Compositional Protocol Logic
Outline

- Floyd-Hoare logic of programs
  - Compositional reasoning about properties of programs
- DDMP protocol logic
  - Developed by Datta, Derek, Mitchell, and Pavlovic for logical reasoning about security properties
Floyd-Hoare Logic

◆ **Main idea:** before-after assertions
  - $F \mathsf{<P>} G$
    - If $F$ is true before executing $P$, then $G$ is true after

◆ **Total correctness or partial correctness**
  - **Total correctness:** $F \mathsf{[P]} G$
    - If $F$ is true, then $P$ will halt and $G$ will be true
  - **Partial correctness:** $F \mathsf{\{P\}} G$
    - If $F$ is true and if $P$ halts, then $G$ will be true
While Programs

\[ P ::= \]
\[ x := e \mid \]
\[ P ; P \mid \]
\[ \text{if } B \text{ then } P \text{ else } P \mid \]
\[ \text{while } B \text{ do } P \]

where \( x \) is any variable

\( e \) is any integer expression

\( B \) is a Boolean expression (true or false)
Assignment and Rule of Consequence

◆ Assignment axiom: \( F(t) \ { x := t } \ F(x) \)
  - If \( F \) holds for \( t \), and \( t \) is assigned to \( x \), then \( F \) holds for \( x \) afterwards
  - This assumes that there is no aliasing!
  - Examples:
    - \( 7=7 \ { x := 7 } \ x=7 \)
    - \( (y+1)>0 \ { x := y+1 } \ x>0 \)
    - \( x+1=2 \ { x := x+1 } \ x=2 \)

◆ Rule of consequence:
  If \( F \ { P } \ G \) and \( F' \rightarrow F \) and \( G \rightarrow G' \),
  then \( F' \ { P } \ G' \)
Simple Examples

 Assertion: \( y > 0 \) \{ x := y+1 \} \ x > 0

 Proof:

 \[(y+1) > 0 \] \{ x := y+1 \} \ x > 0 \quad \text{(assignment axiom)}
\[y > 0 \quad \{ x := y+1 \} \ x > 0 \quad \text{(rule of consequence)}
\]

 Assertion: \( x = 1 \) \{ x := x+1 \} \ x = 2

 Proof:

 \[x+1 = 2 \quad \{ x := x+1 \} \ x = 2 \quad \text{(assignment axiom)}
\[x = 1 \quad \{ x := x+1 \} \ x = 2 \quad \text{(rule of consequence)}
\]
Conditional

\[ F \& B \{ P \} G \]
\[ F \& \neg B \{ Q \} G \]
\[ \underline{F \{ \text{if } B \text{ then } P \text{ else } Q \} G} \]

• Example:

\[ \text{true } \{ \text{if } y \geq 0 \text{ then } x := y \text{ else } x := -y \} \ x \geq 0 \]
Sequence

\[
\begin{align*}
F \{P\} G \\
G \{Q\} H \\
F \{P; Q\} H
\end{align*}
\]

• Example:

\[
x=0 \{x := x+1 ; x := x+1\} \quad x=2
\]
Loop Invariant

\[ F \land B \{ P \} F \]

\[ F \{ \text{while } B \text{ do } P \} F \land \neg B \]

• Example:

\[ \text{true} \{ \text{while } x \neq 0 \text{ do } x := x-1 \} x=0 \]
Example: Compute $d = x - y$

 Assertion: $y \leq x$ \[ \{d := 0; \text{ while } (y+d) < x \text{ do } d := d + 1\} y+d=x \]  

 Proof:

 - Choose loop invariant $F = y+d \leq x$
  
  \[
  y+d \leq x \land B \quad \{Q\} \quad y+d \leq x \]

  \[
  y+d \leq x \quad \{\text{while } B \text{ do } Q\} \quad y+d \leq x \land \neg B
  \]

  Important: proving a property of the entire loop has been reduced to proving a property of one iteration of the loop

