Recent approaches to computational semantics for first-order logical analysis of cryptographic protocols

Gergei Bana
Technical University, Lisbon

with Koji Hasebe (Tsukuba Univ) and Mitsuhiro Okada (Keio Univ)

Linking Formal and Computational Views

- Linking the two approaches started with Martin Abadi and Philip Rogaway around 2000 - passive adversaries.
- Active adversaries in two groups:
  - Two-world view
    - Symbolic and computational executions are formalized separately as well as security properties
    - Soundness: Try to prove that no successful symbolic (Dolev-Yao) attacker implies no successful computational attacker.
    - Such are
      - Reactive Simulatability of M. Backes, B. Pfitzmann, M. Waidner
      - D. Micciancio, B. Warinschi, Cortier (mapping lemma)
      - V. Cortier, H. Comon-Lundh (soundness of observational equivalence)
  - Logical view
    - Only computational execution, symbolic formulas have direct computational meaning
    - Logical theory axiomatizes the relevant properties cryptographic primitives.
    - Security properties are directly proven from the axioms and derivation rules
      - Computational Protocol Compositional Logic of Stanford (John Mitchell's group)
      - Computational Basic Protocol Logic (Keio)

Computational Soundness and Dolev-Yao Adversaries

- Two-world Soundness Theorems:
  - Don't assume much about the specifics of the formal system
  - Prove that no formal Dolev-Yao adversary implies no computational adversary
  - Controls the network
  - It is explicitly formulated what symbolic operations it may do:
    - Encrypt, decrypt with a key it has, pair, etc, things expressible syntactically
    - From a, b, it can compute (a,b)
    - From a, K, it can compute $\{a\}_K$
    - If it has the decryption key, it can compute a from $\{a\}_K$
    - DY adversary does not give complete description of adversarial capabilities.
    - For the soundness proofs complete axiomatization is needed.
- Problems arising from incomplete description
  - Maybe for N, N' nonces, K key, R randomness, an adversary can generate a key K' and R' randomness such that $\{N\}_K^R = \{N'\}_{K'}^{R'}$ - such an equality is usually not listed among the Dolev-Yao rules, and there might be countless others.
  - Counterexamples can be created, i.e. no DY adversary but there is computational adversary
  - To avoid it: Strong assumptions for avoiding arbitrary parsing, such as appending half of the encrypting key to the end of the encryption

Incompleteness

- First order logic
  - Security properties are proved directly from a set of axioms
  - Only state as axioms what we can actually prove to be sound
  - Eg: K, K', R, R' are honestly generated, then $\{N\}_K^R = \{N'\}_{K'}^{R'}$ implies N = N'
  - Soundness of the axioms ensure that if the security property is proven formally, then there is no successful adversary.
- Logical view axioms
  - Usual first order logic axioms
  - Some further term axioms such as

$$\forall \boldsymbol{m}(t_1 = t_2 \rightarrow t_2 = t_1), \forall \boldsymbol{m}(t_1 = t_2 \wedge t_2 = t_3 \rightarrow t_1 = t_3), \forall \boldsymbol{m}(t_1 \sqsubseteq t_2 \wedge t_2 \sqsubseteq t_3 \rightarrow t_1 \sqsubseteq t_3),$$

$$\forall \boldsymbol{m}P(t_1 \sqsubseteq_{(\neg)P} t_2 \rightarrow t_1 \sqsubseteq t_2), \forall \boldsymbol{m}P(t_1 = t_2 \rightarrow t_1 \sqsubseteq_{(\neg)P} t_2), \forall \boldsymbol{m}P(t_1 \sqsubseteq_{(\neg)P} t_2 \wedge t_2 \sqsubseteq_{(\neg)P} t_3 \rightarrow t_1 \sqsubseteq_{(\neg)P} t_3)$$

$$\forall \boldsymbol{m}Qss'(\{t_1\}_Q^s = \{t_2\}_Q^{s'} \rightarrow t_1 = t_2),$$

