notation for X encrypted with the key K
- M is session identifier, N_a and N_b are nonces ("random" arbitrary numbers), A and B are clients, S is trusted server
- goal is to distibute shared key K_{ab} for later communication over secure channel between A and B
- there are a variety of attacks on this protocol

Cryptographic protocol - tree representation

Modified Dolev-Yao Model

- encrypt(X, K) - symetric encryption primitive taking a piece of data X and key K
- pair(A, B) - primitive for packaging (pairing) data
- a protocol making use of symmetric encryption is modeled with messages built on the following algebra: encrypt$(\cdot, \cdot)$) and pair$(\cdot, \cdot)$

$$\mathtt{pair}(A,\mathtt{pair}(B,\mathtt{pair}(M,\mathtt{encrypt}(\mathtt{pair}(\mathtt{pair}(N_a, M),\mathtt{pair}(A, B)), K_{as}))))$$



Formal Semantics

- signature $\mathcal{F}$ is a couple (Σ, a), where Σ is a set of function names and a is a function from Σ to set of nonnegative integers N called arity. From previous example function names are `encrypt` and `pair`, each of arity 2 and various constants (N_a) of arity 0
- free algebra of terms $T(\mathcal{F})$
- $\mathcal{O}$ is a subset of the function symbols found in $\mathcal{F}$, $\mathcal{O}_n$ is a subset of elements of arity n

Bottom-up Tree automata

Bottom-up Tree Automata is a tuple:

$$M = (Q, \mathcal{F}, Q_f, \Delta)$$

- Q is a set of states, $Q_f \subseteq Q$
- $\mathcal{F}$ is a ranked alphabet, which consists of an alphabet Σ and a function $a : \Sigma \rightarrow \mathbf{N}$
  Example: $\mathcal{F} = \{0, 1, not(), and(,), or(,)\}$
- Δ is a set of rewrite rules of the following type:
  $f(q_1, \cdots, q_n) \rightarrow q$, where $n = a(f)$
- transition relation is defined as:

$$t \rightarrow_M t' \begin{cases} \exists C \in \mathcal{C}(\mathcal{F} \cup Q), \\ \exists f(q_1, \cdots, q_n) \rightarrow q \in \Delta, \\ t = C[f(_1, \cdots, q_n)], \\ t' = C[q] \end{cases}$$

Bottom-up Tree automata - Example

Let $M = (Q, \mathcal{F}, Q_f, \Delta)$, where $\mathcal{F} = \{0, 1, not(), and(,), or(,)\}$,
$Q = \{q_0, q_1\}, Q_f = \{q_1\}$ and $\Delta =$

$$\begin{array}{llll} \{0 \rightarrow q_0 & 1 \rightarrow q_1 & not(q_0) \rightarrow q_1 & not(q_1) \rightarrow q_0 \\ or(q_0, q_0) \rightarrow q_0 & or(q_0, q_1) \rightarrow q_1 & or(q_1, q_0) \rightarrow q_1 & or(q_1, q_1) \rightarrow q_1 \\ and(q_0, q_0) \rightarrow q_0 & and(q_0, q_1) \rightarrow q_0 & and(q_1, q_0) \rightarrow q_0 & or(q_1, q_1) \rightarrow q_1\} \end{array}$$

- accepted tree language by M is set of true boolean expressions over $\mathcal{F}$
- Example evaluation of $and(not(or(0, 1)), or(1, not(0)))$:

$$and(not(or(0, 1)), or(1, not(0))) \xrightarrow[M]{*}$$

$$and(not(or(q_0, q_1)), or(q_1, not(q_0))) \xrightarrow[M]{*}$$

$$and(not(q_1), or(q_1, q_1)) \xrightarrow[M]{*} and(q_0, q_1) \xrightarrow[M]{} q_0$$

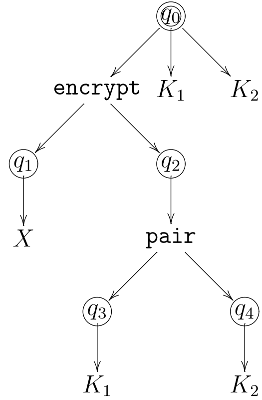
Top-Down Tree automata

Top-Down Tree Automata is a tuple:

$$M = (Q, \mathcal{F}, q_0, \Delta)$$

- Q is a set of states, $q_0 \in Q$ is the inital state
- Δ is a set of rewrite rules of the following type: $q(f(x_1, \cdots, x_n)) \to f(q_1(x_1), \cdots, q_n(x_n))$, where $n = a(f)$,$x_1, \cdots, x_n$ being variables and q_x states
- When $n = 0$, rewrite rule has form $q(a) \to a$. Define $L_q(a) = \{t \in T(\mathcal{F}) | q(t) \xrightarrow[M]{*} t\}$, then $L(a) = L_{q_0}(M)$ is language recognized by M.

Top-Down Tree automata



$$
\begin{aligned}
q_0(\texttt{encrypt}(x,y)) &\rightarrow_A \texttt{encrypt}(q_1(x), q_2(y))\\
q_0(K_1) &\rightarrow_A K_1\\
q_0(K_2) &\rightarrow_A K_2\\
q_1(X) &\rightarrow_A X\\
q_2(\texttt{pair}(x,y)) &\rightarrow_A \texttt{pair}(q_3(x), q_4(y))\\
q_3(K_1) &\rightarrow_A K_1\\
q_4(K_2) &\rightarrow_A K_2
\end{aligned}
$$

- $M = (\{q_0, q_1, q_2, q_3, q_4\}, \mathcal{O}_c, q_0, \Delta)$
- $\mathcal{O}_c$ is a signature with added constants $\{X, K_1, K_2\}$
- recognizing $\{$ encrypt $(X,$ encrypt $(K_1, K_2)), K_1, K_2\}$

Conclusion

Operations with tree automata

- Union (joining at root)
- Substituion
- Matching (intersect test)

What is it good for?

- modeling cryptographic protocol
- use for verification

Bibliography

- H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison and M. Tommas: Tree Automata Techniques and Applications, Available on: `http://www.grappa.univ-lille3.fr/tata`, release October, 12th 2007
- D. Monniaux. Abstracting Cryptographic Protocols with Tree Automata. In Static Analysis Symposium (SAS'99), volume 1694 of Lecture Notes on Computer Science, pages 149 - 163. Springer Verlag, Sept. 1999.
- R. Shaikh and S. Devane: Formal verification of payment protocol using AVISPA, International Journal for Infonomics, Vol.3, Issue 3, September 2010
- Smart Card Tutorial, 1992-1994, Available on: `http://www.smartcard.co.uk/tutorials/sct-itsc.pdf`
- P. Otčenášek: IBS: Security and computer networks, BUT FIT, 2010

Thank you for attention