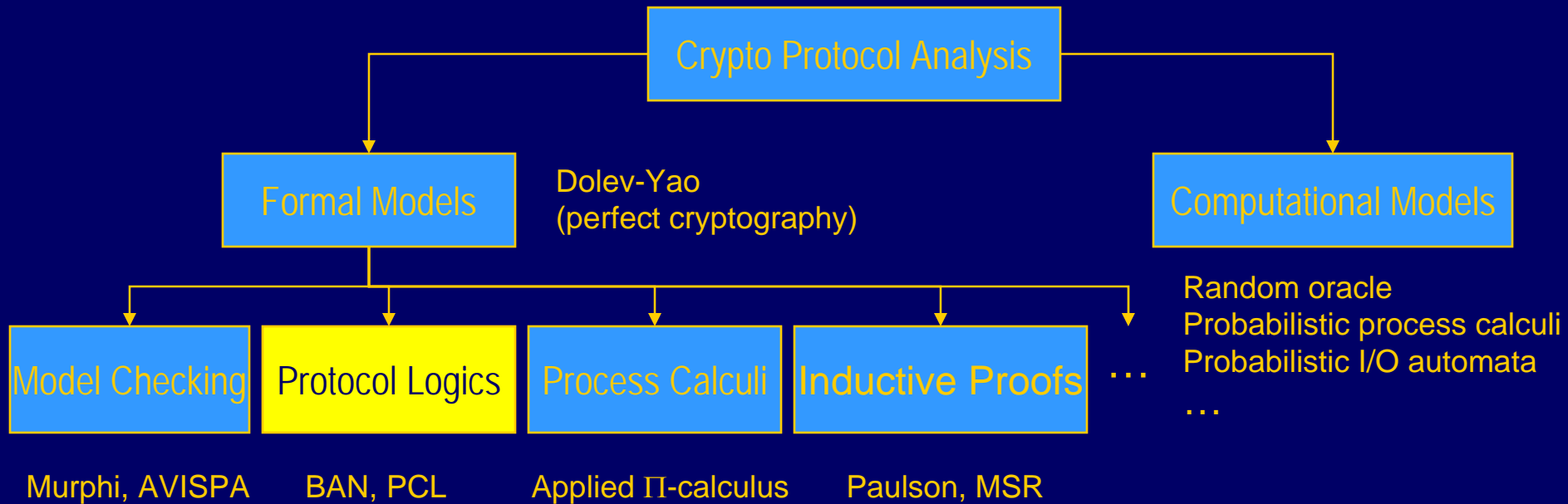


15-819: Logics and Languages for Security

Protocol Composition Logic (PCL)

Anupam Datta
Fall 2007-08

Protocol Analysis Techniques



Protocol Logics

◆ BAN Logic

A Logic of Authentication by Michael Burrows, Martin Abadi, Roger Needham (1989)

- ◆ Historically, the first logic for reasoning about security protocols
- ◆ Syntax and proof system (axioms and rules) for proving authentication properties (semantics added in a later paper)

BAN Logic (1)

◆ Advantages

- Proofs are relatively short (~ 2-3 pages)
 - cf. Paulson's inductive proofs
- Proofs follow protocol design intuition
 - cf. model-checking, low-level theorem-proving
- Relatively easy to use
 - Still taught widely in security courses
- No explicit reasoning about traces and intruder
 - cf. Paulson's inductive proofs

BAN Logic (2)

◆ Disadvantages

- Not sound wrt now accepted model of protocol execution and attack
 - Protocols "proved" secure may be insecure
e.g. NS was proved secure using BAN
- Protocols are modeled using logical formulas (*idealization step*) as opposed to state machines or programs
- Many uses of non-standard logical concepts
 - Jurisdiction, control, "belief", messages = propositions
- Only authentication properties, not secrecy
- Applicable to restricted classes of protocols

See Harper's slides on BAN from last lecture

Today

◆ Protocol Composition Logic (PCL)

- Developed over the last few years (2001-07)
- Retain advantages of BAN; rectify deficiencies
- Semantic model similar to Paulson's Inductive Method
- New proof techniques
 - Modular proofs
 - Cryptographic soundness