If t_1 and t_2 contain constants only, then $\{t_1\}_A^r = \{t_2\}_B^{r'} \rightarrow A = B$

$$\forall \boldsymbol{m}(t \sqsubseteq \langle t_1, t_2\rangle \rightarrow t \sqsubseteq t_1 \vee t \sqsubseteq t_2 \vee t = \langle t_1, t_2\rangle), \forall \boldsymbol{m}Qs(t_1 \sqsubseteq \{t_2\}_Q^s \rightarrow t_1 = \{t_2\}_Q^s \vee \exists mQ's'(\{t_2\}_Q^s = \{m\}_{Q'}^{s'} \wedge t_1 \sqsubseteq m)),$$

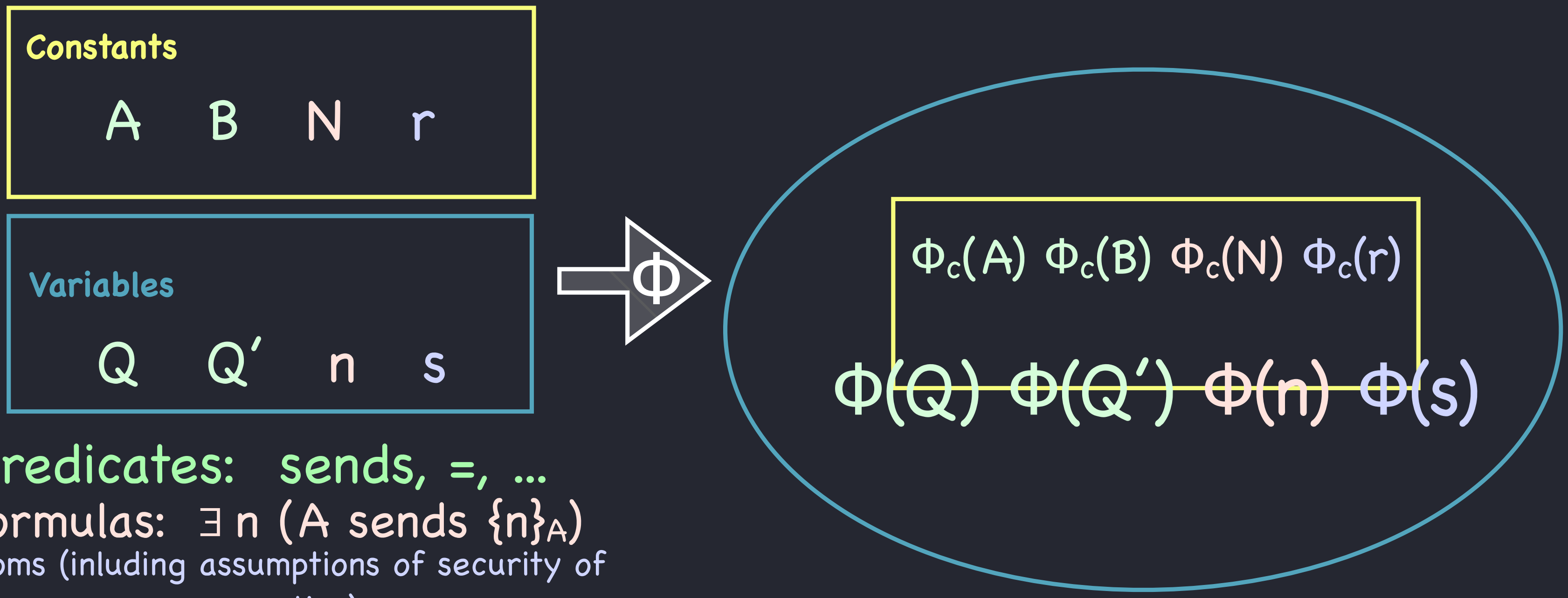
$$\forall \boldsymbol{m}(t \sqsubseteq_P \langle t_1, t_2\rangle \rightarrow t \sqsubseteq_P t_1 \vee t \sqsubseteq_P t_2 \vee t = \langle t_1, t_2\rangle), \forall \boldsymbol{m}Qs(t_1 \sqsubseteq_P \{t_2\}_Q^s \rightarrow t_1 = \{t_2\}_Q^s \vee \exists ms'(\{t_2\}_Q^s = \{m\}_P^{s'} \wedge t_1 \sqsubseteq_P m)), \forall \boldsymbol{m}s(t_1 \sqsubseteq_P \{t_2\}_P^s \rightarrow t_1 = \{t_2\}_P^s \vee t_1 \sqsubseteq_P t_2))$$

  - Malicious participants can use bad keys: no security for those
  - Axioms about 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, accessible to A via decryption.
  - The set of axioms is not complete, there may be more that are sound
  - But as long as only sound axioms are used, it is ok.

