On the Use of Probabilistic Automata for Security Proofs

Roberto Segala
University of Verona
Motivation

• **Proofs of cryptographic protocols are hard**
  - Especially in the computational model
  - Limited mathematical tools available
    • ... or limited willingness to work out the details

• **Symbolic methods help**
  - But proving soundness requires classical proofs

• **Many proofs rely on correspondence between computations of different systems**
  - Concurrency theory has a lot to say

• **Can we take advantage of concurrency theory**
  - ... directly in the computational model?
Hierarchical Compositional Verification

Modules verified separately

Some properties verified here

\[ S \]

\[ S_1 \quad S_2 \quad S_3 \]

\[ I_{11} \quad I_{12} \quad I_2 \quad I_3 \]
Implementation

• Typically some form of behavioral inclusion
  - Traces
    • Ordinary, complete, quiescent, fair
  - Failures
    • Traces followed by actions the system refuses to perform
  - Tests
    • Occurrence of some success event in appropriate contexts

• Nice properties
  - Transitive
  - Compositional
  - Affine with logical implication
    • ... when properties are sets of behaviors

• Hard to check
  - Usually Pspace-complete
  - But simulation relations help
Proving Implementation

• Behavioral inclusion
  - Behaviors are full computations
    • Possibly infinite length
  - Properties of complex objects
    • Global reasoning
  - Easy to end up with “proofs by intuition”

• Simulation relations
  - Sound for behavioral inclusion
  - Properties of single computational steps
    • Local reasoning
  - Easier to be rigorous
Nondeterminism and Probability

- **Nondeterminism**
  - Relative speeds of processes
  - Unknown behavior of users
    - Adversary in DY model
  - Underspecification
  - Abstraction
    - Forget about probabilities

- **Probability**
  - User behavior may obey probability laws
  - Processes may flip coins
    - Randomized algorithms, protocols
    - Nonces, keys, ...
Overview

• Probabilistic Automata
  - Definition, executions, traces
  - Composition, projection
  - Behavioral inclusion
  - Simulation relations

• Task Probabilistic I/O Automata
  - A way to restrict nondeterminism
  - Case study with oblivious transfer
  - Nondeterminism may leak information
  - Reasoning up to negligible errors

• Approximated simulation relations
  - Relate automata that fail with negligible probability with automata that do not fail
  - Case study with agent authentication

• Using Probabilistic Automata for DY-soundness
  - A possibility?
The Main Idea

- Add probability to Concurrency Theory
  - Nondeterminism should remain
  - Should obtain a conservative extension

- Proposals to tackle the problem
  - Replace points with measures
  - Replace functions with measurable functions
Automata

\[ A = (Q, q_0, E, H, D) \]

- Transition relation: 
  \[ D \subseteq Q \setminus (E \cup H) \setminus Q \]
- Internal (hidden) actions
- External actions: \( E \cap H = \emptyset \)
- Initial state: \( q_0 \in Q \)
- States
Probabilistic Automata

\[ PA = (Q, q_0, E, H, D) \]

- Transition relation: \( D \subseteq Q \setminus (E \cup H) \setminus \text{Disc}(Q) \)
- Internal (hidden) actions
- External actions: \( E \cap H = \emptyset \)
- Initial state: \( q_0 \in Q \)
- States
Example: Automata

\[ A = (Q, q_0, E, H, D) \]

Execution: \[ q_0 \; n \; q_1 \; n \; q_2 \; ch \; q_3 \; coffee \; q_5 \]

Trace: \[ n \; n \; coffee \]
Example: Probabilistic Automata
Example: Probabilistic Automata

\[
\begin{align*}
q_0 & \xrightarrow{\text{flip}} q_t \\
q_t & \xrightarrow{\text{flip}} q_h \\
q_h & \xrightarrow{\text{beep}} q_p \\
q_t & \xrightarrow{\text{buzz}} q_z
\end{align*}
\]
Example: Probabilistic Automata

What is the probability of beeping?
Example: Probabilistic Executions

\[ \mu(\text{beep}) = \frac{1}{2} \]

\[ \mu(\text{beep}) = \frac{2}{3} \]
Example: Probabilistic Executions

\begin{itemize}
  \item \text{q}_0 \rightarrow \text{q}_1 \quad \text{fair} \quad \frac{1}{2}
  \item \text{q}_1 \rightarrow \text{q}_2 \quad \text{unfair} \quad \frac{1}{2}
  \item \text{q}_2 \rightarrow \text{q}_3 \quad \text{flip} \quad \frac{2}{3}
  \item \text{q}_3 \rightarrow \text{q}_4 \quad \text{beep} \quad \frac{1}{3}
  \item \text{q}_4 \rightarrow \text{q}_5 \quad \text{beep} \quad \frac{2}{6}
  \item \text{q}_5 \rightarrow \text{q}_5 \quad \text{flip} \quad \frac{1}{4}
\end{itemize}

\[ \frac{7}{12} \]
Cones and Measures

- **Cone of** \( \alpha \)
  - Set of executions with prefix \( \alpha \)
  - Represent event “\( \alpha \) occurs”

- **Measure of a cone**
  - Product edges of \( \alpha \)

extends uniquely \( \sigma \)-field generated by cones
## Schedulers - Probabilistic Executions

**Scheduler**

**Function**

\[ \sigma : \text{exec}^*(A) \rightarrow \text{SubDisc}(D) \]

\[
\begin{align*}
\text{if } \sigma(\alpha)((q,a,\nu)) > 0 \text{ then } q &= lstate(\alpha)
\end{align*}
\]

**Probabilistic execution**

generated by \( \sigma \) from state \( r \)

**Measure**

\[ \mu_{\sigma,r}(C_s) = 0 \quad \text{if} \quad r \neq s \]

\[ \mu_{\sigma,r}(C_r) = 1 \]

\[ \mu_{\sigma,r}(C_{\alpha a q}) = \mu_{\sigma,r}(C_\alpha) \cdot \left( \sum_{(s,a,\nu) \in D} \sigma(\alpha)((s,a,\nu)) \nu(q) \right) \]
Summing Up

Automata

Probabilistic Automata

Executions

Probabilistic Executions
(measures over executions)

Traces

Trace inclusion

Trace function

???

???

implementation relation

schedulers
Related Models

- **Rabin Probabilistic Automata** [Rab63]
  - Deterministic Probabilistic Automata
  - Introduced in context of language theory
  - Actions have a different use

- **Reactive Systems** [LS89, GSST90]
  - Deterministic Probabilistic Automata

- **Markov Decision Processes** [Bel57]
  - Deterministic Probabilistic Automata
    - Though actions have a completely different use
    - ...plus reward functions