Protocol Composition Logic

- ◆ A logic for proving security of network protocols
- ◆ Illustrates use of programming language methods in computer security
 - Concurrency theory
 - Network protocols are *concurrent* programs
 - Floyd-Hoare style logic
 - Before-after assertions

Roadmap

◆ Intuition

◆ Formalism

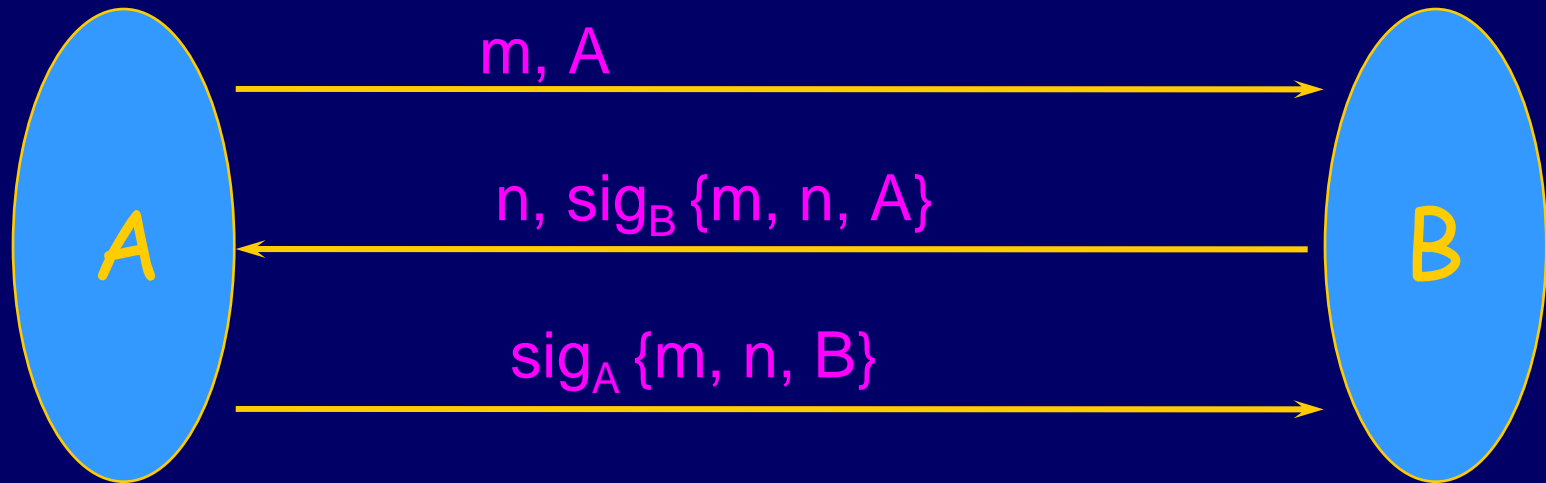
- Protocol programming language
- Protocol logic
- Proof System

◆ Example

- Signature-based challenge-response

◆ Proof techniques

Example: Challenge-Response



- ◆ *Alice reasons: if Bob is honest, then:*
 - only *Bob* can generate his signature
 - if *Bob* generates a signature of the form $\text{sig}_B\{m, n, A\}$,
 - he sends it as part of msg2 of the protocol, and
 - he must have received msg1 from *Alice*
- ◆ *Alice deduces: Received (B, msg1) \wedge Sent (B, msg2)*

Formalizing the Approach

- ◆ Language for protocol description
 - Arrows-and-messages are informal.
- ◆ Protocol Operational Semantics
 - How does the protocol execute?
- ◆ Protocol logic
 - Stating security properties.
- ◆ Proof system
 - Formally proving security properties.

Protocol Programming Language

◆ A protocol is described by specifying a "program" for each role

- Server = [receive x; new n; send {x, n}]

◆ Building blocks

- Terms (think "messages")

- names, nonces, keys, encryption, ...

- Actions (operations on terms)

- send, receive, pattern match, ...