Using First Order Logic

Syntax

Semantics: a computational run controlled by an adversary



Predicates: sends, =, ...

Formulas: $\exists\, n\ (A \text{ sends } \{n\}_A)$

Axioms (inluding assumptions of security of encryption)

Derivation rules

Protocol: Protocol Roles, Honesty assumptions

$\Downarrow$

Security Property (some formula e.g. agreement)

Φ: computational interpretation of symbols

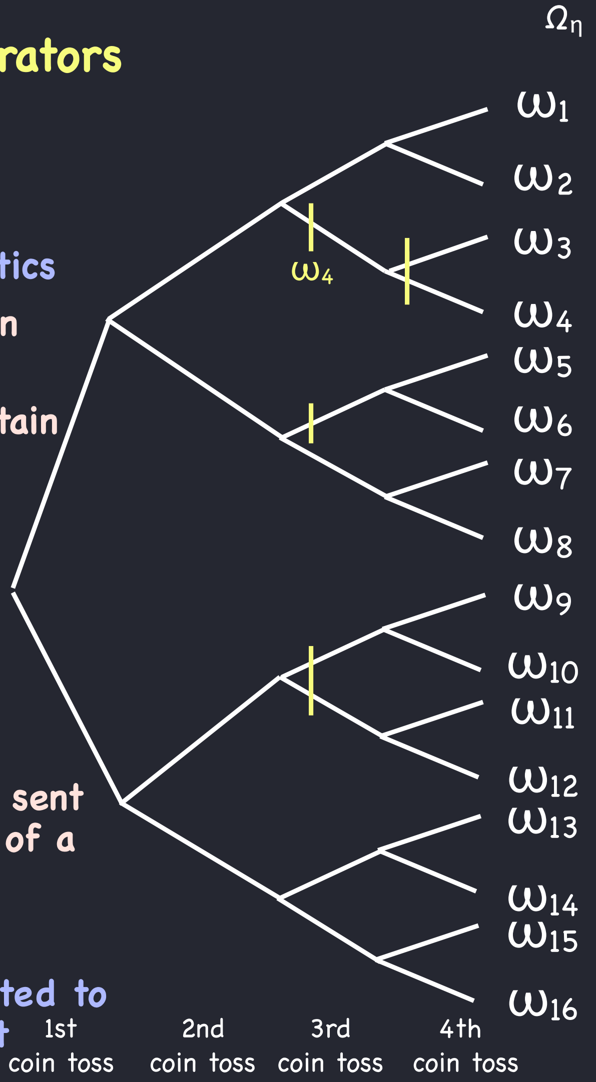
Satisfaction/validity of formulas have to be defined

If symbolic axioms and derivation rules are valid computationally, then proving a security property symbolically implies computational validity

Computational PCL

Protocol Composition Logic: Datta, Mitchell and cooperators

- Modal logic similar to Floyd-Hoare logic
- Proof system: first order logic with axioms and proof rules for protocol actions and temporal reasoning - syntax
- Adversary appears as a particular run of the protocol - semantics
- Computational semantics: A run of the protocol (controlled by an adversary) is a set of (equiprobable) computational traces.
- The satisfaction of a modal formula (as long as it does not contain Indist can be checked on each trace and from there the satisfaction in a run, if satisfaction on traces holds with overwhelming prob and validity is defined
- Soundness: If a formula is provable in the syntax, then it is satisfied (computationally) by any run of the protocol
- E.g. $\exists n$ (A sends $\{n\}_K^R$) is satisfied in an execution if on an overwhelming number of traces A has a send action where the sent item equals the encryption with the given key and randomness of a possible interpretation of a nonce.
- But does it matter where the send action happened?
- For soundness theorems, executable algorithms have to be created to get a counterexample. If things depend on future, they are not executable.