- **Labeled Concurrent Markov Chains** [HJ89]
  - Probabilistic Automata where
    - States are partitioned into deterministic and probabilistic
    - Nondeterministic states enable several ordinary transitions
    - Probabilistic states enable one transition
Parallel Composition
Composition of Probabilistic Automata

\[ A_1 = (Q_1, q_1, E_1, H_1, D_1) \quad \Box \quad A_2 = (Q_2, q_2, E_2, H_2, D_2) \]

\[ A_1 \parallel A_2 = (Q_1 \cupdot Q_2, (q_1, q_2), E_1 \cup E_2, H_1 \cup H_2, D) \]

\[ D = \begin{cases} 
(q, a, (s_1, s_2)) & \text{if } a \in E_i \cup H_i \text{ then } (\pi_i(q), a, s_i) \in D_i \\
& \text{if } a \notin E_i \cup H_i \text{ then } s_i = \pi_i(q) \quad i \in \{1, 2\} 
\end{cases} \]

\[ D = \begin{cases} 
(q, a, \mu_1 \times \mu_2) & \text{if } a \in E_i \cup H_i \text{ then } (\pi_i(q), a, \mu_i) \in D_i \\
& \text{if } a \notin E_i \cup H_i \text{ then } \mu_i = \delta(\pi_i(q)) \quad i \in \{1, 2\} 
\end{cases} \]
Example: Composition of Automata

\[ E = \{n,d,choc,coffee\} \]

\[
\begin{align*}
(q_0,s_0) &\xrightarrow{d} (q_2,s_1) &\xrightarrow{choc} (q_4,s_2) \\
(q_3,s_1) &\xrightarrow{ch} (q_5,s_3)
\end{align*}
\]
Ex. Composition of Probabilistic Automata

\[
\begin{aligned}
&
\text{\textbf{ch}} & \quad s_0 \rightarrow s_1 & \quad \text{fair} & \quad s_3 \\
& & s_0 \rightarrow s_2 & \quad \text{unfair} & \quad s_4 \\
& & s_1 \rightarrow s_3 & & \\
& & s_2 \rightarrow s_4 & & \\
& & s_3 \rightarrow (s_3, q_1) & \quad \text{flip} & \quad 1/2 & \quad \text{beep} \\
& & s_4 \rightarrow (s_4, q_3) & \quad \text{flip} & \quad 2/3 & \quad \text{beep} \\
& \quad (s_0, q_0) \quad \text{ch} & \quad 1/2 & \quad & \quad (s_0, q_0) \quad \text{unfair} & \quad 1/2 \\
& & (s_1, q_0) \quad \text{fair} & \quad (s_3, q_3) & \quad (s_3, q_4) \\
& & (s_2, q_0) \quad \text{unfair} & \quad (s_4, q_2) & \quad (s_4, q_4) \\
& & (s_3, q_1) \quad \text{flip} & \quad 1/2 \\
& & (s_3, q_3) \quad \text{flip} & \quad 1/2 & \quad (s_3, q_5) \\
& & (s_3, q_4) \quad \text{flip} & \quad 2/3 \\
& & (s_4, q_2) \quad \text{flip} & \quad 1/3 \\
& & (s_4, q_4) \quad \text{flip} & \quad 1/3 & \quad (s_4, q_5) \\
\end{aligned}
\]
Projections

Let $\alpha$ be an execution of $A_1 \parallel A_2$

$$\alpha = (q_0,s_0) d (q_2,s_1) ch (q_3,s_1) coffee (q_5,s_3)$$

What are the contributions of $A_1$ and $A_2$?

$$\pi_1(\alpha) \equiv q_0 \; d \; q_2 \; ch \; q_3 \; coffee \; q_5$$

$$\pi_2(\alpha) \equiv s_0 \; d \; s_1 \; coffee \; s_3$$

Theorem

$$\alpha \in \text{execs}(A_1 \parallel A_2) \iff \forall i \in \{1,2\} \; \pi_i(\alpha) \in \text{execs}(A_i)$$
Measure Theory: Image Measure

- **Measurable function** from \((\Omega_1, F_1)\) to \((\Omega_2, F_2)\)
  - Function \(f\) from \(\Omega_1\) to \(\Omega_2\)
  - For each element \(X\) of \(F_2\), \(f^{-1}(X) \in F_1\)

- **Image measure** \(f(\mu)\)
  - \(f(\mu)(X) = \mu(f^{-1}(X))\)
Projections

The projection function is measurable
\[ \pi(\mu) : \text{image measure under } \pi \text{ of } \mu \]

**Theorem**

If \( \mu \) is a probabilistic execution of \( A_1 \parallel A_2 \) then
\[ \pi_i(\mu) \] is a probabilistic execution of \( A_i \)
Example: Projection

Projection onto right component

Note that the scheduler is randomized.
Let $M = MP || CF$

- Suppose that $MP$ satisfies $\Phi$ provided that the environment ($CF$) satisfies $\Psi$
- Suppose that $CF$ satisfies $\Psi$ with probability $p$
- Can I conclude that $M$ satisfies $\Phi$ with probability $p$?

\[
\begin{align*}
\text{MP} & \models \Psi \Rightarrow \Phi & \text{CF} & \models \lceil \Psi \rceil \geq p \\
\hline
M & \models \lceil \Phi \rceil \geq p
\end{align*}
\]

- This example is taken from a real case study [PLS01]
  - Randomized consensus protocol of Aspnes and Herlihy [AH90]
  - $MP$ is a complex non randomized protocol
  - $CF$ is a relatively simple randomized coin flipper
Formal Argument

Let $\mu$ be a probabilistic execution of $M$.

$\mu(\pi_2^{-1}(\Psi)) \geq p$

$\pi_1(\mu)$ sat. $\Phi$

$\pi_2(\mu)(\Psi) \geq p$

projection

inverse image
Language Inclusion
Summing Up

Automata

scheduled

Probabilistic Automata

Executions

Probabilistic Executions (measures over executions)

Probabilistic Executions

trace function

(Probabilistic Executions (measures over executions))

Traces

Trace distributions (measures over traces)

Trace distributions

implementation relation

Trace inclusion

Trace distribution inclusion

On the use of Probabilistic Automata for Security Proofs
Atagawa, April 6-9 2009
Roberto Segala - University of Verona
Trace Distributions

The *trace* function is measurable

Trace distribution of $\mu$

$tdist(\mu) : \text{image measure under trace of } \mu$

Trace distribution inclusion preorder

$A_1 \leq_{TD} A_2 \iff tdist(A_1) \subseteq tdist(A_2)$
Trace Distribution Inclusion is not Compositional

\[
\begin{align*}
(s_0, c_0) & \xrightarrow{a} (s_1, c_0) \\
(s_0, c_1) & \xrightarrow{b} (s_2, c_3) \\
(s_0, c_2) & \xrightarrow{f} (s_1, c_4) \\
(s_1, c_2) & \xrightarrow{e} (s_1, c_3) \\
(s_1, c_3) & \xrightarrow{b} (s_2, c_3) \\
(s_1, c_4) & \xrightarrow{c} (s_3, c_4)
\end{align*}
\]
How to Get Compositionality

- Restrict the power of composition
  - Probabilistic reactive modules [AHJ01]
  - Switched probabilistic I/O automata [CLSV04]

- Trace Distribution Precongruence
  - Coarsest precongruence included in preorder
    - That is: close under all contexts
  - Alternative characterizations
    - Principal context [Seg95]
    - Testing [Seg96]
    - Forward simulations [LSV03]
... yet, Proving Language Inclusion is Difficult

- Language inclusion is a global property
  - Need to see the whole result of resolving nondeterminism

- We seek local proof techniques
  - Local arguments are easier

- We use simulation relations
Simulations
Forward Simulations (Automata)

Forward simulation from $A_1$ to $A_2$ ($A_1 \leq_F A_2$)

Relation $R \subseteq Q_1 \times Q_2$ such that

$$\forall q, s, a, q' \exists s'$$

Diagram:

- $q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_3$
- $q_1 \xrightarrow{a} q_2 \xrightarrow{c} q_4$
- $s_0 \xrightarrow{a} s_1$
- $s_1 \xrightarrow{b} s_3 \xrightarrow{c} s_4$
- $s \xrightarrow{a} s'\quad R$
- $q \xrightarrow{a} q'\quad R$
Simulation Implies Trace Inclusion

- The step condition can be applied repeatedly

\[ S \xrightarrow{a} S_1 \xrightarrow{b} S_2 \xrightarrow{c} S_3 \xrightarrow{d} S_4 \xrightarrow{} \ldots \]
\[ q \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{c} q_3 \xrightarrow{d} q_4 \xrightarrow{} \ldots \]

