Formal Methods (Models)

Jeannette M. Wing

15-827 Security and Cryptography
20 October 1997

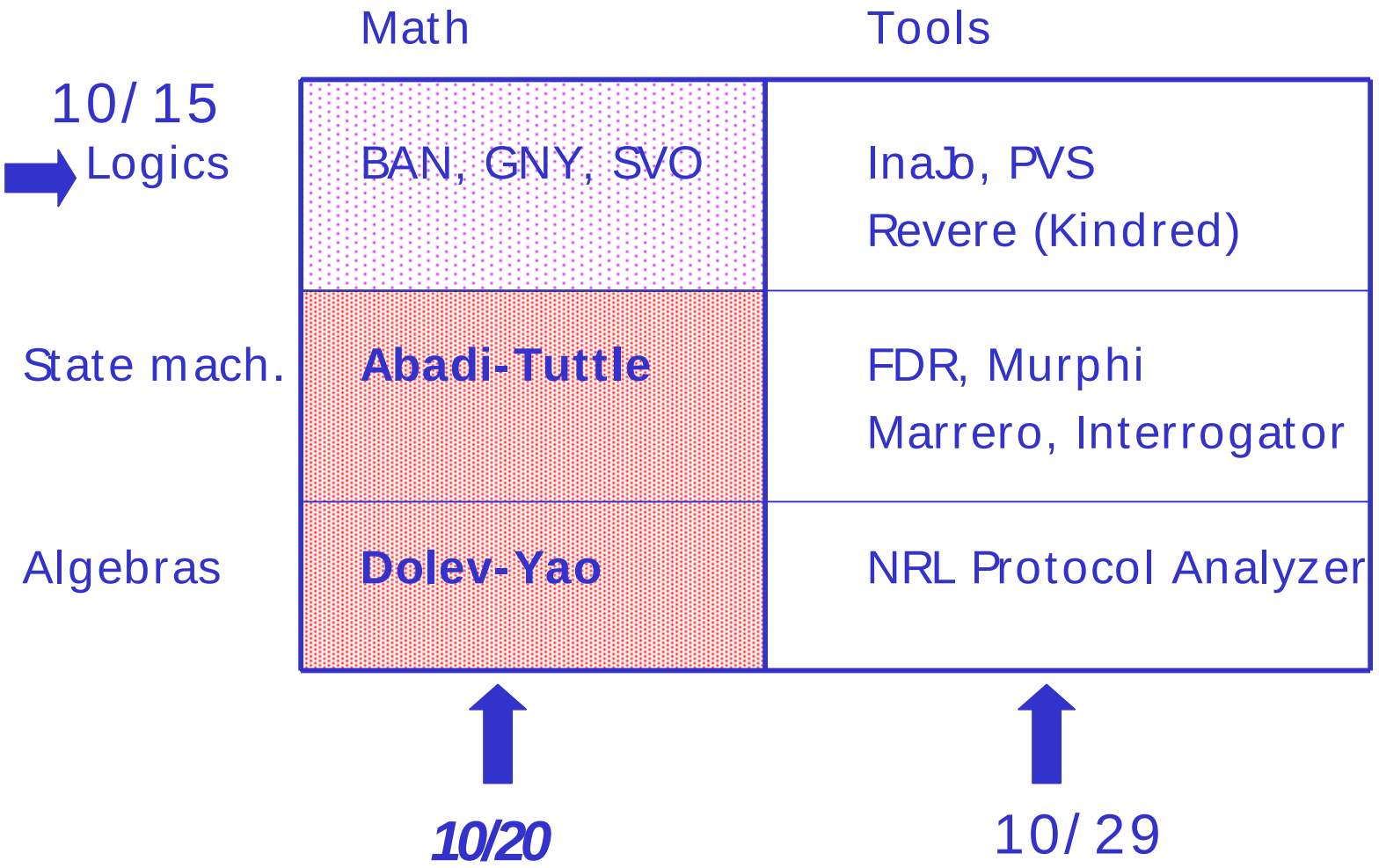
Plan for Today

- Dolev-Yao
- Abadi-Tuttle
- Others
  - Meadows
  - Heintze-Tygar
  - Woo-Lam

Goals for Today

- Towards a formal model for crypto protocols
  - What you can and cannot do with or say in these models
- Semantics of BAN

Taxonomy of Approaches



Dolev-Yao Overview

- Approach: *algebraic*
  - operations on strings
    - models operations of principals and intruder
- Protocols
  - *Ping-pong*: two parties
    - Cascade
    - Name-stamp
  - Public-key crypto
- Property
  - "Secure" means *secrecy*
- Efficient algorithms for checking

Public Key Encryption Review

E_X Encryption function for user X

D_X Decryption function for user X

Secure public directory contains (X, E_X) pairs.