Further issues

- Coincidences
  - Suppose that the encryption is such that randomly generated nonce bit-strings n1 and n2, any public-key bit-string e2, and random seed r2, there is a public key e1 and random seed r1 such that E(k1, n1, r1) = E(k2, n2, r2). Suppose that principal A generates a nonce n1, and then B receives E(k2, n2, r2) from the adversary. However, in this case, in their semantics, $\exists N \exists R \exists K.\mathrm{New}(A,N) \wedge \mathrm{Receive}(B,\{N\}_K^R)$ is satisfied, which is strange.
  - This contradicts an axiom from (not computational) PCL, $\mathrm{FirstSend}(X,t,t') \wedge a(Y,t'') \rightarrow \mathrm{Send}(X,t') < a(Y,t'')$, meaning in the above case that the first send action of A sending N had to occur before B could do anything with N
  - Good axiom but not sound in this computational interpretation
- Lacking term axioms
  - Term axioms are defined through the semantics (that is, those are true which are sound by definition)

Our Suggestion

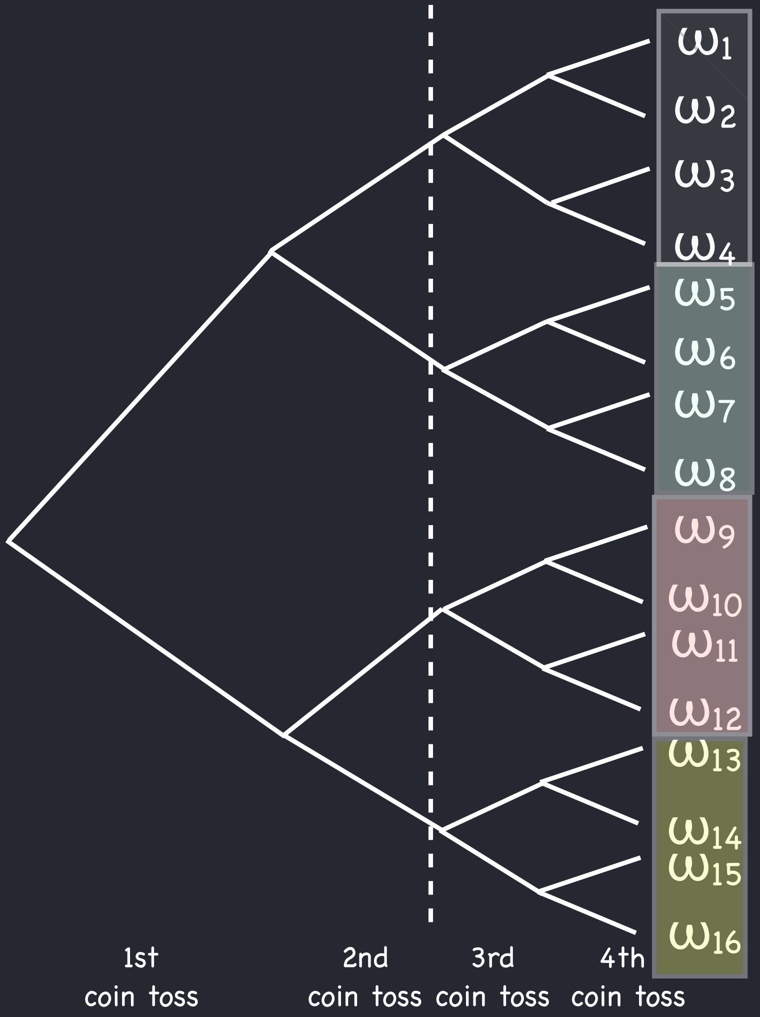
- Execution of a cryptographic protocol is a stochastic process even if we ignore that. So let's not ignore that, and use the tools developed to handle such them.
- Instead of focusing on individual traces, focus on probability distributions on non-negligible sets of traces
- A run of the protocol (controlled by an adversary) is a probability distribution of computational traces (with an underlying sample space).
- Terms are interpreted as random variables
- A formula is satisfied on non-negligible sets of traces if a cross section of the computational traces gives the correct random variables.
- 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, but can be employed to PCL

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.

