Formal Verification of Security Protocols using Automata

Roberto Segala
University of Verona
Verification of Security Protocols

Our Question

- Can we use Probabilistic Automata?
  - Hierarchical verification
  - Compositional analysis
  - Simulation method
    - Local arguments to derive global properties
    - Rigorous proofs
  - Potentials for automatic verification
  - Potentials to draw connections to other areas
Probabilistic Automata

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

- Transition relation:
  \[ D \subseteq Q \times (E \cup H) \times \text{Disc}(Q) \]

- Internal (hidden) actions

- External actions:
  \[ E \cap H = \emptyset \]

- Initial state:
  \[ q_0 \in Q \]

- States
Example: Probabilistic Automata

What is the probability of beeping?
Cones and Measures

- **Cone of** \( \alpha \)
  - Set of executions with prefix \( \alpha \)
  - Represent event “\( \alpha \) occurs”
- **Measure of a cone**
  - Product edges of \( \alpha \)

A measure on cones extends uniquely to a measure on the \( \sigma \)-field generated by cones.
Composition of Probabilistic Automata

\[ A_1 = (Q_1, q_1, E_1, H_1, D_1) \]

\[ A_2 = (Q_2, q_2, E_2, H_2, D_2) \]

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

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

\[ D = \left\{ (q, a, \mu_1 \times \mu_2) \mid \begin{array}{ll}
\text{if } a \in E_i \cup H_i & \text{then } (\pi_i(q), a, \mu_i) \in D_i \\
\text{if } a \notin E_i \cup H_i & \text{then } \mu_i = \delta(\pi_i(q)) \\
i \in \{1, 2\} \end{array} \right\} \]
Strong Bisimulation on Probabilistic Automata

Strong bisimulation between $A_1$ and $A_2$

Relation $R \subseteq Q \times Q$, $Q=Q_1 \uplus Q_2$, such that

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

$\mu R \mu'$ [LS89]

$\forall C \in Q/R, \mu(C) = \mu'(C)$
Trace Distributions

The *trace* function is measurable

**Trace distribution of** $\mu$

$tdist(\mu)$: image measure under *trace* of $\mu$

**Trace distribution inclusion preorder**

$A_1 \leq_{TD} A_2$ iff $tdists(A_1) \subseteq tdists(A_2)$
Trace Distribution Precongruence

- Coarsest precongruence included in preorder
- Simulations sound for precongruence
- There are also complete characterizations
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

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

\[
\begin{array}{c}
q_0 \\
q_1 \quad q_2 \\
q_3 \quad q_4 \\
\end{array}
\hspace{1cm}
\begin{array}{c}
s_0 \\
s_1 \\
s_3 \quad s_4 \\
\end{array}
\]

\[
\begin{array}{c}
s \\
q \\
\end{array}
\overset{a}{\longrightarrow}
\begin{array}{c}
s' \\
q' \\
\end{array}
\]

\[
\begin{array}{c}
r \\
R \\
\end{array}
\overset{a}{=} \quad \begin{array}{c}
r \\
R \\
\end{array}
\]
Simulations on Probabilistic Automata

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

\[ \forall q, s, a, \mu' \exists \sigma' \]

Lifting of $R$
The Consensus Problem

- There are $n$ processes
- Each process proposes a value in $\{0,1\}$.
- Each process may decide on a value.
- Processes may fail by stopping.

**Required properties**

**Validity**: decide values that were proposed.

**Agreement**: no different decisions.

**Wait-Free Termination**: each non-failed process decides eventually.
Randomized Consensus

Validity: decide values that were proposed.
Agreement: no different decisions.

Probabilistic Wait-Free Termination:
each non-failed process decides with probability 1.