Terms

$t ::= c$	constant term
x	variable
N	name
K	key
t, t	tupling
$\text{sig}_K\{t\}$	signature
$\text{enc}_K\{t\}$	encryption

Example: $x, \text{sig}_B\{m, x, A\}$ is a term

Typed term algebra

Actions

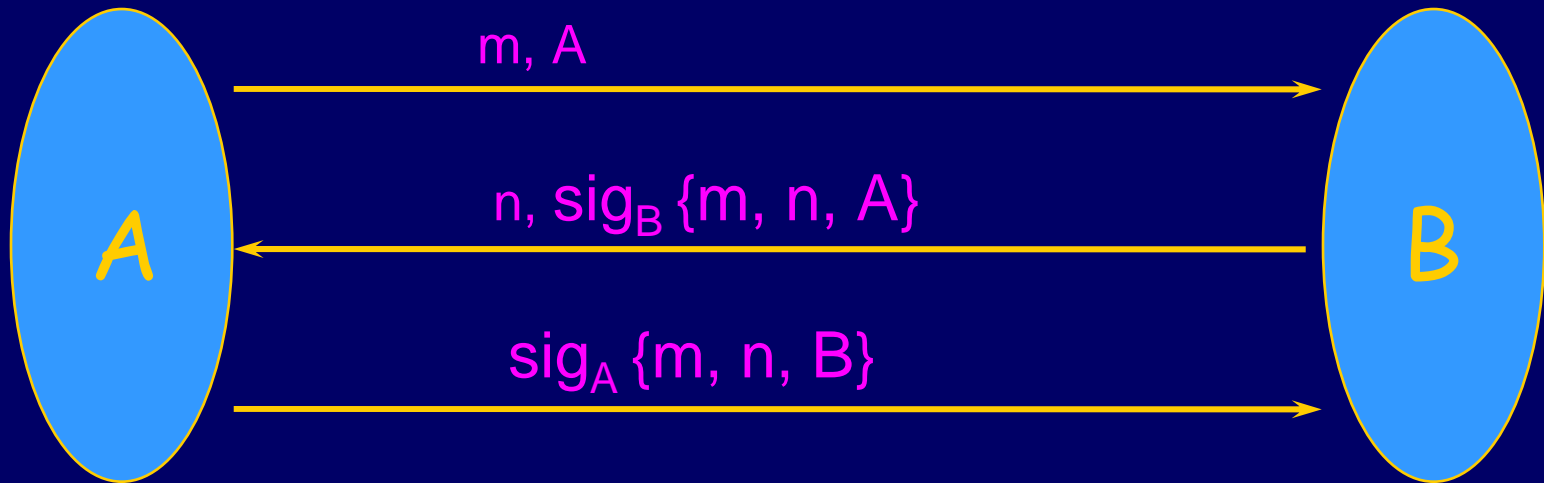
send t; send a term t
receive x; receive a term into variable x
match t/p(x); match term t against p(x)

◆ A program or cord is a sequence of actions

◆ Notation:

- we often omit match actions
- receive $\text{sig}_B\{A, n\} = \text{receive } x; \text{ match } x/\text{sig}_B\{A, n\}$

Challenge-Response Programs

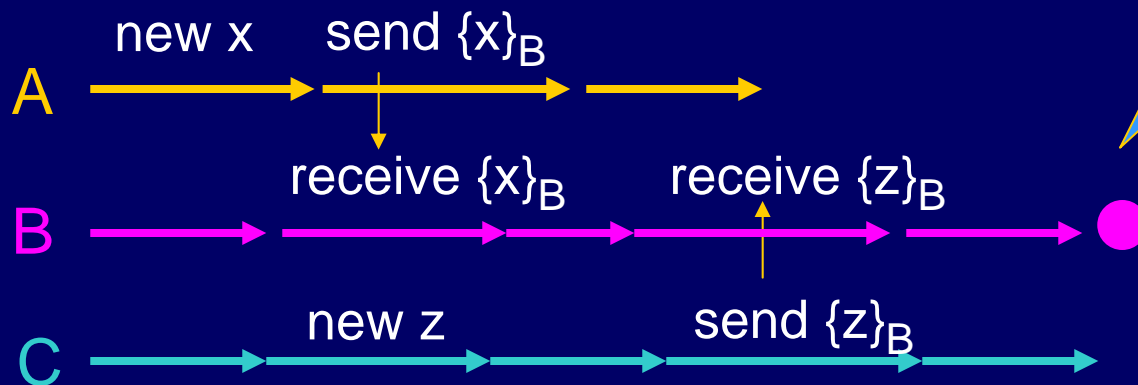


```
InitCR(A, X) = [  
  new m;  
  send A, X, {m, A};  
  receive X, A, {x, sig_X {m, x, A}};  
  send A, X, sig_A {m, x, X};  
]
```

```
RespCR(B) = [  
  receive Y, B, {y, Y};  
  new n;  
  send B, Y, {n, sig_B {y, n, Y}};  
  receive Y, B, sig_Y {y, n, B};  
]
```