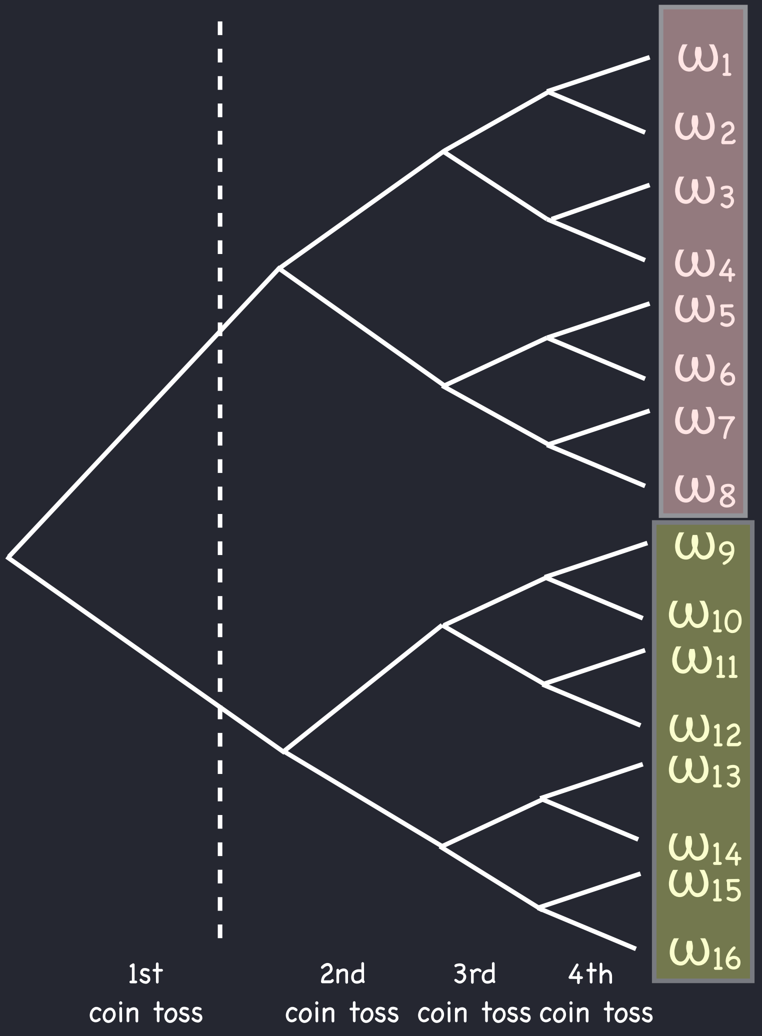


Let F_2 denote the set of these sets.

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.



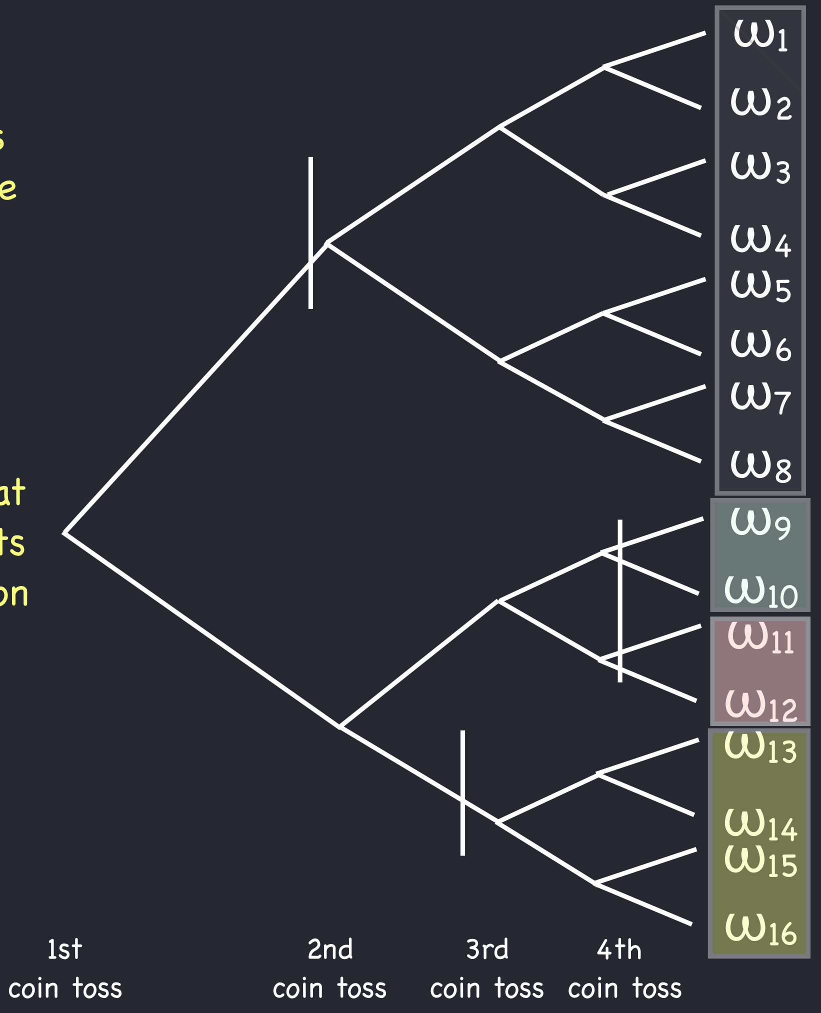
Let F_1 denote these sets.