 - To prove $y+d \leq x \land B \quad \{Q\} \quad y+d \leq x$, use assignment axiom and sequence rule

After loop execution, $y+d \leq x \land \neg(y+d < x)$, thus $y+d = x$
Goal: Logic for Security Protocols

◆ “Floyd-Hoare” reasoning about security properties
  • Would like to derive global properties of protocols from local assertions about each protocol participant
  • Use a rigorous logical framework to formalize the reasoning that each participant carries out

◆ Compositionality is important
  • Security properties must hold even if the protocol is executed in parallel with other protocols
  • Compositionality is the main advantage of process calculi and protocol logics
Intuition

◆ Reason about local information
  • I chose a fresh, unpredictable number
  • I sent it out encrypted
  • I received it decrypted
  • Therefore: someone decrypted it

◆ Incorporate knowledge about protocol into reasoning
  • According to the protocol specification, server only sends $m$ if it received $m'$
  • If server not corrupt and I receive $m$ signed by server, then server received $m'$
Alice’s “View” of the Protocol

- Protocol spec
- Private data
- Sent and received messages
- Honest principals, attacker
Example: Challenge-Response

Alice’s reasoning:
- If Bob is honest, then only Bob can generate his signature
- If honest Bob generates a signature of the form \( \text{sig}_B\{m, n, A\} \), then
  1. Bob must have received \( m, A \) from Alice
  2. Bob sent \( \text{sig}_B\{m, n, A\} \) as part of his 2\(^{nd}\) message

Alice concludes:
- Received(\(B, \text{msg1}\)) \& Sent(\(B, \text{msg2}\))
Protocol Composition Logic

- A formal language for describing protocols
  - Terms and actions instead of informal arrows-and-messages notation
- Operational semantics
  - Describe how the protocol executes
- Protocol logic
  - State security properties (in particular, secrecy and authentication)
- Proof system
  - Axioms and inference rules for formally proving security properties
Terms
\[ t ::= c \mid x \mid N \mid K \mid t, t \mid \text{constant} \]
\[ \text{variable} \]
\[ \text{name} \]
\[ \text{key} \]
\[ \text{tuple} \]
\[ \text{signature} \]
\[ \text{encryption} \]
Actions

- **new m** generated fresh value
- **send U, V, t** send term t from U to V
- **receive U, V, x** receive term and assign into variable x
- **match t/p(x)** match term t against pattern p(x)

◆ **A thread is a sequence of actions**
  - Defines the “program” for a protocol participant, i.e., what messages he sends and receives and the checks he performs
  - For notational convenience, omit “match” actions
    - Write “receive sig_B{A, n}” instead of “receive x; match x/sig_B{A, n}”
Challenge-Response Threads

InitCR(A, X) = [
    new m;
    send A, X, {m, A};
    receive X, A, {x, sig_A{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_B{y, n, B};
]
A protocol is a finite set of roles
- Initial configuration specifies a set of principals and keys; assignment of \( \geq 1 \) role to each principal

A run is a concurrent execution of the roles
- Models a protocol session
- Send and receive actions are matched up

Position in run:
- New action: \( \text{new } x \)
- Send action: \( \text{send}\{x\}_B \)
- Receive action: \( \text{receive}\{x\}_B \) and \( \text{receive}\{z\}_B \)
- New action: \( \text{new } z \)
- Send action: \( \text{send}\{z\}_B \)
Action Formulas