D_X known only to X.

1. $E_X D_X = D_X E_X = 1$
2. Knowing $E_X(M)$ and the public directory does not reveal anything about the value of M.

Example 1

a. A -> B: $(A, E_B(M), B)$

b. B -> A: $(B, E_A(M), A)$

Breaking Example 1

a. A -> B: (A, E_B(M), B)

b. B -> A: (B, E_A(M), A)

1. A -> Z(B): (A, E_B(M), B) %Z intercepts message from A
2. Z -> B: (Z, E_B(M), B) %Z sticks Z in for A
3. B -> Z: (B, E_Z(M), Z) %B follows its protocol
4. Z decodes E_Z(M) to get M.

Example 2 (variation of Needham-Schroeder)

a. A -> B: $(A, E_B(MA), B)$

b. B -> A: $(B, E_A(MB), A)$

Example 3: Extra encryption is not always a good idea.

a. A -> B: $(A, E_B(E_B(M)A), B)$

b. B -> A: $(B, E_A(E_A(M)B), A)$

Breaking Example 3

a. A -> B: (A, $E_B(E_B(M)A)$, B)

b. B -> A: (B, $E_A(E_A(M)B)$, A)

1. B -> Z(A): (B, $E_A(E_A(M)B)$, A) %Let M' = $E_A(M)B$.
   %Z extracts $E_A(M')$.
2. Z -> A: (Z, $E_A(E_A(M')Z)$, A) %Z starts protocol with A.
3. A -> Z: (A, $E_Z(E_Z(M')Z)$, Z) %A responds accordingly.
   %Z now has $E_Z(M')$.
4. Z -> A: (Z, $E_A(E_A(M)Z)$, A) %Z starts protocol with A again.
5. A -> Z: (A, $E_Z(E_Z(M)Z)$, Z) %A responds accordingly.
   %Z discovers M.

Summary of Dolev-Yao Results

- Cascade protocols (like Example 1) are *secure* iff:
  - the messages transmitted between X and Y always contain some layers of encryption functions E_X or E_Y;
  - in generating a reply message, each participant A (= X, Y) never applies D_A without also applying E_A.

- Name-stamp protocols (like Examples 2 and 3):
  - There is a polynomial-time ($O(n^8)$) algorithm for checking whether it is *secure*.

Cascade Protocols

- Two parties, X and Y:
  - X does strings of E_X, E_Y, and D_X.
  - Y does strings of E_X, E_Y, and D_Y.
- A protocol is a series of strings of what X does and of what Y does:
  1. X -> Y: $<\text{what X does}>_1 M$
  2. Y -> X: $<\text{what Y does}>_1 <\text{what X does}>_1 M$
  3. X -> Y: $<\text{what X does}>_2 <\text{what Y does}>_1 <\text{what X does}>_1 M$

  ...
- Let γ be a string of operators over Es and Ds; γ is the reduced form of γ after applying algebraic laws: $E_X D_X = D_X E_X = 1$

Example

<what X does> = $E_Y D_X$

<what Y does> = $E_X D_Y E_X E_X D_Y$

Protocol:

X -> Y: $E_Y D_X M$

Y -> X: $E_X D_Y E_X E_X D_Y E_Y D_X M$ % or simply $E_X D_Y E_X M$

Notation ($N_i(X, Y)$):

$\overline{N_1(X, Y)} = E_Y D_X$

$\overline{N_2(X, Y)} = E_X D_Y E_X$

Definition of Secure (Cascade Protocol)

$\Sigma(Z) = E \cup \{D_Z\}$

Σ_2 = all strings of what X does

Σ_3 = all strings of what Y does

A protocol is insecure if there exists some string γ in

$(\Sigma(Z) \cup \Sigma_2 \cup \Sigma_3)^*$ such that

$$\overline{\gamma\ N_i(X, Y)} = \lambda$$

for some $N_i(X, Y)$.

A protocol is secure otherwise.

Example 1 Again

a. A -> B: (A, E_B(M), B)

b. B -> A: (B, E_A(M), A)

Normal: $E_X D_Y E_Y$ M

1. A -> Z(B): (A, E_B(M), B) %Z intercepts message from A
2. Z -> B: (Z, E_B(M), B) %Z sticks Z in for A
3. B -> Z: (B, E_Z(M), Z) %B follows its protocol
4. Z decodes E_Z(M) to get M.