Aspnes and Herlihy
Termination within expected polynomial time.
Algorithm of Aspnes and Herlihy: Structure

Main protocol

MP

\[ \begin{align*}
1 \\
i \\
n
\end{align*} \]

Coin flipper

CF

\[ \begin{align*}
CF_1 \\
CF_j \\
1 \\
i \\
n
\end{align*} \]

Shared counter

\[ \begin{align*}
\text{inc}_i \\
\text{dec}_i \\
\text{read}_i \\
\text{end-read(c)}_i \\
\text{end-inc}_i \\
\text{end-dec}_i
\end{align*} \]
Algorithm for the Main Protocol

\textbf{Algorithm}

\begin{verbatim}
scan pref and round
if obs-leader(i) and obs-agree(round(i)-1,v)
    then decide(v)
elseif obs-leaders-agree(v)
    then pref(i) = v and round(i) = round(i) + 1
else pref(i) = ⊥
    scan pref and round
    if obs-leaders-agree(v)
        then pref(i) = v and round(i) = round(i) + 1
        else invoke coin flipper to choose a value for pref(i)
        round(i) = round(i) + 1
\end{verbatim}
Coding the Main Protocol

**Read1(k)_i**
- **Pre:** \( pc_i = \text{read1}, \ k \notin \text{obs}_i \)
- **Eff:**
  - \( \text{values}_i[k] \leftarrow \text{pref}(k) \)
  - \( \text{rounds}_i[k] \leftarrow \text{round}(k) \)
  - \( \text{obs}_i \leftarrow \text{obs}_i \cup \{k\} \)
  - If \( \text{obs}_i = \{1,...,n\} \)
    - Then
      - \( pc_i \leftarrow \text{check1} \)

**Check2_i**
- **Pre:** \( pc_i = \text{check2} \)
- **Eff:**
  - If \( \exists v \in \{0,1\} \) \( \text{obs-leader-agree}(v) \)
    - Then
      - \( \text{pref}(i) \leftarrow \text{obs-leader-value} \)
      - \( \text{round}(i) \leftarrow \text{rounds}_i[i]+1 \)
      - \( \text{obs}_i \leftarrow \emptyset \)
      - \( pc_i \leftarrow \text{read1} \)
  - Else
    - \( pc_i \leftarrow \text{flip} \)
Proof of Correctness: Safety

Validity: ordinary invariant proof

\[ \text{agree}(1,v) \text{ and obs-agree}(1,v) \]

Agreement: ordinary invariant proof

\[(\text{obs-agree}(r-1,v) \text{ obs-leader}(i), \text{obs}_i = \{1, \ldots, n\}) \]

implies

\[ \Rightarrow \text{agree}(r,v) \]
Proof of Correctness: Progress

Assume the invocations to the coin flippers on non-failing ports always get an answer (M1)

\[ R \xrightarrow{1} F_0 \cup F_1 \cup D \]

Assume that the all answers at round \( r \) are 0 (where, for \( s \in F_0, \text{max-round}(s) = r \)) (M2)

\[ F_0 \xrightarrow{2} D \]
Assumptions on Coin Flippers

Each coin flipper satisfies the properties

\textbf{C1}: each invocation on a non-failing port gets an answer with probability 1.

\textbf{C2}: fixed a value v in \{0,1\} the probability that all the answers are v is at least p.

We will show that p is independent of n.
Combination of Claims

From $C_1$ and $C_2$ and $M_1$ and $M_2$

\[ R \xrightarrow{1} F_0 \cup F_1 \cup D \]
\[ F_0 \xrightarrow{2} p \ D \]

Combining the statements above

\[ R \xrightarrow{3} p \ D \]

Thus termination within expected $3/p$ rounds.
Let $\mu$ start in a state $s$ of $F_0$.

$\mu$ projection

$\mu(\pi_2^{-1}(C_1 \cap C_2)) \geq p$

$\mu$ inverse image

$\pi_2(\mu)(C_1 \cap C_2) \geq p$

$\pi_1(\pi_2^{-1}(C_1 \cap C_2))$ sat. $M_1 \cap M_2$

$F_0 \xrightarrow{\pi} D$
The Global Picture

MP

1

i

n

CF

CF

CF

inc

dec

read

end-read

end-inc

end-dec

SCT

Shared counter

ACT

Atomic counter

MP

1

i

n

CF

CF

CF

start-flip(j)

ret-flip(v,j)

start-flip(j)

ret-flip(v,j)

start-flip(j)

ret-flip(v,j)

start-flip(j)

ret-flip(v,j)

start-flip(j)

ret-flip(v,j)

start-flip(j)

ret-flip(v,j)
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 $n_1,n_2,...,n_{p(k)}$ 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

Adversary
Keeps history
(PPT function f)

Key generator
Nonce generator (coin flip)