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

Basic Protocol Logic

- Ordered sorts:
  - names, nonces are both messages
  - for A constant, sort coin^A is also sort coin
- Notation for principal 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 messages:
  - constants: either names or nonces: M, M′,...
  - variables: m, m′, m_1, ...
- Notation for coins, coins:
  - constants: r,...
  - variables: s,...
  - either: ρ, ...
- Terms:

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

- Formulas:

$$\varphi ::= P_1 \text{ acts}_1\ t_1; \ldots ;P_k \text{ acts}_k\ t_k \mid t_1 = t_2 \mid t_1 \sqsubseteq t_2 \mid t_1 \sqsubseteq_P t_2 \mid t_1 \sqsubseteq_{\neg P} t_2 \mid |\overline{t_1 \sqsubseteq t_2 \sqsubseteq t_3}| \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \exists m\varphi \mid \forall m\varphi$$

  - where acts is one of sends, receives, generates
  - = represents equality, $\sqsubseteq$ represents subterm, $\sqsubseteq_P$ $\sqsubseteq_{\neg P}$...
  - $|\overline{t_1 \sqsubseteq t_2 \sqsubseteq t_3}|$: $t_1 \sqsubseteq t_2 \sqsubseteq t_3$ and the only way t_1 occurs in t_3 is within t_2....
  - $\alpha \equiv P_1 \text{ acts}_1\ t_1; \ldots ;P_k \text{ acts}_k\ t_k$ 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.

Subterm Relations

- A number of different Subterm Relations:
  - $t_1 \sqsubseteq t_2$ usual subterm relation
  - $t_1 \sqsubseteq_P t_2$ t_1 is a subterm of t_2 such that it can be received from t_2 via decryption by only the private key of P
  - $t_1 \sqsubseteq_{\neg P} t_2$ t_1 is a subterm of t_2 such that it can be received from t_2 via decryption by other than the private key of P
  - $|\overline{t_1 \sqsubseteq t_2 \sqsubseteq t_3}|$: $t_1 \sqsubseteq t_2 \sqsubseteq t_3$ and the only way t_1 occurs in t_3 is within t_2.
- Reason:
  - No decrypt etc action as opposed to PCL
  - During verification, it is proven that something, e.g. a nonce is contained in something else via one of these subterm relations, and that implies that the given nonce is identical with another.

Roles, Axioms, Security

- 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

$\forall \boldsymbol{m}(t_1 = t_2 \rightarrow t_2 = t_1), \forall \boldsymbol{m}(t_1 = t_2 \wedge t_2 = t_3 \rightarrow t_1 = t_3), \forall \boldsymbol{m}(t_1 \sqsubseteq t_2 \wedge t_2 \sqsubseteq t_3 \rightarrow t_1 \sqsubseteq t_3),$
$\forall \boldsymbol{m}P(t_1 \sqsubseteq_{(\neg)P} t_2 \rightarrow t_1 \sqsubseteq t_2), \forall \boldsymbol{m}P(t_1 = t_2 \rightarrow t_1 \sqsubseteq_{(\neg)P} t_2), \forall \boldsymbol{m}P(t_1 \sqsubseteq_{(\neg)P} t_2 \wedge t_2 \sqsubseteq_{(\neg)P} t_3 \rightarrow$
$t_1 \sqsubseteq_{(\neg)P} t_3)$
$\forall \boldsymbol{m}Qss'(\{t_1\}^s_Q = \{t_2\}^{s'}_Q \rightarrow t_1 = t_2),$
If t_1 and t_2 contain constants only, then $\{t_1\}^r_A = \{t_2\}^{r'}_B \rightarrow A = B$

