Computational BPL - A Stochastic Approach

Gergei Bana

with Koji Hasebe and Mitsuhiro Okada

Two Approaches to Modeling Cryptographic Protocols

- **Symbolic**
  - Uses high level formal language
  - Amenable to automatization
  - Treats encryption as black-box operation
  - Accuracy is unclear
- **Computational**
  - More detailed description using probabilities and complexity
  - Proofs by hand
  - More accurate

Formal Model

- **Description via formal logic**

1. $A \rightarrow B : \{\!|A\ N_1|\!\}_{K_B}$
2. $B \rightarrow A : \{\!|N_1\ N_2|\!\}_{K_A}$
3. $A \rightarrow B : \{\!|N_2|\!\}_{K_B}$

- **Finding adversaries:**
  - Encryption is a black-box operation
  - Given a protocol, use automation to find attacks or prove security

1. $A \rightarrow M : \{\!|A, N_1|\!\}_{K_M}$
2. $M(A) \rightarrow B : \{\!|A, N_1|\!\}_{K_B}$
3. $B \rightarrow M(A) : \{\!|N_1, N_2|\!\}_{K_A}$
4. $M \rightarrow A : \{\!|N_1, N_2|\!\}_{K_A}$
5. $A \rightarrow M : \{\!|N_2|\!\}_{K_M}$
6. $M(A) \rightarrow B : \{\!|N_2|\!\}_{K_B}$

Computational View

- Sequence (ensemble) of probability distributions indexed by security parameter η. As η increases, the distributions become more spread.
- Negligible function:
  - $f(\eta)$: for any n natural, there is an η_0 such that whenever $\eta > \eta_0$, $f(\eta) < 1/\eta^n$.
- Adversary:
  - An adversary is successful if it can do bad things with non-negligible probability:
- Security proofs rely on the assumption that are that certain operations (e.g. computing logarithm) are infeasible
  - CCA: the function

$$\begin{aligned}\Pr[\ &(e,d) \longleftarrow \mathcal{K}_\eta\ ; i \longleftarrow \{0,1\}\ ;\\ &m_0, m_1 \longleftarrow \mathsf{A}_\eta^{\mathcal{D}_1(\cdot)}(e);\\ &c \longleftarrow \mathcal{E}_\eta(e, m_i); j \longleftarrow \mathsf{A}_\eta^{\mathcal{D}_2(\cdot)}(e,c) :\\ &i = j \ ] - \tfrac{1}{2}\end{aligned}$$

is neglibible in η

Many approaches

- Linking the two approaches started with Martin Abadi and Philip Rogaway around 2000.
- Today, many approaches are investigated, most importantly:
  - Original Abadi-Rogaway for passive adversaries and its expasions
  - Computational Protocol Compositional Logic of Stanford
  - Reactive Simulatability of M. Backes, B. Pfitzmann, M. Waidner
  - Universial Composability - R. Canetti, J Herzog
  - D. Micciancio - B. Warinschi
  - P. Laud - an A-R style approach to active adversaries
  - some others

Common aspects

- Formal model is simpler, amenable to automation so we want to analyze the formal model, but get computational conclusions
- Natural transition between the models by giving specific computational meaning to formal objects (such as encryption, concatenation etc...)
- Aim: Develop formal models such that
  - Security proven formally should imply computational security - Soundness
  - Formal attack should imply computational attack - Completeness
- Soundness and Completeness are important notions in all these approaches, but their exact meaning differ from one approach to another.

- Watch out:
  - The different approaches are investigated by different groups. It is not quite clear what the limitations of these approaches are and how they relate to each other
  - Some of the proofs seem to be only understood by the authors. Are they all correct?