Intruder: $D_Z E_Z D_Y E_Y$ M **% = M**

% I.e., $D_Z E_Z D_Y E_Y = \lambda$

Name-Stamp Protocols

- γ = head(γ)tail(γ), where tail(γ) is a suffix of m bits.

- A user Y can do any of these operations to a string γ:
  a. encrypt E_X
  b. decrypt D_Y
  c. append i_X: $i_X \gamma = \gamma X$
  d. name-match d_X: $d_X \gamma$ = head(γ) if tail(γ) = X; otherwise.
  e. delete d: $d \gamma$ = head(γ)

- Algebraic laws
  - $E_X D_X = D_X E_X = 1$
  - $d_X i_X = d i_X = 1$ %but $i_X d_X = 1$

Restriction

- When a d_X is applied to γ, transmission will not proceed unless tail(γ) = X.
- Thus, we require for normal users, X and Y, that for any text γ transmitted, no d_X or d_Y remains after applying algebraic laws.

Definition of Secure (Name-Stamp Protocol)

$\Sigma(Z) = \{E_A, i_A, d_A, d \mid \text{all } A\} \cup \{D_Z\}$

Σ_2 = all strings of what X does (including new ops)

Σ_3 = all strings of what Y does (including new ops)

A protocol is insecure if there exists some string γ in

$(\Sigma(Z) \cup \Sigma_2 \cup \Sigma_3)^*$ such that

$$\overline{\gamma\ N_i(X, Y)} = \lambda$$

for some $N_i(X, Y)$.

A protocol is secure otherwise.

Example 2 again

a. A -> B: $(A, E_B(MA), B)$

b. B -> A: $(B, E_A(MB), A)$

$\overline{N_1(X, Y)} = E_Y i_X$

$\overline{N_2(X, Y)} = E_X I_Y$

This protocol is secure. (See paper for proof.)

Example 3 again (normal)

a. A -> B: $(A, E_B(E_B(M)A), B)$

b. B -> A: $(B, E_A(E_A(M)B), A)$

$\alpha\,(X, Y) = E_Y i_X E_Y$

$\beta\,(X, Y) = E_X i_Y E_X D_Y d_X D_Y$

$\overline{N1(X, Y)} = E_Y i_X E_Y$

$\overline{N2(X, Y)} = E_X i_Y E_X$

Breaking Example 3

a. A -> B: $(A, E_B(E_B(M)A), B)$
b. B -> A: $(B, E_A(E_A(M)B), A)$

$\alpha(X, Y) = E_Y i_X E_Y$
$\beta(X, Y) = E_X i_Y E_X D_Y d_X D_Y$
$\beta(Z, X) = E_Z i_X E_Z D_X d_Z D_X$

1. B -> Z(A):	$(B, E_A(E_A(M)B), A)$	%Let M' = $E_A(M)B$.
		%Z extracts $E_A(M')$.
2. Z -> A:	$(Z, E_A(E_A(M')Z), A)$	%Z starts protocol with A.
3. A -> Z:	$(A, E_Z(E_Z(M')Z), Z)$	%A responds accordingly.
		%Z now has $E_Z(M')$.
4. Z -> A:	$(Z, E_A(E_A(M)Z), A)$	%Z starts protocol with A again.
5. A -> Z:	$(A, E_Z(E_Z(M)Z), Z)$	%A responds accordingly.
		%Z discovers M.

Breaking Example 3

$$\gamma\,\overline{N_2} = D_Z d D_Z E_Z i_X E_Z D_X d_Z D_X E_X i_Z d D_Z d D_Z E_Z i_X E_Z D_X d_Z D_X E_X i_Z E_X i_Y E_X$$

Breaking Example 3 (parsed)

$$\gamma\ \overline{N_2} = D_z d D_z E_z i_x E_z D_x d_z D_x E_x i_z d D_z d D_z E_z i_x E_z D_x d_z D_x E_x i_z E_x i_Y E_x$$

Abadi-Tuttle Overview

- "Standard" state machine model of distributed systems
  - Included constructs driven by what is needed to model in BAN logic
- Possible worlds semantics a la Hintikka [1962] and logics of knowledge and belief.
- Does not handle multiple runs of a protocol.