$\forall \boldsymbol{m}(t \sqsubseteq \langle t_1, t_2 \rangle \rightarrow t \sqsubseteq t_1 \vee t \sqsubseteq t_2 \vee t = \langle t_1, t_2 \rangle), \forall \boldsymbol{m}Qs(t_1 \sqsubseteq \{t_2\}^s_Q \rightarrow t_1 = \{t_2\}^s_Q \vee$
$\exists mQ's'(\{t_2\}^s_Q = \{m\}^{s'}_{Q'} \wedge t_1 \sqsubseteq m)),$
$\forall \boldsymbol{m}(t \sqsubseteq_P \langle t_1, t_2 \rangle \rightarrow t \sqsubseteq_P t_1 \vee t \sqsubseteq_P t_2 \vee t = \langle t_1, t_2 \rangle), \forall \boldsymbol{m}Qs(t_1 \sqsubseteq_P \{t_2\}^s_Q \rightarrow t_1 = \{t_2\}^s_Q \vee$
$\exists ms'(\{t_2\}^s_Q = \{m\}^{s'}_P \wedge t_1 \sqsubseteq_P m)), \forall \boldsymbol{m}s(t_1 \sqsubseteq_P \{t_2\}^s_P \rightarrow t_1 = \{t_2\}^s_P \vee t_1 \sqsubseteq_P t_2))$

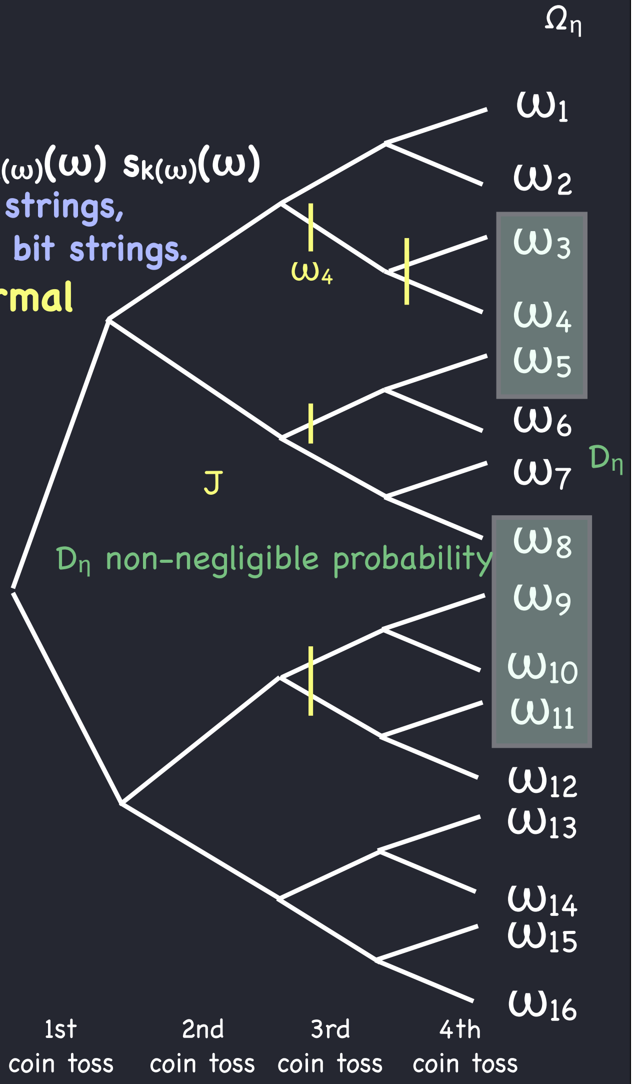
  - Malicious participants can use bad keys: no security for those
  - Axioms about ordering and security: E.g. if a nonce is generated and is sent out encrypted with the key of A, then, if it later appears in some other way, then it had to go through A, accessible to A via decryption.
  - The set of axioms is not complete, there may be more that are sound
- Security Properties
  - E.g Agreement. If A, B are honest, and B finishes the protocol, then both A finished and they agree on the variables.