Using First Order Logic

- **Already existing attempt: Protocol Composition Logic: Datta, Mitchell and cooperators mostly at Stanford**
  - Modal logic similar to Floyd-Hoare logic
  - Proof system: first order logic with axioms and proof rules for protocol actions and temporal reasoning - syntax
  - Formal adversary appears as a particular run of the protocol - semantics
  - Computational semantics: A run or the protocol (controlled by an adversary) is a set of (equiprobable) computational traces.
  - The satisfaction of a modal formula can be checked on each trace and from there the satisfaction in a run, and validity is defined
  - Soundness: If a formula is provable in the syntax, then it is satisfied by any run of the protocol
  - Problems: Dependence of future is allowed, confusion of what strings correspond to, not tracking correlations, questionable proofs
- **Our suggestions:**
  - Instead of focusing on individual traces, focus on probability distributions
  - A run of the protocol (controlled by an adversary) is a probability distribution of computational traces.
  - Syntactic formulas are interpreted as probability distributions
  - A formula is satisfied if a cross section of the computational traces gives the correct distributions.
  - Computational Soundness: If a formula is provable in the syntax, then it is valid; that is, true in any computational run
- **So far only done on a simpler syntax: Basic Protocol Logic by M. Okada and K. Hasebe**

Basic Protocol Logic

- Ordered sorts:
  - names, nonces are both messages
  - for A constant, sort coin^A is also sort coin
- Notation for names:
  - constant: A, B,..
  - variables: Q, Q', Q_1,...
  - either: P, P', P_1,...
- Notation for nonces:
  - constants: N, N', N_1,..
  - variables: n, n', n_1,...
  - either: ν, ...
- Notation for coins, coins^A:
  - constants: r,... r^A,..
  - variables: s,... s^A,..
  - either: ρ, ... ρ^A,..
- Notation for messages:
  - constants: either names or nonces: M, M',...
  - variables: m, m', m_1, ...
- Terms:

$$t ::= M \mid m \mid (t_1,t_2) \mid \{t\}^{\rho}_{P}$$

- Formulas:

$$\varphi ::= P_1 \text{ acts}_1\ t_1; \ldots ;P_k \text{ acts}_k\ t_k \mid t_1 = t_2 \mid t_1 \sqsubseteq t_2 \mid \neg\varphi \mid \varphi_1\wedge\varphi_2 \mid \varphi_1\vee\varphi_2 \mid \varphi_1\rightarrow\varphi_2 \mid \exists m\varphi \mid \forall m\varphi$$

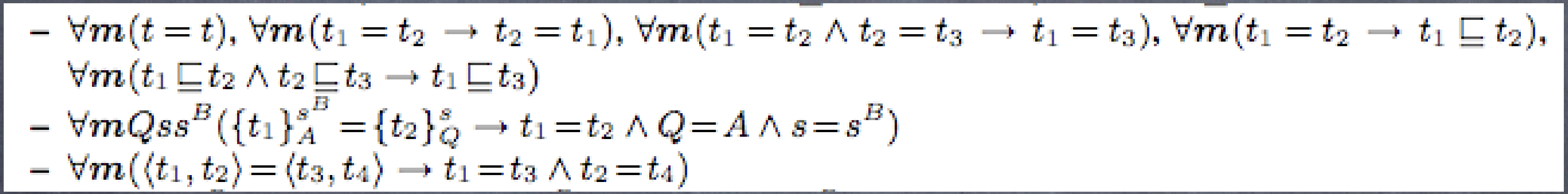
  - where acts is one of sends, receives, generates
  - $=$ represents equality
  - $\sqsubseteq$ represents subterm
  - $\alpha \equiv P_1 \text{ acts}_1\ t_1; \ldots ;P_k \text{ acts}_k\ t_k$ is called a trace formula
- An order preserving merge of two trance formulas
  - The merge of α and β is a γ that contains both α and β in the right order and nothing else.

Roles and Axioms

- A role is a trace formula of the form

  $\alpha \equiv Q \text{ acts}_1\ t_1; \ldots ;Q \text{ acts}_k\ t_k$

- A protocol is a set of roles.
- Axioms
  - Usual first order logic axioms
  - Some further term axioms such as