Protocol Execution

- Initial configuration
 - Protocol is a finite set of roles
 - Set of principals and keys
 - Assignment of ≥ 1 role to each principal
- Run (trace)



Process
calculus
operational
semantics

Process Calc. Op. Semantics

◆ Cord space is a multiset of cords

◆ Cords may react

- via communication
- via internal actions

◆ Sample reaction steps:

• Communication:

$$[\text{send } t; S]_x \mid [\text{receive } x; T]_y \Rightarrow [S]_x \mid [T(t/x)]_y$$

• Matching:

$$[\text{match } p(t)/p(x); S]_x \Rightarrow [S(t/x)]_x$$

Reaction steps

- (1) $[\text{receive } x; S]_X \mid [\text{send } t; T]_Y \longrightarrow [S(t/x)]_X \mid [T]_Y$
- (2) $[\text{match } p(t)/p(x); S]_X \longrightarrow [S(t/x)]_X$
- (3) $[\text{new } x; S]_X \longrightarrow [S(m/x)]_X$
- (4) $[x := \text{enc } t, K; S]_X \longrightarrow [S(\text{ENC}_K\{t\}/x)]_X$
- (5) $[x := \text{dec } \text{ENC}_K\{t\}, K; S]_X \longrightarrow [S(t/x)]_X$
- (6) $[x := \text{sign } t, K; S]_X \longrightarrow [S(\text{SIG}_K\{t\}/x)]_X$
- (7) $[\text{verify } \text{SIG}_K\{t\}, t, K; S]_X \longrightarrow [S]_X$

Where the following conditions must be satisfied:

- (1) $FV(t) = \emptyset$
- (2) $m \notin FV(C) \cup FV(S)$, where C is the entire cord space

Table 3
Basic reaction steps

Attacker capabilities

◆ Controls complete network

- Can read, remove, inject messages

◆ Fixed set of operations on terms

- Pairing
- Projection
- Encryption with known key
- Decryption with known key
- ...

Commonly referred to as “Dolev-Yao” attacker

PCL: Syntax

◆ Action formulas

$a ::= \text{Send}(P,t) \mid \text{Receive}(P,t) \mid \text{Verify}(P,T) \mid \dots$

◆ Formulas

$\varphi ::= a \mid \text{Has}(P,t) \mid \text{Honest}(N) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x \varphi$
 $\mid a < a \mid \dots$

◆ Modal formula

$\varphi [\text{actions}]_P \varphi$

◆ Example

$\text{Has}(X, \text{secret}) \supset (X = A \vee X = B)$

Specifying secrecy

Challenge-Response Property

◆ Specifying authentication for Initiator

$$\text{true [InitCR(A, B)]}_A \text{ Honest(B) } \supset$$
$$\left(\begin{array}{l} \text{Send(A, \{A,B,m\})} < \\ \text{Receive(B, \{A,B,m\})} < \\ \text{Send(B, \{B,A,\{n, sig}_B \{m, n, A\}\})} < \\ \text{Receive(A, \{B,A,\{n, sig}_B \{m, n, A\}\})} \\ \end{array} \right)$$

Semantics: Property must hold in all protocol traces (similar to Paulson's Inductive Method)

PCL: Semantics

◆ Protocol Q

- Defines set of roles (e.g, initiator, responder)
- Run R of Q is sequence of actions by principals following roles, plus attacker

◆ Satisfaction

- $Q, R \models \theta [\textit{actions}]_P \phi$

If some role of P in R does exactly *actions* starting from state where θ is true, then ϕ is true in state after *actions* completed irrespective of actions executed by other agents concurrently

- $Q \models \theta [\textit{actions}]_P \phi$

$Q, R \models \theta [\textit{actions}]_P \phi$ for all runs R of Q

Semantics of action formulas

$\mathcal{Q}, R \models \text{Send}(A, m)$ if in the run R , thread A executed action `send` m .