- Thus existence of simulation implies trace inclusion
  - Even more it implies a close correspondence between executions
Forward Simulations

Forward simulation from $A_1$ to $A_2$ ($A_1 \leq_F A_2$)
Relation $R \subseteq Q_1 \times Q_2$ such that

Lifting of $R$

\[
\begin{align*}
1/2 & \ q_1 \\
1/2 & \ q_2
\end{align*}
\]

\[
\begin{align*}
& \ 1/3 \\
& \ 1/6 \\
& \ 1/6 \\
& \ 1/3 \\
& \ 1/3 \\
& \ 1/3 \\
& \ 1/6 \\
& \ 1/6 \\
& \ 1/3
\end{align*}
\]

∀ q, s, a, μ' ∃ σ'

S → α → σ'

R → α → μ'
Considerations about Lifting

- It is the solution of a maximum flow problem
- Alternative characterization
  - $\mu_1 R \mu_2$ iff for each upward closed set $X$
    - $\mu_1(X) \equiv \mu_2(X)$

\[
\begin{array}{c}
\text{S} \\
\quad \quad 1/2 q_1 \\
\quad \quad 1/2 q_2 \\
\text{Lifting of } R \\
\quad \quad 1/6 \\
\quad \quad 1/6 \\
\quad \quad 1/3 \\
\quad \quad 1/3 \\
\quad \quad 1/6 \\
\text{d} \\
\end{array}
\]

\[
\begin{array}{c}
\text{s}_1 \quad 1/2 \\
\text{s}_2 \quad 1/3 \\
\text{s}_3 \quad 1/3 \\
\end{array}
\]
Lifting and Transfer of Masses

$q_1 \quad q_2 \quad s_1 \quad s_2 \quad s_3$
Lifting and joint Measures

\( \mu_1 \preceq \mu_2 \) iff there exists a probability measure \( w \) on \( Q_1 \times Q_2 \) such that

- \( \text{support}(w) \subseteq R \)
  - That is, \( w(s_1,s_2)>0 \) implies \( s_1 \preceq s_2 \)
- \( w(s_1,Q_2) = \mu_1(s_1) \)
  - That is, the left marginal is \( \mu_1 \)
- \( w(Q_1,s_2) = \mu_2(s_2) \)
  - That is, the right marginal is \( \mu_2 \)
Example: Simulations
Simulation Implies Trace Inclusion

- The step condition can be applied repeatedly

\[ s \rightarrow \rho_1 \rightarrow \rho_2 \rightarrow \rho_3 \rightarrow \rho_4 \rightarrow \cdots \]

\[ q \rightarrow \mu_1 \rightarrow \mu_2 \rightarrow \mu_3 \rightarrow \mu_4 \rightarrow \cdots \]
Probabilistic I/O Automata

- Probabilistic Automata where:
  - External actions partitioned
    - Input actions
    - Output actions
  - Input actions always enabled

- In parallel composition:
  - Each action is output of at most one automaton

- Therefore:
  - The environment never blocks output actions
  - Language inclusion preserves more properties
  - We know always who controls each action
Case Study:

Oblivious Transfer

Even, Goldreich, Lempel 85

Canetti, Cheung, Kaynar, Liskov, Lynch, Pereira, Segala
UC-Framework [Canetti]

- Ideal functionality
- Simulator
- Environment
- Real protocol
- Adversary
Oblivious Transfer

- **Ideal functionality**
  - **Receive**
    - input \( x \in \{0,1\} \rightarrow \{0,1\} \) (just to avoid writing \( x_0, x_1 \))
    - input \( i \in \{0,1\} \)
  - **Return**
    - \( x(i) \) (or could be \( x_i \))

- **Failure model**
  - Either Transmitter or Receiver may be corrupt
  - Adversary sees input of faulty agents
  - Faulty agents send output to adversary
  - Adversary may only forward messages and/or talk to environment

- **In practice we have four cases**
  - We consider case where no agent is faulty
### Automaton for Ideal Functionality

#### No Faulty Agents

<table>
<thead>
<tr>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
</tr>
</tbody>
</table>
| $\text{in}(x)_T$ $x \in \{0,1\} \rightarrow \{0,1\}$
| $\text{in}(i)_R$ $i \in \{0,1\}$ |
| **Output**       |
| $\text{out}(w)_R$ |
| **State**        |
| $x_{val} \in \{0,1\} \rightarrow \{0,1\}$ initially $\bot$
| $i_{val} \in \{0,1\} \cup \{\bot\}$ initially $\bot$ |

<table>
<thead>
<tr>
<th>Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>in</strong>(x)$_T$ Effect</td>
</tr>
<tr>
<td>If $x_{val} = \bot$ then $x_{val} := x$</td>
</tr>
<tr>
<td><strong>in</strong>(i)$_R$ Effect</td>
</tr>
<tr>
<td>If $i_{val} = \bot$ then $i_{val} := i$</td>
</tr>
<tr>
<td><strong>out</strong>(w)$_R$ Pre</td>
</tr>
</tbody>
</table>
| $x_{val}, i_{val} \neq \bot$
| $w = x_{val}(i_{val})$ |
| **Effect** none |

#### Transitions

- **wait**
  - $\text{in}(x)_T, \text{in}(i)_R$
- **output**
  - $\text{out}(x(i))_R$
The Protocol

Tdp: trap-door permutation
D: domain of Tdp
B: hard-core predicate for Tdp

Sender(x)

p := R Tdp
send(1,p)

rec(2,z)
bval(0):= B(p^{-1}(z(0))) \oplus x(0)
bval(1):= B(p^{-1}(z(1))) \oplus x(1)
send(3,bval)

Receiver(i)

y := R \{0,1\} \rightarrow D
rec(1,p)
zval(i):= p(y(i))
zval(1-i) := y(1-i)
send(2,zval)

rec(3,b)
w := b(i) \oplus B(y(i))
out(w)

b(i) \oplus B(y(i)) = B(p^{-1}(z(i))) \oplus x(i) \oplus B(y(i)) = B(y(i)) \oplus x(i) \oplus B(y(i)) = x(i)
Real Protocol

wait
  in(x)
  rand(p)
send(1,p)
wait
  rec(2,z)

\[ bval(0) := B(p^{-1}(z(0))) \oplus x(0) \]
\[ bval(1) := B(p^{-1}(z(1))) \oplus x(1) \]
send(3,bval)

wait
  in(i)
  rec(1,p)
  rand(y)

zval(i) := p(y(i))
zval(1-i) := y(1-i)
send(2,zval)
wait
  rec(3,b)

w := b(i) \oplus B(y(i))
out(w)

\[ \text{Trans}(D,Tdp) \]
\[ \text{Rec}(D,Tdp) \]
\[ \text{Src}_{pval}^{Tdp} \]
\[ \text{Src}_{yval}^{D} \]
\[ \text{Env} \]
Ideal Protocol with Simulator

\[\text{Ideal} \quad \text{Sim} \quad \text{Adv} \quad \text{Tdp} \quad \text{D} \quad \text{Src}\_pval \quad \text{Src}\_zval \quad \text{Src}\_bval\]

- \text{wait} \quad \text{rand}(p) \quad \text{rand}(z) \quad \text{rand}(b)
- \text{send}(1,p) \quad \text{send}(2,z) \quad \text{send}(3,b)
- \text{Env} \quad \text{in}(i) \quad \text{in}(x) \quad \text{out}(w)
- \text{Ideal} \quad \text{send}(m)_{T} \quad \text{send}(m)_{R}
- \text{Adv} \quad \text{rec}(m)_{T} \quad \text{rec}(m)_{R}
- \text{Sim} \quad \text{rand}(b) \quad \text{rand}(1) \quad \text{rand}(0)
What we should Prove