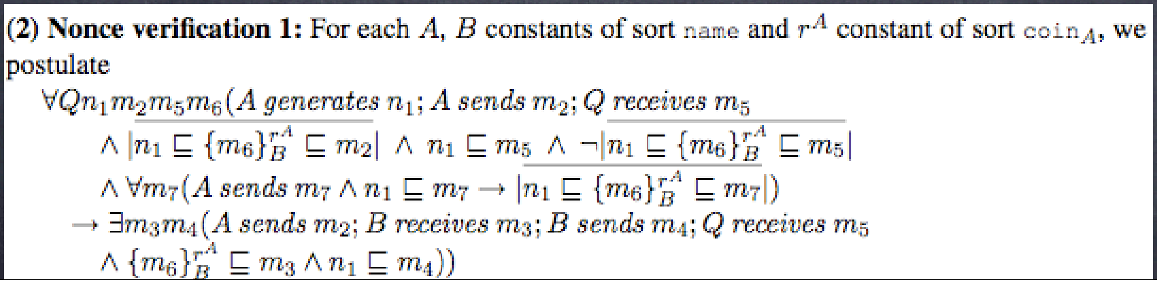


– $\forall m(t=t)$, $\forall m(t_1=t_2 \rightarrow t_2=t_1)$, $\forall m(t_1=t_2 \wedge t_2=t_3 \rightarrow t_1=t_3)$, $\forall m(t_1=t_2 \rightarrow t_1 \sqsubseteq t_2)$, $\forall m(t_1 \sqsubseteq t_2 \wedge t_2 \sqsubseteq t_3 \rightarrow t_1 \sqsubseteq t_3)$

– $\forall m Q s s^B(\{t_1\}_A^{s^B} = \{t_2\}_Q^s \rightarrow t_1=t_2 \wedge Q=A \wedge s=s^B)$

– $\forall m(\langle t_1, t_2\rangle = \langle t_3, t_4\rangle \rightarrow t_1=t_3 \wedge t_2=t_4)$

  - $\beta \rightarrow \alpha$ for $\alpha \subseteq \beta$
  - $\gamma_1 \vee \ldots \vee \gamma_n \leftrightarrow \alpha \wedge \beta$ for γ_i all order preserving merges of α and β.
  - Non-logical axioms about ordering and security: E.g. if a nonce is generated and is sent out encrypted with the key of A, then, if it later appears in some other way, then it had to go through A.
  -



(2) Nonce verification 1: For each A, B constants of sort name and r^A constant of sort $\texttt{coin}_A$, we postulate

$$\begin{aligned}
&\forall Q n_1 m_2 m_5 m_6(\underline{A \textit{ generates } n_1; A \textit{ sends } m_2; Q \textit{ receives } m_5}\\
&\quad \wedge |n_1 \sqsubseteq \{m_6\}_B^{r^A} \sqsubseteq m_2| \ \wedge\ n_1 \sqsubseteq m_5 \ \wedge\ \neg|n_1 \sqsubseteq \{m_6\}_B^{r^A} \sqsubseteq m_5|\\
&\quad \wedge \forall m_7(A \textit{ sends } m_7 \wedge n_1 \sqsubseteq m_7 \rightarrow |n_1 \sqsubseteq \{m_6\}_B^{r^A} \sqsubseteq m_7|)\\
&\rightarrow \exists m_3 m_4(A \textit{ sends } m_2; B \textit{ receives } m_3; B \textit{ sends } m_4; Q \textit{ receives } m_5\\
&\quad \wedge \{m_6\}_B^{r^A} \sqsubseteq m_3 \wedge n_1 \sqsubseteq m_4))
\end{aligned}$$

Honesty and Query Forms

- Q does only the role α^Q:

  $\mathrm{Only}(\alpha^Q) \equiv \forall n(Q \text{ generates } n \rightarrow n \in \mathrm{Generates}(\alpha^Q)) \wedge$

  $\forall m_1(Q \text{ sends } n \rightarrow m_1 \in \mathrm{Sends}(\alpha^Q)) \wedge$

  $\forall m_2(Q \text{ receives } n \rightarrow m_1 \in \mathrm{Receives}(\alpha^Q))$

- Q does only the role α^Q but may not finish:

  $\mathrm{Honest}(\alpha^Q) \equiv \exists Q n \bigvee_{i \in \{0\} \cup \{ j \mid acts_j = sends \} \cup \{k\}} \alpha^Q_{\leq i} \wedge \mathrm{Only}(\alpha^Q_{\leq i})$

- Query form:

  $\mathrm{Honest}(\alpha^A) \wedge \beta^B \wedge \mathrm{Only}(\beta^B) \rightarrow \gamma$