A1 A2 A3 A4 A5

Adversary
Keeps history
(PPT function f)

Key generator
Nonce generator (ideal)

A1 A2 A3 A4 A5

Adversary
Keep history
(no forged signatures)

Key generator
Nonce generator (ideal)

A1 A2 A3 A4 A5
Nonce Generators

- **State**
  - $value_{X,Y,t}$ initially $\perp$
  - $FreshNonces$ initially $\{0,1\}^k$

- **Transitions**
  - **Input** $NonceRequest_{X,Y,t}$
  - **Effect**
    - Let $v \in_R \{0,1\}^k$
    - $value_{X,Y,t} = v$
    - $FreshNonces = FreshNonces - \{v\}$
  - **Output** $NonceResponse_{X,Y,t}(n)$
  - **Precondition**
    - $n = value_{X,Y,t}$
  - **Effect**
    - $value_{X,Y,t} = \perp$

- Ideal Coin flip

Coin flip

- Let $v \in_R FreshNonces$
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**
  
  \[ \mu_1 \equiv_\varepsilon \mu_2 \text{ iff} \]
  
  \[ \begin{align*}
  \mu_1 &= (1-\varepsilon)\mu_1' + \varepsilon\mu_1'' \\
  \mu_2 &= (1-\varepsilon)\mu_2' + \varepsilon\mu_2'' \\
  \mu_1' &\equiv \mu_2'
  \end{align*} \]

• **Add parameterizations**
  
  - Consider families of PIOA parameterized by \( k \)
  
• **Require \( \varepsilon \) smaller than any polynomial in \( k \)**

• **Absence of simulation becomes failure**
  
  - Existence of a forger for a signature
  
  - Random nonces that are equal
  
  - Existence of a distinguisher
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

\[
(1 - \gamma) \nu_2 \equiv (1 - \gamma - k - c) \gamma \nu_2 \quad k - c \quad \gamma
\]

\[
(1 - \gamma) \nu_1 \equiv (1 - \gamma - k - c) \gamma \nu_1' \quad k - c \quad \gamma
\]
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 \( \sigma_2 \)
  - \( \nu_1 L(R_k, p(k)k^c) \nu_2 \)

- 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
Nonce Generation

\{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 \(A_k\)
  - \(\nu_1 \rightarrow \nu_1^{'}, \nu_2 \rightarrow \nu_2^{'}\)
- There is no \(\nu_2^{'}\) such that
  - \(\nu_2 \rightarrow \nu_2^{'}\)
  - \(\nu_1^{'} \rightarrow \nu_1^{'} \nu_2^{'}\)

- Signature unforgeable in \(\nu_1^{'}\)
  - Probability at least \(k^{-c}\)
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 $\Pr[\exists_{i \neq j} n_i \neq n_j] < k^{-c}$
Applicability

• **Dolev-Yao Model**
  - Soundness w.r.t. indistinguishability
  - How about correspondence of computations?

• **Cryptographic library**
  - More rigorous/local proofs?
  - Alternative to error sets?

• **Game transformations**
  - Proof method?
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)
Approaches to Nondeterminism

- **Reduce the power of the scheduler**
  - Process Algebras
    - Scheduler sees only enabled action type
  - UC framework
    - ITMs have a token passing mechanism
    - No nondeterminism

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

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

- **Careful specifications**
  - Avoid dangerous nondeterminism in the specification
  - Is it always possible?
Concluding Remarks

• Formal methods are useful
  - Semi-formal proofs may be wrong
  - Semi-formal proofs require attention

• There are several approaches
  - Computational, symbolic, compositional
  - Suitable for crypto-primitives or protocols

• We need proof techniques
  - Algebraic, symbolic, automata theoretic

• Nondeterminism arises and gives problems
  - Restrict resolution of nondeterminism
  - Avoid dangerous nondeterminism
What Else?

• A lot to understand on approximated simulations
  - Are they connected to metrics?
  - Can we define them incrementally
    • How far can we go without polynomial bounds?
  - How about approximated language inclusion?

• Need more techniques
  - Can we have a uniform view?
  - Can we relate better computational and symbolic approaches?
  - Any crucial differences between crypto-primitives and protocols?
  - How about cross migration of techniques?

• Need more automation
  - ... but we need to understand what we automate