Objective:

Env should not distinguish real from ideal
Let Env have a special accept action

≤ \text{neg,pt}

for each PPT environment Env
for each trace distribution of Real | Env
there exists a trace distribution of Ideal | Env
the probabilities of accept differ by a negligible value
Implementation Relation
Extends Computational Indistinguishability

- Families of probabilistic automata
  - Indexed by security parameter $k$
- Time bounded automata (by some polynomial $p$)
  - Elements representable with $p(k)$ bits
  - Elements computable in time $p(k)$

\[
\{A_k\} \leq_{neg,pt} \{B_k\} \text{ iff } \\
\begin{align*}
\text{For each polynomial } p, p_1, p_2 & \\
\text{There exists a polynomial } p_2 & \\
\text{There exists a function } \varepsilon \text{ negligible in } k & \\
\text{For each Environment } \{E_k\} & \\
\quad \text{p-bounded} & \\
\quad \text{with special action accept} & \\
\text{For each trace distribution of } A_k|E_k \text{ of length at most } p_1(k) & \\
\text{There exists a trace distribution of } B_k|E_k \text{ of length at most } p_2(k) & \\
\quad \text{Probabilities of accept differ at most by } \varepsilon(k) & \left(\forall c \exists k \forall k > k\right) \\
\end{align*}
\]
Hard Core Predicate
Trap-door permutation

- Domain $D = \{D_k\}$
- Trap-door permutation $Tdp = \{Tdp_k\}$
- Hard-core predicate $B : \{D_k \rightarrow \{0,1\}\}$
  - Poly-time computable
  - For each poly-time predicate $G$ there exists negligible $\varepsilon$

\[
\begin{align*}
\Pr \left[ \begin{array}{l}
  f \leftarrow Tdp_k; \\
  z \leftarrow D_k \\
  b \leftarrow B(f^{-1}(z)); \\
  G_k(f,z,b) = 1
\end{array} \right] - \Pr \left[ \begin{array}{l}
  f \leftarrow Tdp_k; \\
  z \leftarrow D_k \\
  b \leftarrow \{0,1\}; \\
  G_k(f,z,b) = 1
\end{array} \right] & \leq \varepsilon(k)
\end{align*}
\]
Hard-Core Predicate Definition as Implementation

\[ H : \text{wait } \text{rand}(p)p_{\text{val}}, \text{rand}(y)y_{\text{val}} \]
\[ z := p(y) \]
\[ b := B(y) \]
\[ \text{output } \text{rand}(z)z_{\text{val}}, \text{rand}(b)b_{\text{val}} \]
Playing with Hard-Core Predicates

On the use of Probabilistic Automata for Security Proofs
Atagawa, April 6-9 2009

Roberto Segala - University of Verona
Playing with Hard-Core Predicates

Hide_{SH2} \rightarrow \text{SH2} \rightarrow \text{Ifc} \rightarrow \text{SHOT} \rightarrow \leq_{\text{neg,pt}}

Hide_{SH2} \rightarrow \text{SH2} \rightarrow \text{Ifc} \rightarrow \text{SHOT}

\text{Ifc:}
- \text{wait} in(x)
- r_p,r_{z0},r_{b0},r_{z1},r_{b1}
- b(0) := x(0) \oplus b_0
- b(1) := x(1) \oplus b_1
- z(0) := z_0
- z(1) := z_1
- send(1,p)
- send(2,z)
- send(3,b)
Ideal Protocol with Intermediate Simulator 1

wait
rand(p), rand(z), in(x)
b(0):=B(p^{-1}(z(0))) \oplus x(0)
b(1):=B(p^{-1}(z(1))) \oplus x(1)
send(1,p)
send(2,z)
send(3,b)
Real Protocol

\[
\begin{align*}
\text{wait} & \quad \text{in}(x) \\
\text{rand}(p) & \\
\text{send}(1,p) & \\
\text{wait} & \quad \text{rec}(2,z) \\
bval(0) & \equiv B(p^{-1}(z(0))) \oplus x(0) \\
bval(1) & \equiv B(p^{-1}(z(1))) \oplus x(1) \\
\text{send}(3,bval) & \\
\text{wait} & \\
in(i) & \\
\text{rec}(1,p) & \\
\text{rand}(y) & \\
zval(i) & \equiv p(y(i)) \\
zval(1-i) & \equiv y(1-i) \\
\text{send}(2,zval) & \\
\text{wait} & \\
\text{rec}(3,b) & \\
w & \equiv b(i) \oplus B(y(i)) \\
\text{out}(w) & \\
\text{out}(w) & \\
in(i) & \in(x) \\
\text{rand}(p) & \\
\text{send}(1,p) & \\
\text{wait} & \quad \text{rec}(2,z) \\
bval(0) & \equiv B(p^{-1}(z(0))) \oplus x(0) \\
bval(1) & \equiv B(p^{-1}(z(1))) \oplus x(1) \\
\text{send}(3,bval) & \\
\text{wait} & \\
in(i) & \\
\text{rec}(1,p) & \\
\text{rand}(y) & \\
zval(i) & \equiv p(y(i)) \\
zval(1-i) & \equiv y(1-i) \\
\text{send}(2,zval) & \\
\text{wait} & \\
\text{rec}(3,b) & \\
w & \equiv b(i) \oplus B(y(i)) \\
\text{out}(w) & \\
\text{out}(w) & \\
in(i) & \in(x) \\
\text{rand}(p) & \\
\text{send}(1,p) & \\
\text{wait} & \quad \text{rec}(2,z) \\
bval(0) & \equiv B(p^{-1}(z(0))) \oplus x(0) \\
bval(1) & \equiv B(p^{-1}(z(1))) \oplus x(1) \\
\text{send}(3,bval) & \
\end{align*}
\]
The Proof

wait
  rand(p), rand(z), rand(c), in(x)
b(0):=c(0) ⊕ x(0)
b(1):=c(1) ⊕ x(1)
send(1,p)
send(2,z)
send(3,b)

wait
  in(x)
  rand(p)
send(1,p)
wait
  rec(2,z)
bval(0):=B(p⁻¹(z(0))) ⊕ x(0)
bval(1):=B(p⁻¹(z(1))) ⊕ x(1)
send(3,bval)

wait
  in(i)
  rec(1,p)
  rand(y)
zval(i):= p(y(i))
zval(1-i) := y(1-i)
send(2,zval)
wait
  rec(3,b)
w := b(i) ⊕ B(y(i))
out(w)
Ideal Protocol with Intermediate Simulator 1

wait
rand(p), rand(z), in(x)

 Env

Ideal

 b(0) := B(p^{-1}(z(0))) \oplus x(0)
 b(1) := B(p^{-1}(z(1))) \oplus x(1)

 send(1, p)
 send(2, z)
 send(3, b)