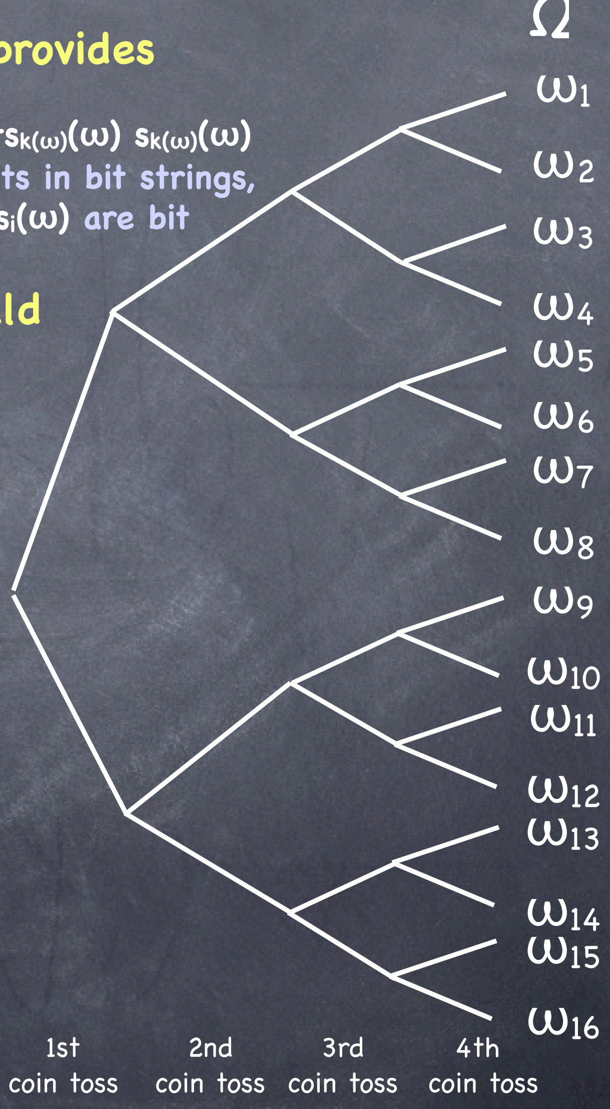
- Protocol correctness holds if the query form can be proven in the syntax

Computational Execution

- Fix a computational public key encryption scheme that is CCA-2 secure
  - Key generation algorithm K_η output randomly generated encryption decryption key pair (e,d)
  - Encryption algorithm: for a key e, plaintext s, and random seed r, outputs E(e, s, r)
  - Decryption for decryption key d and ciphertext c outputs D(d, c) such that D(d, E(e, s, r)) = s if (e,d) is generated by K_η
- Fix a computational pairing
  - [,]
- Honest participants follow their role, other malicious participants may be present

Computational Execution

- Computational execution for each value of η provides
  - a probability space (Ω,p) E.g. see graph.
  - a trace $Tr(\omega) = P_1(\omega)\ acts_1(\omega)\ s_1(\omega); \ldots ;P_{k(\omega)}(\omega)\ acts_{k(\omega)}(\omega)\ s_{k(\omega)}(\omega)$ for each ω, where $P_i(\omega)$ are names of the participants in bit strings, $acts_{k(\omega)}(\omega)$ are either send receive or generate, and $s_i(\omega)$ are bit strings.
- When can we say that A sends t? When should this formal expression be satisfied?
- Should be something like:
  - Computational interpretation of a term gives a probability distribution of bit strings on Ω.
  - First constants have to be interpreted, then variables, then terms.
  - $t_1 = t_2$ is satisfied in the computational semantics if the interpretations are equal
  - $t_1 \sqsubseteq t_2$ is satisfied in the semantics if there is a term t with $t_1 \sqsubseteq t$ such that the interpretation of t and of t_2 are equal.
  - P acts t is satisfied if there is a function J on Ω with natural values such that $Tr_{J(\omega)}(\omega)$ is of the form P acts $s_1(\omega)$ where $s_1(\omega)$ is the interpretation of t.
- But there are problems... Can J be arbitrary? No, it cannot depend on the future...