Model of Computation

- Principals P_1, ..., P_n, each with
  - local state
  - a set of actions (state-transition relation that can change only P_i's local state and P_e's state.
  - a local protocol, A_i, a function from P_i's local state to the next action Pi is to perform.
- Special principal P_e, the environment
- Global state = $(s_e, s_1, ..., s_n)$
- Protocol = $(A_e, A_1, ..., A_n)$

Runs

- run = infinite sequence of global states.
- (r, k) = kth point in time in run r (k an integer).
  - r(k) = global state at time k in run r.
  - $r_i(k)$ = local state of P_i at time k in run r
  - r(0) indicates first state of the current epoch (the first state of the current authentication). Call it the initial state.
- system = set *R* of runs (typically the set of executions of a given protocol)

State Variables

- For each P_i
  - local history
    - sequence of all actions P_i has ever performed
  - key set
- For P_e
  - global history
    - sequence of actions any principal has ever performed
  - key set
  - message buffers, one per P_i containing all messages sent but not yet delivered to P_i

Set of Actions Per Principal

send(m, Q): This denotes Ps' sending of the message m to Q. Adds m to Q's buffer.

receive(): This denotes P's receipt of a message, nondeterministically chosen from P's message buffer.

newkey(K): This denotes P's coming into possession of a new key. Adds K to P's key set.

- Bookkeeping
  - Each action also adds itself to local and global histories.
  - Tag receive(m) by m for local and by m and P for global.

Auxiliary State Information

- State variables
  - messages P has received
  - messages P has sent
- State functions, given key set *K*, set *M* of received messages, define inductively:
  - seen-submsgs$_K$(M)
    - components of M that P can read
  - said-submsgs$_{K,M}$(M)
    - components of M that P has said as the result of sending M
    - N.B.: If P has said $\{M\}_K$ and P holds K used to encrypt the message, then P is considered to have said M as well. This is a severe assumption since P could just be forwarding $\{M\}_K$. P is held accountable for what he forwards.

Syntactic Restrictions on Runs

- A principal's key set never decreases.
- A message must be sent before received.
- A principal must possess keys it uses for encryption.
- A system principal (not P_e) sets from fields correctly.
- A system principal (not P_e) must see messages it forwards.

Semantics to Logic

Fix system *R*, and fix an *interpretation* π of primitive propositions that maps each p in Φ to the set of points $\pi(p)$ in *R* at which p is true. Define truth of Φ at (r, k), denoted

$$(r, k) \models \Phi$$

$(r, k) \models p$ iff (r, k) in $\pi(p)$ for primitive p in Φ

$(r, k) \models \Phi \& \Psi$ iff $(r, k) \models \Phi$ and $(r, k) \models \Psi$

$(r, k) \models \sim \Phi$ iff $(r, k) \models \Phi$

Semantics (I)

Seeing

(r, k) |= P **sees** X

iff, for some message M, at time k in r

1. receive(M) appears in P's local history, and
2. X in seen-submsgs$_K$(M), where K is P's key set.

Saying

(r, k) |= P **said** X

iff, for some message M, at some time k' <= k in r

1. P performs send(M, Q), and
2. X in said-submsgs$_{K,M}$(M), where *K* is P's key set and *M* is the set of messages P has received.

Semantics (II)

Jurisdiction

(r, k) |= P **controls** Φ

iff (r, k') |= P says Φ => (r, k') |= Φ for all k' >= 0

P **controls** Φ is shorthand for P **says** Φ => Φ

Freshness

(r, k) |= **fresh**(X) iff X not in *submsgs*(*M*(r, 0))

Semantics (III)