 Adv

 TransRec_1(D, Tdp)

 src_{pval}^{Tdp}

 rand(p)

 src_{zval}^{D}

 rand(z)

 send(m)_T \rightarrow rec(m)_T \leftarrow send(m)_R \rightarrow rec(m)_R
Playing with Hard-Core Predicates

\[
\text{Ifc:} \begin{align*}
\text{wait} & \quad \text{in}(x) \\
s & \quad r_p, r_{z0}, r_{b0}, r_{z1}, r_{b1} \\
b(0) & \quad := x(0) \oplus b_0 \\
b(1) & \quad := x(1) \oplus b_1 \\
z(0) & \quad := z_0 \\
z(1) & \quad := z_1 \\
\text{send}(1, p) & \\
\text{send}(2, z) & \\
\text{send}(3, b) & 
\end{align*}
\]

\[
\text{Hide}_{\text{SH2}} \quad \text{SH2} \quad \text{Ifc} \quad \text{SHROT}
\]

\[
\text{SH2} \quad \text{Ifc} \quad \text{SHOT}
\]

\[
\leq_{\text{neg,pt}}
\]
The Proof

\[ \text{Env} \]

\[ \text{TransRec}(D, Tdp) \]

\[ \text{Adv} \]

\[ \text{Ideal} \]

\[ \text{Src}_{pval}^{Tdp}, \text{Src}_{zval} \]

\[ \{0,1\}, \text{Src}_{yval} \]

\[ \leq 0 \]

\[ \text{SH2} \]

\[ \text{Ifc} \]
Ideal Protocol with Intermediate Simulator 2

$$\text{TransRec}_2(D, Tdp)$$

wait
- rand(p), rand(z), rand(c), in(x)
- b(0) := c(0) \oplus x(0)
- b(1) := c(1) \oplus x(1)
- send(1, p)
- send(2, z)
- send(3, b)

Adv
- send(m)_T
- send(m)_R
- rec(m)_T
- rec(m)_R

Env
- in(x)
- in(i)
- out(w)

Ideal
- out(w)

$$\text{Src}_{pval}^{Tdp}$$
- rand(p)

$$\text{Src}_{zval}^{D}$$
- rand(z)

$$\text{Src}_{cval}^{\{0,1\}}$$
- rand(c)
On the use of Probabilistic Automata for Security Proofs
Atagawa, April 6-9 2009

Roberto Segala - University of Verona

68
The Proof

On the use of Probabilistic Automata for Security Proofs
Atagawa, April 6-9 2009

Roberto Segala - University of Verona
Problems with Nondeterminism
Ideal Protocol with Simulator

if $x(1-i) = 0$
  schedule $send(m)_T \rightarrow rec(m)_T \rightarrow send(m)_R$
else
  schedule $send(m)_T \rightarrow send(m)_R \rightarrow rec(m)_T$

Adv learns $x(1-i)$
by ending in different states
Adv communicates $x(1-i)$ to Env
Problems with Nondeterminism

- Order of messages may reveal one bit of $s$ to $E$
Approaches to Nondeterminism

- **UC framework**
  - ITMs have a token passing mechanism
  - No nondeterminism

- **Reactive simulatability**
  - Again token passing mechanism (implicit)
  - Nondeterminism based on local information only

- **Symbolic Dolev-Yao**
  - No probability
  - Symbols hide information

- **Process Algebras**
  - Scheduler sees only enabled action type

- **Switched PIOAs**
  - Token passing mechanism (explicit)
  - Nondeterminism based on local information only

- **Task PIOAs**
  - Define equivalence classes of actions
  - Scheduler sees only equivalence classes, not elements

- **Careful specifications**
  - Avoid dangerous nondeterminism in the specification
  - Is it always possible?
Task PIOAs

- Probabilistic I/O Automata with ...
  - Action determinism
    - For each action at most one transition enabled
  - Output and internal actions partitioned into tasks
  - Task determinism
    - For each task at most one transition enabled

- A scheduler is a sequence of tasks
  - Upon scheduling a task from a state \( q \)
    - Automaton performs unique transition enabled if it exists
    - Automaton idles if task not enabled

- Essentially scheduling does not depend on secret info
Task PIOAs What???

- Scheduler are oblivious
  - Not quite
  - We can encode the token passing mechanism
  - We could elect an automaton as adversary

- Do simulations continue to work?
  - We have to change the step condition
    - A task should be matched by a task
    - A simulation relates measures over executions
      - Need to know what tasks induced the measure

- Can we do better?
  - We do not know
  - But tasks work better than we expected
  - We can generalize them in many simple ways
  - Yet it would be nice to find something less “oblivious”
Case Study:

Agent Authentication

Bellare Rogaway 93

Segala, Turrini
Bellare and Rogaway MAP1 Protocol

- Nonces are generated randomly
- The key \( s \) is the secret for a Message Authentication Code
  - Specifically, MAC based on pseudo-random functions
Nonces

- **Number ONCE**
  - Typically drawn randomly

- **Claim**
  - For each constant $c$ and polynomial $p$
  - There exists $k$ such that for each $k \geq k$
  - If $n_1, n_2, \ldots, n_{p(k)}$ are random nonces from $\{0,1\}^k$
  - Then $\Pr[\exists i \neq j, n_i = n_j] < k^{-c}$
Message Authentication Code

• Triple \((G,A,V)\)
  - \(G\) on input \(1^k\) generates \(s \in \{0,1\}^k\)
  - For each \(s\) and each \(a\)
    - \(\Pr[V(s,a,A(s,a))=1]=1\)

• Forger
  - On input \(1^k\) obtains MAC of strings of its choice
  - Outputs a pair \((a,b)\)
  - Successful if \(V(s,a,b)=1\) and \(a\) different from previous queries

• Secure MAC
  - Every feasible forger succeeds with negligible probability
MAP1: Matching Conversations

- Matching conversation between A and B
  - Every message from A to B delivered unchanged
    - Possibly last message lost
    - Response from B returned to A
  - Every message received by A generated by B
    - Messages generated by B delivered to A
    - Possibly last message lost

- Correctness condition
  - Matching conversation implies acceptance
  - Negligible probability of acceptance without matching conversation
MAP1: Correctness Proof

- Let $A$ be a PPT machine that interacts with the agents

- Show that $A$ induces “no-match” with negligible probability
  - Argue that repeated nonces occur with negligible probability
  - Argue that $A$ is an attack against a message authentication code

- Features
  - Relies on underlying pseudo-random functions
  - Proves correctness assuming truly random functions
  - Builds a distinguisher for PRFs if an attack exists

- Criticism
  - The arguments are semi-formal and not immediate
  - Three different concepts intermixed
    - Nonces
    - Message authentication codes
    - Matching conversations
MAP1: Hierarchical Analysis

- Agents indexed by $X, Y, t$
- Need to find suitable simulations
  - Step conditions lead to local arguments
  - Yet transitions cannot be matched exactly
Nonce Generators

- **State**
  - \(\text{value}_{X,Y,t}\) **initially** \(\perp\)
  - \(\text{FreshNonces}\) **initially** \(\{0,1\}^k\)

- **Transitions**
  - **Input** \(\text{NonceRequest}_{X,Y,t}\)
  - **Effect**
    - Let \(v \in R \{0,1\}^k\)
    - \(\text{value}_{X,Y,t} = v\)
    - \(\text{FreshNonces} = \text{FreshNonces} - \{v\}\)

  - **Output** \(\text{NonceResponse}_{X,Y,t}(n)\)
  - **Precondition**
    - \(n = \text{value}_{X,Y,t}\)
  - **Effect**
    - \(\text{value}_{X,Y,t} = \perp\)
Adversary

- **Keeps a variable** *history*
  - Holds all previous messages

- **Real adversary**
  - Runs a cycle where
    - Computes the next message to send using a PPT function $f$
    - Sends the message
    - Waits for the answer if expected

- **Ideal adversary**
  - Highly nondeterministic
  - Stores all input
  - Sends messages that do not contain forged authentications
Problems with Simulations

• Problem
  - Consider a transition of the real nonce generator
  - With some probability there is a repeated nonce
  - The ideal nonce generator does not repeat nonces
  - Thus, we cannot match the step

• Solution
  - Match transitions up to some error
Approximate Simulations [ST07]