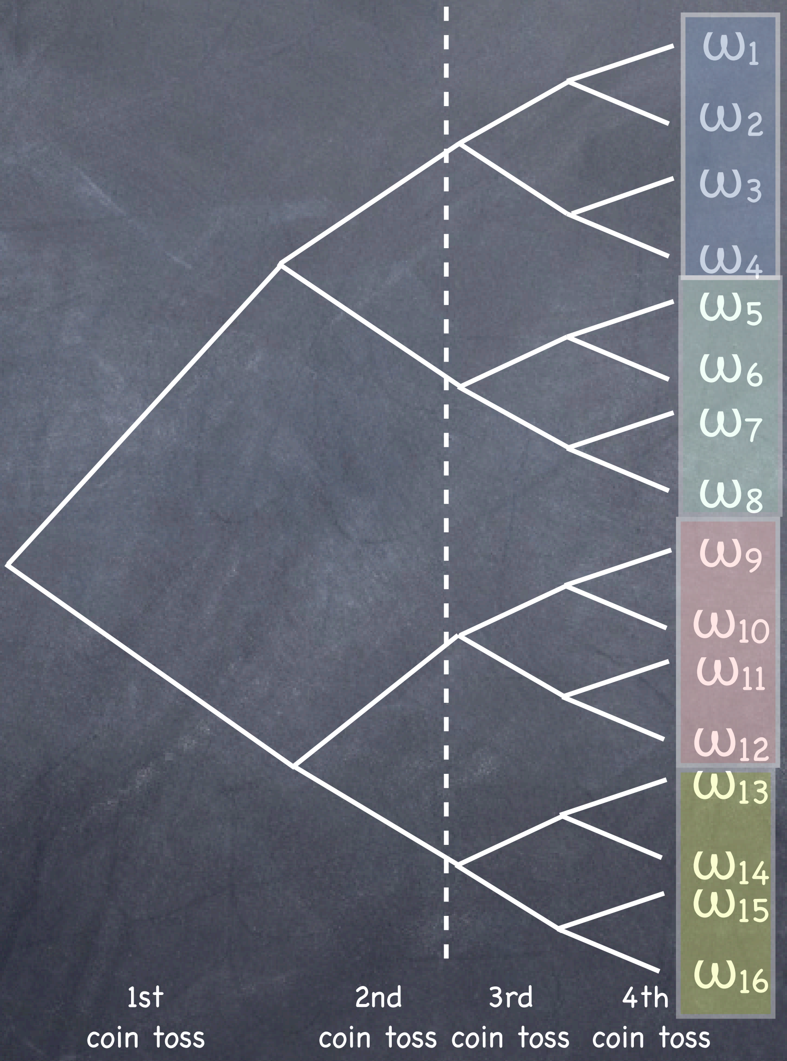


Filtration 1

How are random variables that depend only on randomness until 2nd coin toss characterised?

Random variables that are determined until the 2nd coin toss are constant on these four sets.

Random variables that are independent of what happened until the 2nd coin toss have the same distributions restricted to these sets.