$\mathcal{Q}, R \models \text{Receive}(A, m)$ if there exists a variable x such that in the run R , thread A executed action `receive` x , receiving data m into variable x .

$\mathcal{Q}, R \models \text{New}(A, m)$ if there exists a variable x such that in the run R , thread A executed action `new` x , receiving data m into variable x .

$\mathcal{Q}, R \models \text{Encrypt}(A, ENC_K\{m\})$ if there exists a variable x such that in the run R , thread A executed action $x := \text{enc } m, K$, receiving data $ENC_K\{m\}$ into variable x .

$\mathcal{Q}, R \models \text{Decrypt}(A, ENC_K\{m\})$ if there exists a variable x such that in the run R , thread A executed action $x := \text{dec } ENC_K\{m\}, K$, receiving data m into variable x .

$\mathcal{Q}, R \models \text{Sign}(A, SIG_K\{m\})$ if there exists a variable x such that in the run R , thread A executed action $x := \text{sign } m, K$, receiving data $SIG_K\{m\}$ into variable x .

$\mathcal{Q}, R \models \text{Verify}(A, SIG_K\{m\})$ if in the run R , thread A executed the signature verification action `verify` $SIG_K\{m\}, K$.

Semantics of other formulas

$Q, R \models \text{Start}(A)$ if $R|_A$ is empty. Intuitively this formula means that A didn't execute any actions in the past.

$Q, R \models \text{Fresh}(A, t)$ if $Q, R \models \text{New}(A, t)$ holds and furthermore for all m such that $Q, R \models \text{Send}(A, m)$ it holds that t is not a subterm of m .

$Q, R \models \text{Gen}(A, t)$ if there is a prefix R' of R such that $Q, R' \models \text{Fresh}(A, t)$ holds.

$Q, R \models \text{FirstSend}(A, t, t')$ if t is a subterm of t' , $Q, R \models \text{Send}(A, t')$ holds and for all prefixes R' of R and all terms t'' such that $t \subseteq t''$ and $Q, R' \models \text{Send}(A, t'')$ it has to be that $Q, R' \models \text{Send}(A, t')$.

$Q, R \models \text{Honest}(\hat{A})$ if $\hat{A} \in \text{HONEST}(C)$ in initial configuration C for R and all threads of \hat{A} are in a “pausing” state in R . More precisely, $R|_{\hat{A}}$ is an interleaving of basic sequences of roles in Q .

$Q, R \models \text{Contains}(t_1, t_2)$ if $t_2 \subseteq t_1$, where \subseteq is the subterm relation between terms.

$Q, R \models (\phi_1 \wedge \phi_2)$ if $Q, R \models \phi_1$ and $Q, R \models \phi_2$

$Q, R \models \neg\phi$ if $Q, R \not\models \phi$

$Q, R \models \exists x.\phi$ if $Q, R \models (d/x)\phi$, for some d , where $(d/x)\phi$ denotes the formula obtained by substituting d for x in ϕ .

Proof System

- ◆ Goal: formally prove security properties
- ◆ Axioms
 - Simple formulas provable by hand
- ◆ Inference rules
 - Proof steps
- ◆ Theorem
 - Formula obtained from axioms by application of inference rules

Sample axioms about actions

◆ New data

- $\text{true} [\text{new } x]_p \text{ Has}(P, x)$
- $\text{true} [\text{new } x]_p \text{ Has}(Y, x) \supset Y=P$