- Change equivalence on measures

\[ \mu_1 \equiv_\varepsilon \mu_2 \iff \]
\[ \mu_1 = (1-\varepsilon)\mu_1' + \varepsilon\mu_1'' \]
\[ \mu_2 = (1-\varepsilon)\mu_2' + \varepsilon\mu_2'' \]
\[ \mu_1' \equiv \mu_2' \]

\[
\begin{array}{c}
\{2/3 \, q_1, 1/3 \, q_2\} = 2/3 \{1/2 \, q_1, 1/2 \, q_2\} + 1/3\{1 \, q_1\} \\
{1/3 \, s_1, 1/3 \, s_2, 1/3 \, s_3} = 2/3 \{1/2 \, s_1, 1/2 \, s_2\} + 1/3\{1 \, s_3\}
\end{array}
\]
Approximate Simulations

\{A_k\} \{R_k\} \{B_k\}

- For each constant \(c\) and polynomial \(p\)
- There exists \(k\) such that for each \(k \geq k\)
- Whenever
  - \(v_1\) reached within \(p(k)\) steps in \(A_k\)
  - \(v_1 L(R_k,\gamma) v_2\)
  - \(v_1 \rightarrow v_1'\)
- There exists \(v_2'\) such that
  - \(v_2 \rightarrow v_2'\)
  - \(v_1' L(R_k,\gamma+k-c) v_2'\)
Approximate Simulations
Step Condition

\[ \nu_2' \equiv \nu_1' \]

\[ \nu_2 \rightarrow (1-\gamma) \gamma \]

\[ \nu_1 \rightarrow (1-\gamma) \gamma \]

\[ \nu_2' \rightarrow (1-\gamma-k\cdot c) k\cdot c \gamma \]

\[ \nu_1' \rightarrow (1-\gamma-k\cdot c) k\cdot c \gamma \]
Simulation Implies Behavioral Inclusion

• The step condition can be applied repeatedly

\[
\begin{align*}
S & \xrightarrow{0} \rho_1 \xrightarrow{k^c} \rho_2 \xrightarrow{2k^c} \rho_3 \xrightarrow{3k^c} \cdots \xrightarrow{p(k)k^c} \rho_{p(k)} \\
q & \xrightarrow{} \mu_1 \xrightarrow{} \mu_2 \xrightarrow{} \mu_3 \xrightarrow{} \cdots \xrightarrow{} \mu_{p(k)}
\end{align*}
\]

• Observation
  – \( p(k)k^c \) can be smaller than any \( k^{c'} \) by choosing \( c = c' + \text{degree}(p) \)
Execution Correspondence under Approximated Simulations

If \( \{A_k\} \{R_k\} \{B_k\} \) then

- For each constant \( c \) and polynomial \( p \)
- There exists \( k \) such that for each \( k \geq k \)
- For each scheduler \( \sigma_1 \)
  - \( \nu_1 \) reached within \( p(k) \) steps in \( A_k \) with \( \sigma_1 \)
- There exists \( \sigma_2 \) such that
  - \( \nu_2 \) reached within \( p(k) \) steps in \( B_k \) with \( \sigma_2 \)
  - \( \nu_1 L(R_k d(p(k)k^{-c})) \nu_2 \)

- Observation
  - \( p(k)k^{-c} \) can be smaller than any \( k^{-c'} \) by choosing \( c=c'+\text{degree}(p) \)
Example: Approximate Simulations
Bellare-Rogaway MAP1 Protocol

- **Negation of the step condition**
  - 1: Two random nonces are equal with high probability
  - 2: Function f defines a forger for a signature scheme
Negation of Step Condition

\[ \{A_k\} \{R_k\} \{B_k\} \]

- There exists constant \( c \) and polynomial \( p \)
- For each \( k \) there exists \( k \geq k \)
- There exists
  - \( \nu_1 \) reached within \( p(k) \) steps in \( A_k \)
  - \( \nu_1 L(R_k,\gamma) \nu_2 \)
  - \( \nu_1 \rightarrow \nu_1' \)
- There is no \( \nu_2' \) such that
  - \( \nu_2 \rightarrow \nu_2' \)
  - \( \nu_1' L(R_k,\gamma+k^c) \nu_2' \)

- Signature forged in \( \nu_1' \)
  - Probability at least \( k^{-c} \)
- Nonce replicated in \( \nu_1' \)
  - Probability at least \( k^{-c} (1-\gamma) \)

\( \nu_1 \rightarrow \nu_1' \)
\( \nu_2 \rightarrow \nu_2' \)
\( \nu_1' L(R_k,\gamma+k^c) \nu_2' \)

\( \gamma \rightarrow \gamma + k^{-c} \)
Nonces

- Number ONCE
  - Typically drawn randomly

- Claim
  - For each constant $c$ and polynomial $p$
  - There exists $k$ such that for each $k \geq k$
  - If $n_1, n_2, \ldots, n_{p(k)}$ are random nonces from $\{0, 1\}^k$
  - Then $\Pr[\exists_{i \neq j} n_i = n_j] < k^{-c}$
Problems with Nondeterminism

MAP1 Protocol [BR93]

• Authentication protocol
  - Symmetric key signature schema
  - Computational Dolev-Yao
  - Adversary queries agents

• Potential problems
  - Let $s$ be the shared key
  - Adversary queries $k$ agents
  - Agent $i$ replies if $i^{th}$ bit of $s$ is 1
  - The adversary knows the shared key

• Solution
  - One query at a time
  - Wait for the answer (agents as oracles)
More About Approximated Simulations
Conditional Automata

- Let $A$ be a probabilistic automaton
- Let $B$ be a set of bad states
- Let $G = Q - B$ be a set of good states

- Let $A|G$ be the same as $A$ except that
  - $D_{A|G} = \{(q,a,\mu | G) : (q,a,\mu) \in D_A \text{ and } \mu(G) > 0\}$

Theorem:

$id_Q$ is a polynomially accurate simulation from $A$ to $A|G$
iff $B$ is negligible
$id_Q$ is a polynomially accurate simulation from $A|G$ to $A$
iff $B$ is negligible
A Property of Approximated Lifting

Given a relation $R$ from $Q_1$ to $Q_2$

Then $\mu_1 \text{L}(R,\varepsilon) \mu_2$ iff there exists

$$w: Q_1 \rightarrow Q_2 \rightarrow [0,1]$$
- $w$ supported on $R$
- $w(Q_1,Q_2) = 1-\varepsilon$
- $w(s,Q_2) \leq \mu_1(a)$
- $w(Q_1,s) \leq \mu_2(a)$
Approximated Correspondence

This means that ...

On the use of Probabilistic Automata for Security Proofs
Atagawa, April 6-9 2009
Roberto Segala - University of Verona

97
Transitivity

Claim. \( \mu L(R, \varepsilon) \rho \) and \( \rho L(R', \tau) \eta \) imply \( \mu L(RR', \varepsilon + \tau) \eta \)
Are approximated simulations transitive?

- We do not know
  - ... but the result of the previous slide suffices
Are Approximated Simulations Compositional?

No. Need a more refined relation.

\[ s S(R, \varepsilon) q \text{ iff } \forall q, s, a, \mu' \exists \sigma' \]

\[
\begin{array}{c}
  s \quad a \\
  \quad \downarrow R \varepsilon \\
  q \quad a \\
  \quad \downarrow \quad \mu'
\end{array}
\]

Step condition

For each \( c \) there exists \( k \)
For each \( k > k \), each \( \mu_1, \mu_2, \gamma, w \)

If \( \mu_1 L(R_k, \gamma) \mu_2 \) via \( w \)
then
\[
\Sigma \{ w(q_1,q_2) : q_1 \text{ not}(S(R_k,\epsilon^c)) q_2 \} < k^{-c}
\]

Conditional automata continue to work
How About Weak Relations?