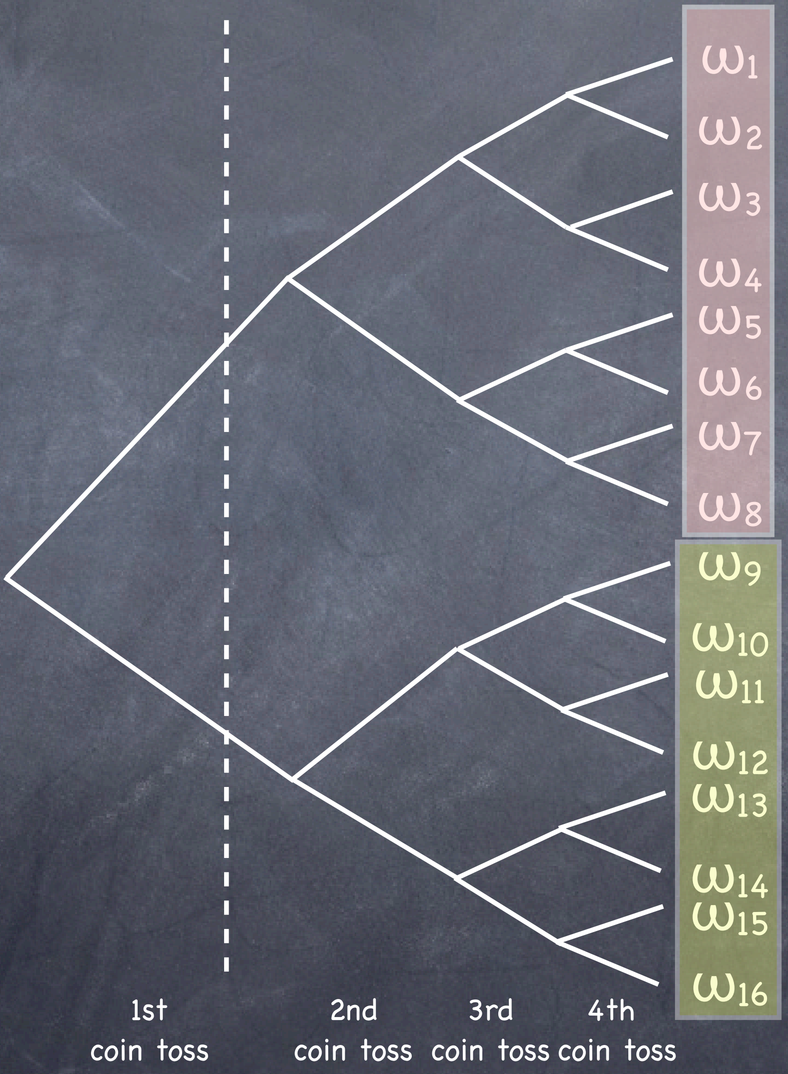


Let F_2 denote the set of these sets.

Filtration 2

Random variables that are determined until the 1st coin toss are constant on these two sets.

Random variables that are independent of what happened until the 1st coin toss have the same distributions restricted to these sets.



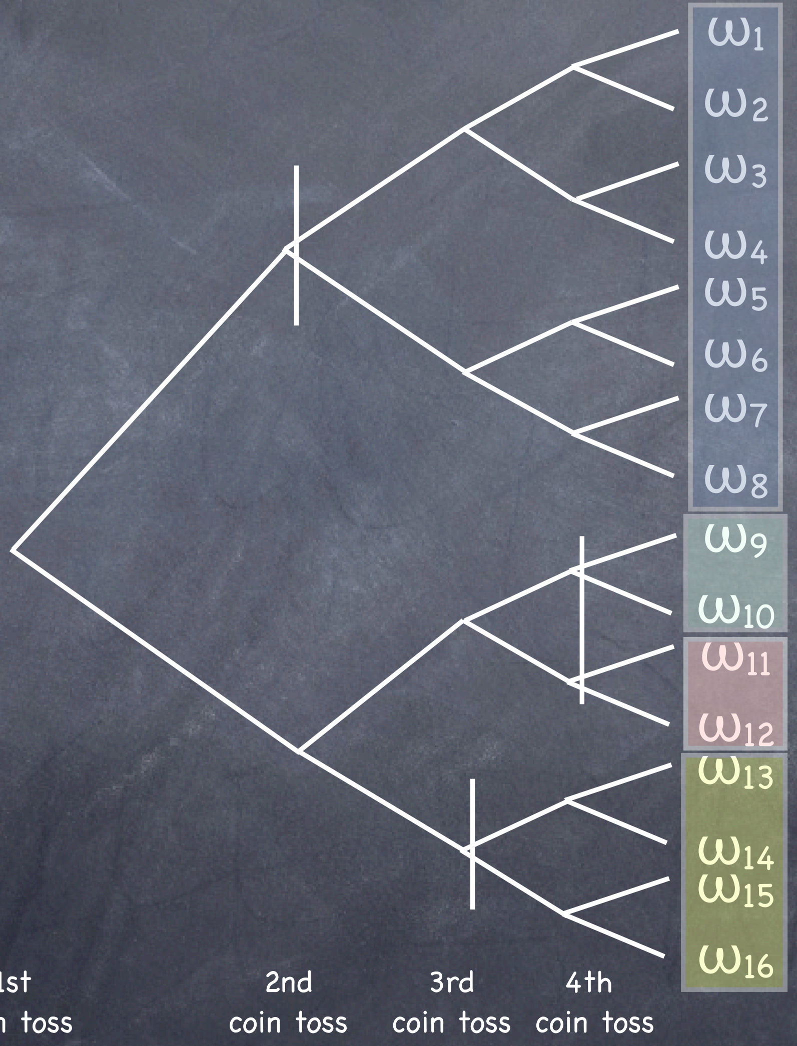
Let F_1 denote these sets.