◆ Actions

- $\text{true} [\text{send } m]_p \text{ Send}(P, m)$

◆ Verify

- $\text{true} [\text{match } x/\text{sig}_x\{m\}]_p \text{ Verify}(P, m)$

Reasoning about knowledge

◆ Pairing

- $\text{Has}(X, \{m, n\}) \supset \text{Has}(X, m) \wedge \text{Has}(X, n)$

◆ Encryption

- $\text{Has}(X, \text{enc}_K(m)) \wedge \text{Has}(X, K^{-1}) \supset \text{Has}(X, m)$

Encryption and signature

◆ Public key encryption

$\text{Honest}(X) \wedge \text{Decrypt}(Y, \text{enc}_X\{m\}) \supset X=Y$

◆ Signature

$\text{Honest}(X) \wedge \text{Verify}(Y, \text{sig}_X\{m\}) \supset$

$\exists m' (\text{Send}(X, m') \wedge \text{Contains}(m', \text{sig}_X\{m\}))$

Sample inference rules

◆ First-order logic rules

$$\frac{\varphi \quad \varphi \supset \theta}{\theta}$$

◆ Generic rules

$$\frac{\theta [\text{actions}]_p \psi \quad \theta [\text{actions}]_p \varphi}{\theta [\text{actions}]_p \psi \wedge \varphi}$$

Honesty rule

(example use)

\forall roles R of Q . \forall protocol steps A of R .

$$\frac{\text{Start}(X) []_X \phi \quad \phi [A]_X \phi}{Q \vdash \text{Honest}(X) \supset \phi}$$

- Example use:

- If Y receives a message m from X , and
- $\text{Honest}(X) \supset (\text{Sent}(X,m) \supset \text{Received}(X,m'))$
- then Y can conclude
 $\text{Honest}(X) \supset \text{Received}(X,m')$

Proved using
honesty rule

Looks like Owicki-Gries non-interference, but
semantically different because of attacker

Correctness of CR

InitCR(A, X) = [

new m;

send A, X, {m, A};

receive X, A, {x, sig_X{m, x, A}};

send A, X, sig_A{m, x, X};

]

RespCR(B) = [

receive Y, B, {y, Y};

new n;

send B, Y, {n, sig_B{y, n, Y}};

receive Y, B, sig_Y{y, n, B};

]

CR |- true [InitCR(A, B)]_A Honest(B) \supset

Send(A, {A, B, m}) <

Receive(B, {A, B, m}) <

Send(B, {B, A, {n, sig_B{m, n, A}}}) <

Receive(A, {B, A, {n, sig_B{m, n, A}}})

} Auth

Correctness of CR - step 1

InitCR(A, X) = [

new m;

send A, X, {m, A};

receive X, A, {x, sig_X{m, x, A}};

send A, X, sig_A{m, x, X}};

]

RespCR(B) = [

receive Y, B, {y, Y};

new n;

send B, Y, {n, sig_B{y, n, Y}};

receive Y, B, sig_Y{y, n, B}};

]

1. A reasons about her own actions

CR \vdash true [InitCR(A, B)]_A

Verify(A, sig_B{m, n, A})

Correctness of CR - step 2

InitCR(A, X) = [

 new m;

 send A, X, {m, A};

 receive X, A, {x, sig_X{m, x, A}};

 send A, X, sig_A{m, x, X};

]

RespCR(B) = [

 receive Y, B, {y, Y};

 new n;

 send B, Y, {n, sig_B{y, n, Y}};

 receive Y, B, sig_Y{y, n, B};

]

2. Properties of signatures

CR \vdash true [InitCR(A, B)]_A Honest(B) \supset

$\exists m' (\text{Send}(B, m') \wedge \text{Contains}(m', \text{sig}_B \{m, n, A\}))$

Recall signature axiom

Correctness of CR - Honesty

InitCR(A, X) = [

new m;

send A, X, {m, A};

receive X, A, {x, sig_X{m, x, A}};

send A, X, sig_A{m, x, X};

]

RespCR(B) = [

receive Y, B, {y, Y};

new n;

send B, Y, {n, sig_B{y, n, Y}};

receive Y, B, sig_Y{y, n, B};

]

Invariant proved with Honesty rule

$CR \vdash \text{Honest}(X) \wedge$

$\text{Send}(X, m') \wedge \text{Contains}(m', \text{sig}_x\{y, x, Y\}) \wedge \neg \text{New}(X, y) \supset$