- Only one constraint to add
  - Length of matching steps bounded
    - By a constant
    - By a polynomial on length of history
Case Study:

Dolev-Yao Soundness

Cortier Warinschi 04

Segala, Turrini
Protocol Syntax

• Sorts
  - SKey, VKey, EKey, DKey
  - Id, Nonce, Label, Ciphertext, Signature, Pair
  - Term: superset that includes all others
    • Labels should be left out

• Operators
  - \langle _, _ \rangle: Term \times Term \rightarrow Pair
  - \{\}_-, _: EKey \times Term \times Label \rightarrow Ciphertext
  - []_-,: SKey \times Term \times Label \rightarrow Signature

• Variables
  - Sorted variables
  - X = X.n \cup X.a \cup X.c \cup X.s \cup X.l
  - X.a = \{A_1, A_2, ..., A_n\}, n number of protocol participants
  - X.n = \bigcup_{A \in X.a} \{X_{A,j} | j \in \mathbb{N}\}
Protocol Syntax

- **Roles**
  - Finite sequence of rules
  - $((\text{init}) \times T_{\Sigma}(X)) \times (T_{\Sigma}(X) \times \{\text{stop}\})^*$

- **k-party protocol**
  - $\Pi : \{1, \ldots, k\} \rightarrow \text{Roles}$
  - $\Pi(i)$ is the program of player $i$

- **Idea**
  - An adversary instantiates protocols and queries parties
  - If role $i$ is ready to execute the pair $(l,r)$ and role $i$ is given input $m$
  - $m$ is parsed according to $l$
    - Pattern matching, unification
    - Some variables may be bound to new values
  - $r$ is returned as a result
Example: Needham-Schroeder-Lowe

\[ A \rightarrow B : \{Na,A\}_{ek(B)} \]
\[ B \rightarrow A : \{Na,Nb,B\}_{ek(A)} \]
\[ A \rightarrow B : \{Nb\}_{ek(B)} \]

\[ \Pi(1) = (init, \quad \{XA_{1,1}, XA_{2,1}, A_2\}_{ek(A_1), L}, \quad \{X_{A_{1,1}, A_1}\}_{ek(A_2), ag(1)}) \]
\[ \Pi(2) = (\{XA_{1,1}, A_1\}_{ek(A_2), L_1}, \quad \{X_{A_{2,1}}\}_{ek(A_2), ag(1)}) \]
\[ \quad \{XA_{1,1}, XA_{2,1}, A_2\}_{ek(A_1), ag(1)} \quad \text{stop} \]
Formal Execution Model

• **Messages are ground terms from an algebra**
  - $T ::= N | a | ek(a) | dk(a) | sk(a) | vk(a) | n(a,j,s) | \langle T,T \rangle | \{T\}_{ek(a),ag(i)} | \{T\}_{ek(a),adv(i)} | [T]_{sk(a),ag(i)} | [T]_{sk(a),adv(i)}$

• **Global state**: $(S\text{Id}, f, H)$
  - $S\text{Id}$: set of session Ids of the form $(n,j,(a_1,\ldots,a_k))$
  - $f$: associates state $(\sigma,i,p)$ to each session id
    - Partial function $\sigma$ associates terms to variables
    - $i$ is the role being executed
    - $p$ is the program counter (next pair to match)
  - $H$ is a set of terms (knowledge of adversary)
Formal Execution Model

- Initially no session ids, $H$ contains nonces of adversary

- Transitions
  - $\text{corrupt}(a_1,\ldots,a_l)$
    - $H$ updated with knowledge of $a_1,\ldots,a_l$
  - $\text{new}(i,(a_1,\ldots,a_k))$
    - New session id $S$ created with index $s$
    - $f(S) = (\sigma, i, 1)$
    - Function $\sigma$ binds agent variable $A_j$ to $a_j$
    - Function $\sigma$ binds nonce variable $X_{A_i,j}$ to $n(a_i,j,s)$
  - $\text{send}(S,t)$
    - Let $f(S)$ be $(\sigma, i, p)$ and let $(l, r)$ be the $p^{th}$ pair of $\Pi(i)$
    - Match $t$ with $l$ updating $\sigma$. Stop if unsuccessful.
    - Compute $r$ and add it to $H$
    - Update $f(S)$ to $(\sigma, i, p+1)$

Restriction:
$t$ must be DY-deducible from $H$
Concrete Execution Model

- Agent id’s, nonces, messages are **bitstrings**
- Security parameter $\nu$ identifies lengths

- **Global state**: $(S\text{Id},g,H)$
  - $H$ is the knowledge of the adversary
  - $S\text{Id}$: set of session Ids of the form $(\eta, j, (\eta_1, ..., \eta_k))$
  - $g$: associates state $(\tau, i, p)$ to each session id
    - Partial function $\tau$ associates **bitstrings** to variables
    - $i$ is the role being executed
    - $p$ is the program counter (next pair to match)
Concrete Execution Model

- Initially no session ids
- Transitions
  - `corrupt(\eta_1, \ldots, \eta_l)`
    - \(H\) updated with knowledge of \(\eta_1, \ldots, \eta_l\)
    - The necessary missing keys are generated
  - `new(i, (\eta_1, \ldots, \eta_k))`
    - New session id \(S\) created with index \(s\)
    - \(g(S) = (\tau, i, 1)\)
    - Function \(\tau\) binds agent variable \(A_j\) to \(\eta_j\)
    - Function \(\tau\) binds nonce variable \(X_{Ai,j}\) to random bitstrings
    - Random coins are flipped for the randomization of encryption and signature
  - `send(S, t)`
    - Let \(g(S) = (\tau, i, p)\) and let \((l, r)\) be the \(p^{th}\) pair of \(\Pi(i)\)
    - Match \(t\) with \(l\) updating \(\tau\). Stop if unsuccessful.
      - May need to decrypt and verify signatures
    - Compute \(r\) and add it to \(H\)
      - May need to encrypt and sign
    - Update \(g(S)\) to \((\sigma, i, p+1)\)
Computations of Concrete Model

• In the model of [CW04]
  - Choice of transitions by PPT adversary
  - Length of computations bounded by a polynomial
  - Number of needed random bits known in advance
  - Unique computation for each value of the random bits
  - This induces a probability measure on computations

• With Probabilistic Automata
  - Random bits generated within transitions
  - Avoid reasoning about guessing future random bits
    • ... though in [CW04] this reasoning is not present
Correspondence Between Computations

• Let $c$ be a mapping from ground terms to bitstrings
• Let $s = (S\text{Id},f,H)$ be a state of the formal model
• Let $t = (C\text{Id},g,H')$ be a state of the concrete model
• Define $s \equiv_c t$ iff
  - $C\text{Id} = \{c(S) \mid S \in S\text{Id}\}$
  - $\forall S \in S\text{Id} \ g(c(S)) = c(f(S))$
• Where
  - $c(n,i,(a_1,\ldots,a_k)) = (n,i,(c(a_1),\ldots,c(a_k)))$
  - $c(s,i,p) = (c(s),i,p)$
• Define $s_0s_1\ldots s_l \equiv t_0t_1\ldots t_l$ iff
  - $\exists c$ injective $\forall j \ s_j \equiv_c t_j$
• Concrete model safe iff
  - For each measure $\mu$ on concrete executions induced by random coins
  - $\mu(\{a \mid \exists b \ a \equiv b\})$ is overwhelming
Structure of Original Proof

- Prove properties of DY-non-deducibility
  1. Signature forged, or
  2. Encrypted data used without decrypting