Shared keys

$(r, k) \models P \overset{K}{\leftrightarrow} Q$

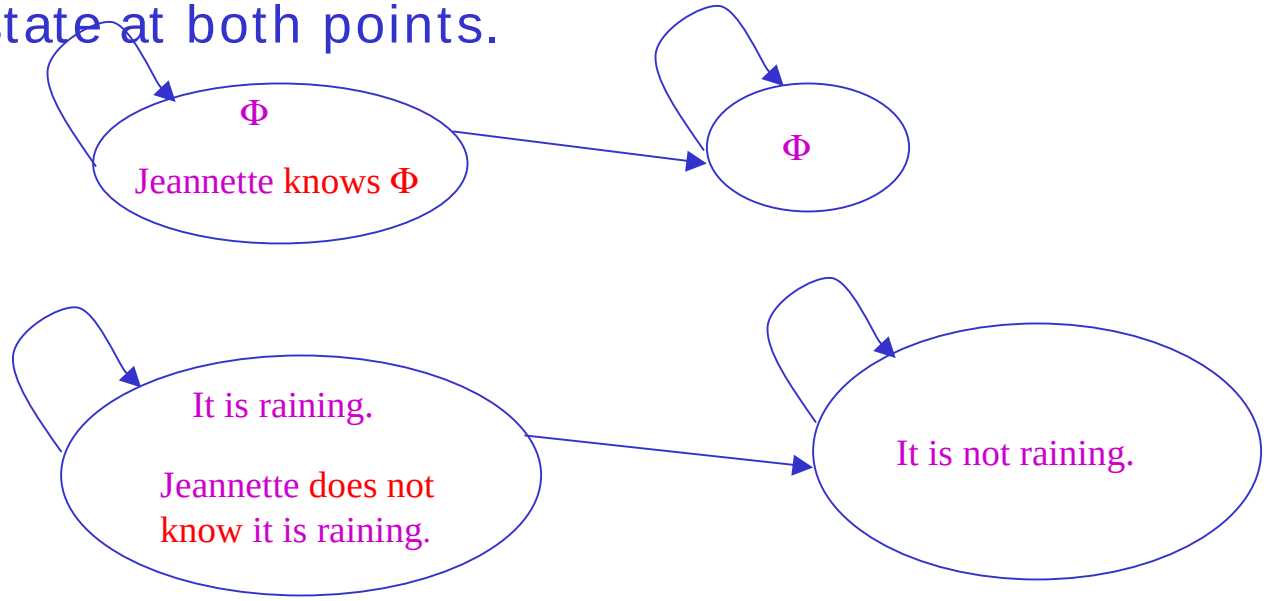
iff, for all k', $(r, k') \models$ R said $\{X^S\}_K$ implies either

$(r, k') \models$ R sees $\{X^S\}_K$ or R in {P, Q}

Shared secrets ...

Possible Worlds Semantics

In any given world (at any point in time), a principal P considers a number of other worlds to be possible. A world (point) is possible if the two points are indistinguishable to P, meaning that P has the same local state at both points.



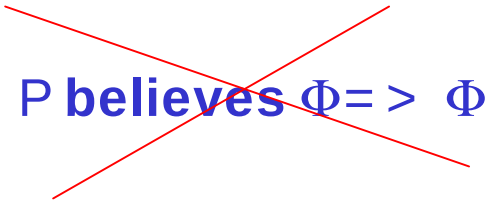
Knowledge vs. Belief

Knowledge logics have this axiom:

$$P \textbf{ knows } \Phi => \Phi$$

Knowledge = "truth in all possible worlds" [HalpernMoses84]

Belief logics do not:



Why not? P can believe in something (K is good) at the current point but where at a different point, indistinguishable to P (K still looks good to P), that something is not true (a malicious environment stumbled upon K so K is not really good).

Knowledge vs. Belief

Knowledge and belief logics are incomparable [BAN89]:

- Erroneous initial beliefs are certainly not knowledge.
- Each principal knows all tautologies, but does not necessarily believe them.

Possible Worlds Semantics For Belief Logics

For BAN belief, the points a principal considers possible are those points of good runs that the principal considers indistinguishable (~) from the current point after hiding the encrypted messages.

A *good* run is one in which initial beliefs are initially true,

Semantics (IV)

Belief

(r, k) |= P_i **believes** Φ

iff (r', k') |= Φ for all (r', k') such that (r, k) $\sim_i$ (r', k')

where

(r, k) $\sim_i$ (r', k') iff r' is a "good" run and $r_i(k) = r'_i(k')$ are the same

after removing unreadable messages.

Other Models

- Meadows (NRL Protocol Analyzer)
  - Based on Dolev-Yao term rewriting
    - Protocol specified as rules generating words in a term algebra
  - Model of computation is similar to Abadi-Tuttle
    - changeset, instead of (just) newkey
    - key sets can grow and shrink (hence non-monotonic)
    - can explicitly model multiple simultaneous runs
- Heintze-Tygar
  - Model of computation is more general than Abadi-Tuttle
    - nondeterministic communicating state machines, trace semantics
    - partial order of events (no global clock; events are asychronous).
    - beliefs and nonces can expire

Other Models (cont'd)

- Woo-Lam
  - Syntax of protocols in terms of terms (as in Dolev-Yao)
  - Semantics in terms of state machines
    - global and local states, state transitions, executions, etc.
  - Addresses not just secrecy, but correspondence:
    - The execution of the different principals proceeds in a lock-step fashion.
      - When an authenticating principal finishes its part of the protocol, the authenticated principal must have been present and participated in its part of the protocol.
    - Want to assure that the authenticating principal is indeed "talking" to the principal it has in mind.