$m = X, Y, \{x, \text{sig}_B\{y, x, Y\}\} \wedge \text{Receive}(X, \{Y, X, \{y, Y\}\})$

Induction over protocol steps

Correctness of CR - step 3

InitCR(A, X) = [

new m;

send A, X, {m, A};

receive X, A, {x, sig_X{m, x, A}};

send A, X, sig_A{m, x, X};

]

RespCR(B) = [

receive Y, B, {y, Y};

new n;

send B, Y, {n, sig_B{y, n, Y}};

receive Y, B, sig_Y{y, n, B};

]

3. Use Honesty invariant

$CR \vdash \text{true} [\text{InitCR}(A, B)]_A \text{Honest}(B) \supset$
 $\text{Receive}(B, \{A, B, m\}), \dots$

Correctness of CR - step 4

InitCR(A, X) = [

 new m;

 send A, X, {m, A};

 receive X, A, {x, sig_X{m, x, A}};

 send A, X, sig_A{m, x, X};

]

RespCR(B) = [

 receive Y, B, {y, Y};

 new n;

 send B, Y, {n, sig_B{y, n, Y}};

 receive Y, B, sig_Y{y, n, B};

]

4. Use properties of nonces for temporal ordering

CR \vdash true [InitCR(A, B)]_A Honest(B) \supset Auth

Nonces are “fresh” random numbers

Freshness and temporal ordering rules

AN3 $\top[\text{new } x]_X \text{ Fresh}(X, x)$

AN4 $\text{Fresh}(X, x) \supset \text{Gen}(X, x)$

P2 $\text{Fresh}(X, t)[a]_X \text{ Fresh}(X, t)$
where $t \not\subseteq a$.

FS1 $\text{Fresh}(X, t)[\text{send } t']_X \text{ FirstSend}(X, t, t')$
where $t \subseteq t'$.

FS2 $\text{FirstSend}(X, t, t') \wedge a(Y, t'') \supset \text{Send}(X, t') < a(Y, t'')$
where $X \neq Y$ and $t \subseteq t''$.

Compare with BAN

We have a proof. So what?

◆ Soundness Theorem:

- if $Q \vdash \phi$ then $Q \models \phi$
- If ϕ is a theorem then ϕ is a valid formula

◆ ϕ holds in any step in any run of protocol Q

- Unbounded number of participants
- Dolev-Yao intruder

PCL Proof Techniques

- ◆ Modular Proofs
- ◆ Generic Template-style Proofs

Modular Analysis / Composition



Laptop



Access Point



Auth Server

EAP-TLS: Certificates to Authorization (PMK)

4WAY Handshake:

PMK to Keys for data communication

Group key:

Keys for broadcast communication

Data protection:

AES based using above keys

← (Shared Secret-PMK)

802.11i Key Management
≈20 msgs in 4 components

[HSDDM CCS'05 ->
TISSEC Special Issue]

Compositional Proofs: Intuition

◆ Protocol specific reasoning

- “if honest Bob generates a signature of the form $\text{sig}_B\{m, n, A\}$,
 - he sends it as part of msg2 ...”
- Could break: Bob’s signature from one protocol could be used to attack another
- PCL proof system: Invariant rule

◆ Protocol independent reasoning

- Axiom stating unforgeability of signatures
- Still good: unaffected by composition
- All other axioms and proof rules for PCL

Generic Template-style Proofs

- ◆ Protocols with *function variables* instead of specific cryptographic operations
 - One template can be instantiated to many protocols
 - Proof of template yields proofs for instances
- ◆ Motivating example:
 - IKEv2: two instances based on symmetric and public-key cryptography

Gets at abstract protocol concepts without taking the BAN approach of confusing messages with propositions

Protocol Template

Challenge-Response

Template

$$\begin{aligned} A &\rightarrow B: m \\ B &\rightarrow A: n, \underline{F}(B, A, n, m) \\ A &\rightarrow B: \underline{G}(A, B, n, m) \end{aligned}$$
$$\begin{aligned} A &\rightarrow B: m \\ B &\rightarrow A: n, \underline{E}_{KAB}(n, m, B) \\ A &\rightarrow B: \underline{E}_{KAB}(n, m) \end{aligned}$$