- Fix random coins and get concrete execution $\alpha$

- Show $\alpha$ is instantiation of some symbolic execution $\beta$
  - Follow $\alpha$ building $\beta$ and mapping bitstrings to abstract terms
    - How do I know the mapping exists?
      - Example: reencrypt a message with a different label and encryptions are the same
    - Let $c$ be the inverse of the mapping above
      - How do I know the mapping is invertible?
        - Example: forward an encrypted message
      - How do I know $c$ is injective?
        - The inverse of a mapping is injective

- Show $\beta$ follows DY-deducibility with overwhelming probability
  - If not, then either 1 or 2 with non-negligible probability
  - Build attacker to corresponding primitive
Properties of non-DY-Deducibility

- Let $S$ be a set of messages and $m$ a message such that
  - $S \not\models m$
  - $m$ built out of atoms of elements in $S$

- Then either
  - There exists subterm $[t]_k$ of $m$ which is not a subterm of terms in $S$, or
  - There exists a subterm $t$ of $m$ such that
    - all its super-terms in $m$ are not deducible
    - $t$ appears encrypted in $S$

- Problem
  - A message that contains atoms not in $S$ is not deducible
  - Scenario not included in the cases above
Structure of the Proof with Probabilistic Automata

Now $c$ is injective

ING | SIG | CS are a PPT
Environment for ENC

ING | ENC | CS are a PPT
Environment for SIG

Here we have also function $c$, though not injective

Actions chosen by PPT function $f$
Primitives solved by NG, ENC, SIG
Problems Encountered
Concrete Model

- **Explicit encoding of**
  - Parsing of left expression
  - Computation of right expression
  - Invocations to cryptographic primitives

- **What arguments are needed for and computed by ...**
  - Left parsing
  - Right computation

- **Answer**
  - The mapping $\tau$
Concrete Model: Some examples

• \((\text{init}, X_{A1,1}) (\{X_{A2,1}\}_{ek(a)}, L, \{X_{A2,1}\}_{ek(a)}, ag(1)) (X_{A2,2}, \text{stop})\)
  - After initialization \(\tau(X_{A1,1}) = \eta_1\)
  - Upon receiving a bitstring \(\eta_2\)
    • It is decrypted with \(dk(a_1)\) and \(\tau(X_{A2,1}) = \eta_3\)
    • What should \(L\) be mapped to?
    • Then \(\eta_3\) is encrypted with \(ek(a_1)\) leading to \(\eta_4\)
  - Upon receiving \(\eta_5\), \(\tau(X_{A2,2}) = \eta_5\) and terminate

• \((\text{init}, X_{A1,1}) (\{X_{A2,1}\}_{ek(a)}, L, \{X_{A2,1}\}_{ek(a)}, L) (X_{A2,2}, \text{stop})\)
  - After initialization \(\tau(X_{A1,1}) = \eta_1\)
  - Upon receiving a bitstring \(\eta_2\)
    • It is decrypted with \(dk(a_1)\) and \(\tau(X_{A2,1}) = \eta_3\)
    • Then \(\eta_3\) is encrypted with \(ek(a_1)\) leading to \(\eta_4\)
  - Upon receiving \(\eta_5\), \(\tau(X_{A2,2}) = \eta_5\) and terminate
Structure of the Proof with Probabilistic Automata

Now c is injective
ING | SIG | CS are a PPT
Environment for ENC

ING | ENC | CS are a PPT
Environment for SIG

Actions chosen by PPT function f
Primitives solved by NG, ENC, SIG

In the diagram, the action c is injective, and ING, SIG, ENC are PPT primitives.
Problems Encountered
Definition of $C + S$

- If the bitstring I receive does not parse what symbolic message should I use?
  - Not said/considered in the original proof

- The bitstring should be kept, though
  - A real system could reuse it later

- Our solution
  - Use a special symbol ⊥
  - Its meaning is that we are sending junk
  - Function $c$ does not map ⊥
Consequences of our Solution

• All the symbols we use in send actions are built from atomic terms that appear in the history.

• The new statement about non-deducibility suffices
  - Do not need to worry about guessing the future.

ING  IENC  SIG  C + S
Structure of the Proof with Probabilistic Automata

now $c$ is injective

$\text{ING} \mid \text{SIG} \mid \text{CS}$ are a PPT

Environment for $\text{ENC}$

$\text{ING} \mid \text{ENC} \mid \text{CS}$ are a PPT

Environment for $\text{SIG}$

Here we have also function $c$, though not injective

Actions chosen by PPT function $f$

Primitives solved by $\text{NG}$, $\text{ENC}$, $\text{SIG}$
Summing Up …

- What we have seen
  - A theory of Probabilistic Automata
    - Conservative extension of automata
    - Language inclusion
    - Simulation relations
    - Hierarchical compositional reasoning
  - A notion of task PIOA with restricted schedulers
    - Task equivalence relation on states
    - Action deterministic
    - At most one action for each task
    - A schedule (sequence of tasks) determines a probabilistic execution
  - A notion of approximated language inclusion
    - For each trace distribution of A there exists an indistinguishable trace distribution of B
  - A notion of approximated simulation
    - Works for PAs
Summing Up …

… what we have seen

• Analysis of oblivious transfer in UC framework
  - Task PIOAs as model
  - Hierarchical verification via simulations
  - Crypto-steps via approximated language inclusion

• Analysis of MAP1 protocol
  - PAs as model
  - Approximated simulations as technique
  - Mixture of Dolev-Yao and computational models
  - No restriction of nondeterminism
    - Yet accurate description of objects

• Analysys of DY-soundness
  - PAs as model
  - Approximated simulations, hierarchical compositional analysis
  - Easy to find problems … more difficult to fix them
Several Open Questions

• Connections
  - Approximated simulations with
    • Approximated language inclusion
    • Restricted schedulers
  - Semantics
    • Metrics and exact equivalences

• Properties of definitions
  - Are we transitive?
  - Are there weaker compositional refinements?

• Flexibility on restrictions
  - Task PIOAs are very restrictive
    • ... though they work
    • Chatzikokolakis and Palamidessi may help (Concur07)

• Understanding of restrictions
  - Are we restricting too much?

• More case studies
  - Need to understand common points
  - Need to discover missing pieces
A Note about Formal Analysis

• Formal methods are too heavy to use
  - Is it reasonable to apply them all the times?
  - Is it reasonable to use them all the times?
  - Is it reasonable to know them?
  - Are automatic tools everything we need?

• Rarely we can be absolutely rigorous
  - We rather limit the places where to use intuition
  - Formal methods give a lot of sanity checks
  - It is useful to be aware of the formal meaning of what we say
  - It is useful to have theoretical results
    • Some doubts can be eliminated quickly
    • Some bugs may be discovered in a few seconds
Thank You
Convex Combination of Measures

- Let $\mu_1$ and $\mu_2$ be probability measures
- Let $p_1$ and $p_2$ be reals in $[0,1]$ such that $p_1+p_2=1$
- Define a new measure $\mu = p_1\mu_1 + p_2\mu_2$ as follows
  - $\forall X, \mu(X) = p_1\mu_1(X) + p_2\mu_2(X)$

- Theorem: $\mu$ is a probability measure

- Same result extends to countable summation
Weak Transition

There is a probabilistic execution $\mu$ such that

- $\mu(\text{exec}^*) = 1$ (it is finite)
- $\text{trace}(\mu) = \delta(a)$ (its trace is $a$)
- $\text{fstate}(\mu) = \delta(q)$ (it starts from $q$)
- $\text{lstate}(\mu) = \rho$ (it leads to $\rho$)

$q \xrightarrow{a} s \iff \exists \alpha: \text{trace}(\alpha) = a, \text{fstate}(\alpha) = q, \text{lstate}(\alpha) = s$