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
  - [,]
- Tagging
- Honest participants (interacting PPT Turing Machines) follow their role
- Fix a PPT interacting Turing Machine for the adversary controlling the network

Computational Execution

- Computational execution for each value of η provides
  - a probability space (Ω,p) E.g. see graph.
  - a trace $Tr(\omega) = P_1(\omega)\ acts_1(\omega)\ s_1(\omega); \dots ;P_{k(\omega)}(\omega)\ acts_{k(\omega)}(\omega)\ s_{k(\omega)}(\omega)$ 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 random variable of bit strings on D_η.
  - First constants have to be interpreted, then variables, then terms.
  - $t_1 = t_2$ is satisfied on D_η in the computational semantics if the interpretations are equal on D_η (up to negligibility)
  - $t_1 \sqsubseteq t_2$ 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 on D_η.
  - P acts t is satisfied on D_η if there is a function J on D_η 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.
- Can J be arbitrary? No, it cannot depend on the future... Has to be stopping time.



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 D_η 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 D_η which look like nonces.
  - Messages D_M: random variables taking on D_η 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 and independent of everything that happened before
  - Φ_C is extended to variables so that $\Phi(Q)$ is in D_P, $\Phi(n)$ is in D_N, $\Phi(m)$ is in D_M, $\Phi(s)$ is in R.
  - $\Phi((t_1,t_2))= [\Phi(t_1),\Phi(t_2)]$,
  - $\Phi(\{t\}_P^\rho) = E(e_{\Phi(P)} , \Phi(t) , \Phi(\rho))$ (some difficulty here as for honest encryptions encrypted item has to be independent of what happened before - use filtration)

Satisfaction and Soundness

- Satisfaction of formulas
  - $t_1 = t_2$ is satisfied in the computational semantics if $\Phi(t_1) = \Phi(t_2)$, on D_η, $t_1 \sqsubseteq t_2$ is satisfied in the semantics if there is a term t with $t_1 \sqsubseteq t$ such that $\Phi(t) = \Phi(t_2)$ on D_η
  - P sends/receives t is satisfied if there is a stopping time J on D_η 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 v is satisfied if there is a stopping time J on D_η such that $Tr_{J(\omega)}(\omega)$ is of the form P generates $s_1(\omega)$ where $s_1(\omega)$ is the interpretation of v 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 $J_1, J_2, \ldots J_k$
  - 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 valid (true in any computational execution.)

Comparison with Computational PCL

- Equiprobable traces
  - They count equiprobable traces to get probabilities: works only for finitely many
- No filtrations
  - They do not use filtrations or anything else instead, therefore there is no guarantee that if a formula such as A sends t is satisfied on all traces, the points picked on the traces will depend only on the past. Dependence on the past however is necessary for the soundness proofs.
- Independence
  - Suppose that the encryption is such that randomly generated nonce bit-strings n1 and n2, any public-key bit-string e2, and random seed r2, there is a public key e1 and random seed r1 such that E(k1, n1, r1) = E(k2, n2, r2). Suppose that principal A generates a nonce n1, and then B receives E(k2, n2, r2) from the adversary. However, in this case, in their semantics, $\exists N \exists R \exists K.\mathrm{New}(A,N) \wedge \mathrm{Receive}(B,\{N\}^R_K)$ is satisfied, which is pathologic (and contradicts one of their axioms)
  - This cannot occur in our setup because K has to be generated independently of N

Conclusions

- Using First Order Logic avoids some problems with DY adversary
  - The formal DY adversary needs to reflect symbolically all possible computational malicious possibilities. Incomplete description prevents soundness.
  - In first order logic, incomplete set of axioms may prevent us to be able to prove security property, but still have soundness.
- 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
- Importance:
  - correctly justify formal proofs in the computational world
  - the way we think about semantics should drive our intuition in formal reasoning.
- Future work
  - Do proofs rigorously using Segala's work
  - Apply the method to Protocol Composition Logic
  - Such problems are not unique to first order logic...