ISO-9798-2

$$\begin{aligned} A &\rightarrow B: m \\ B &\rightarrow A: n, \underline{H}_{KAB}(n, m, B) \\ A &\rightarrow B: \underline{H}_{KAB}(n, m, A) \end{aligned}$$

SKID3

$$\begin{aligned} A &\rightarrow B: m \\ B &\rightarrow A: n, \underline{\text{sig}}_B(n, m, A) \\ A &\rightarrow B: \underline{\text{sig}}_A(n, m, B) \end{aligned}$$

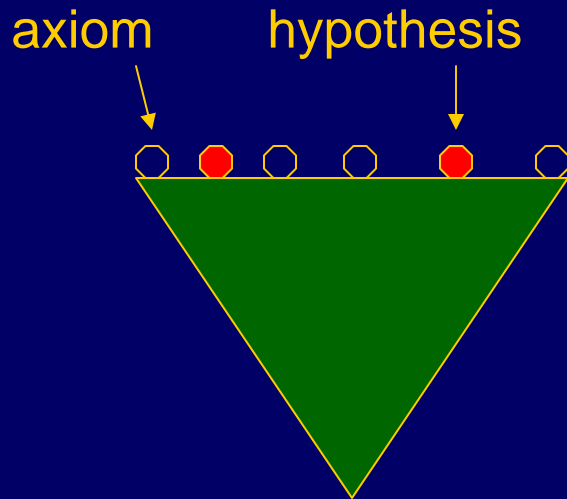
ISO-9798-3

Instantiations

Template Proof Method

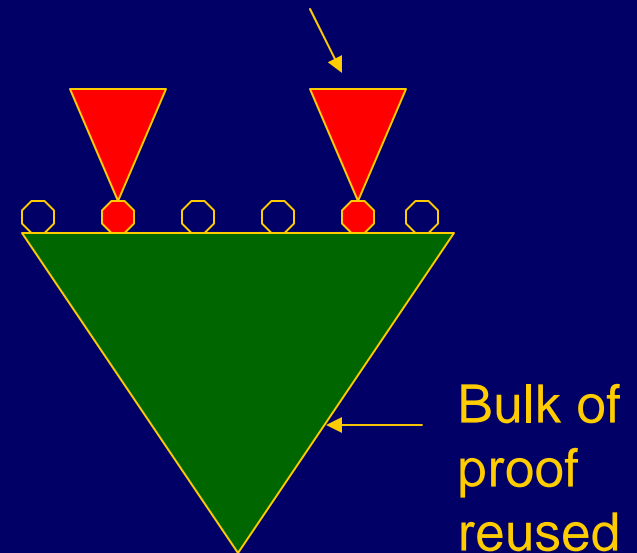
- ◆ **Characterizing protocol concepts**
 - Step 1: Under hypotheses about function variables and invariants, prove security property of template
 - Step 2: Instantiate function variables to cryptographic operations and prove hypotheses.
- ◆ **Benefit:**
 - Proof reuse
- ◆ **Single protocol can be instance of multiple templates allowing modular proofs**

Proof Structure



Templat
e

Additional work to
discharge hypotheses



Instance

Extending Formalism

◆ Language Extensions

- Add function variables to term language for cords and logic (HOL)

◆ Semantics

- $Q \models \varphi \Leftrightarrow \sigma Q \models \sigma\varphi$, for all substitutions σ eliminating all function variables

◆ Soundness Theorem

- Every provable formula is valid

Summary

◆ PCL - Logic for security protocols

- *Sound* wrt symbolic and cryptographic models
- High-level *short* proofs: 2-3 pages

◆ Proof techniques

- Modular/compositional proofs
- Generic template-style proofs

◆ Proofs of industrial protocols

- IEEE 802.11i (w/ TLS), Kerberos, GDOI, IKEv2, Mobile IPv6 (in progress)

◆ Mechanization not done

- We are getting started here
- Some work at Kestrel, Stanford

Thanks !

Questions?