In general: $F_0, F_1, F_2, \ldots, F_n$ -filtration

Stopping time

Stop so that it does not depend on future

Events until the stopping time J. Random variables that depend only on events until J are constant on these sets.



The set of these sets: F_J

Computational Semantics

- Computational objects corresponding to constants and variables:
  - Principals. D_P A set of bit-string for the principals (indexed by η). To each element A, there belongs a pair of random variables $(e_A(\omega),d_A(\omega))$ on Ω for the generated keys such that they are measurable with respect to F_0.
  - Nonces: D_N Elements are random variables (indexed by η) on Ω which have even distribution on a set of bit strings with fixed length.
  - Messages D_M: random variables taking on Ω bit-string values.
  - Random seeds of encryption: R random variables, R_g with good distribution for the encryption in question.
- Interpretation of constants, variables and terms
  - $\Phi_C(A)$ is in D_P such that the associated $(e_A(\omega),d_A(\omega))$ has the correct distribution and for different constants they are independent, $\Phi_C(N)$ is in D_N, $\Phi_C(r)$ is in R_g
  - $\Phi_C(A)$ is extended to variables so that $\Phi(Q)$ is in D_P, $\Phi(n)$ is in D_N, $\Phi(m)$ is in D_M, $\Phi(s^A)$ is in R_g, $\Phi(s)$ is in R.
  - $\Phi(\ (t_1,t_2)\ )= [\Phi(t_1),\Phi(t_2)]$,
  - $\Phi(\ \{t\}^{\rho}_{P})\ = E(\ e_{\Phi(P)}\ ,\ \Phi(t)\ ,\ \Phi(\rho)\ )$ (some difficulty here as for honest encryptions random seed have to be independent of what happened before)
- From this presentation we are omitting technical difficulties arising from the security parameter and equivalence up to negligible probability. Detail are in the proceedings.

Satisfaction and Soundness

- Satisfaction of formulas
  - $t_1 = t_2$ is satisfied in the computational semantics if $\Phi(t_1) = \Phi(t_2)$, $t_1 \sqsubseteq t_2$ is satisfied in the semantics if there is a term t with $t_1 \sqsubseteq t$ such that $\Phi(t) = \Phi(t_2)$
  - P sends/receives t is satisfied if there is a stopping time J on Ω such that $Tr_{J(\omega)}(\omega)$ is of the form P sends/receives $s_1(\omega)$ where $s_1(\omega)$ is the interpretation of t
  - P generates ν is satisfied if there is a stopping time J on Ω such that $Tr_{J(\omega)}(\omega)$ is of the form P generates $s_1(\omega)$ where $s_1(\omega)$ is the interpretation of ν and $s_1(\omega)$ is independent of F_{J-1}
  - For sequence of actions P_1 $acts_1$ t_1; ... ;P_k $acts_k$ t_k same, but J_1, J_2,... J_k
  - Then satisfaction of $\neg\varphi$ $\varphi_1\wedge\varphi_2$ $\varphi_1\vee\varphi_2$ $\varphi_1\rightarrow\varphi_2$ $\exists m\varphi$ $\forall m\varphi$ are defined in the usual way.
  - A formula is true in a model if for each extension of Φ to variables is satisfied.
- Soundness
  - If the encryption scheme is CCA-2, then the axioms of BPL are computationally sound, and therefore, if a formula can be proven in BPL, then it is true in any computational execution.

Conclusions

- Showed a new, fully probabilistic technique to give sound computational semantics to first-order syntax. Filtrations from the theory of stochastic processes play an essential role
- Soundness
  - Apply the method to Protocol composition logic