◆ Predicates over action sequences

\[
a ::= \text{Send}(X,m) \mid \text{Message } m \text{ was sent in thread } X
\]
\[
\text{Receive}(X,m) \mid \text{Message } m \text{ was received in thread } X
\]
\[
\text{New}(X,t) \mid \text{Term } t \text{ was generated as new in } X
\]
\[
\text{Decrypt}(X,t) \mid \text{Term } t \text{ was decrypted in thread } X
\]
\[
\text{Verify}(X,t) \mid \text{Term } t \text{ was verified in } X
\]
Formulas

\[ \varphi ::= a \quad | \quad \text{Action formula} \]
\[ \text{Has}(X,m) \quad | \quad \text{Thread X created m or received} \]
\[ \text{a message containing m and has} \]
\[ \text{keys to extract m from the message} \]
\[ \text{Fresh}(X,t) \quad | \quad \text{Term t hasn’t been “seen” outside X} \]
\[ \text{Honest}(N) \quad | \quad \text{Principal N follows protocol rules in} \]
\[ \text{all of its threads} \]
\[ \text{Contains}(t,t') \quad | \quad \text{Term t contains subterm t’} \]
\[ \neg \varphi \quad | \quad \varphi_1 \land \varphi_2 \quad | \quad \exists x \ \varphi \quad | \quad \Box \varphi \quad | \quad \Diamond \varphi \]
\[ \varphi \text{ was true} \]

Modal operator \[ [\text{actions}]_X \ \varphi \]
\[ \text{After actions, X reasons } \varphi \]
Trace Semantics

◆ Protocol Q
  • Defines a set of roles (e.g., initiator and responder)

◆ Run R
  • Sequence of actions by principals following protocol roles and the attacker (models a protocol session)

◆ Satisfaction
  • \( Q, R \models [ actions ]_p \phi \)
    - Some role of principal \( P \) in \( R \) performs exactly \( actions \) and \( \phi \) is true in the state obtained after \( actions \) complete
  • \( Q \models [ actions ]_p \phi \)
    - \( Q, R \models [ actions ]_p \phi \) for all runs \( R \) of \( Q \)
Specifying Authentication

Initiator authentication in Challenge-Response

CR |= [ InitCR(A, B) ]_A Honest(B) \supset
ActionsInOrder(
    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\}\}\})
)

After initiator executes his program

If B is honest...

...then msg sends and receives must have happened in order prescribed by protocol spec
Specifying Secrecy

◆ Shared secret in key establishment

After initiator executes his program

If B is honest...

\[ KE \models [\text{InitKE}(A, B)]_A \text{ Honest}(B) \supset (\text{Has}(X, m) \supset X=A \lor X=B) \]

...then if some party X knows secret m, then X can only be either A, or B
Goal: formally prove properties of security protocols

Axioms are simple formulas
  • Provable by hand

Inference rules are proof steps

Theorem is a formula obtained from axioms by application of inference rules
Sample Axioms

◆ New data
  • \( [\new x ]_P \text{ Has}(P,x) \)
  • \( [\new x ]_P \text{ Has}(Y,x) \supset Y=P \)

◆ Acquiring new knowledge
  • \( [\text{receive } m ]_P \text{ Has}(P,m) \)

◆ Performing actions
  • \( [\text{send } m ]_P \Diamond \text{Send}(P,m) \)
  • \( [\text{match } x/\text{sig}_x\{m\} ]_P \Diamond \text{Verify}(P,m) \)
Reasoning About Cryptography

◆ Pairing
  • Has(X, \{m,n\}) ⊃ Has(X, m) ∧ Has(X, n)

◆ Symmetric encryption
  • Has(X, enc_k(m)) ∧ Has(X, K^{-1}) ⊃ Has(X, m)

◆ Public-key encryption
  • Honest(X) ∧ ◇ Decrypt(Y, enc_X{m}) ⊃ X=Y

◆ Signatures
  • Honest(X) ∧ ◇ Verify(Y, sig_X{m}) ⊃
  \exists m' (◇ Send(X, m') ∧ Contains(m', sig_X{m})).
Sample Inference Rules

\[
\begin{align*}
\text{[ actions ]}_p \text{ Has}(X, t) \\
\text{[ actions; action ]}_p \text{ Has}(X, t)
\end{align*}
\]

\[
\begin{align*}
\text{[ actions ]}_p \phi & \quad \text{[ actions ]}_p \varphi \\
\hline
\text{[ actions ]}_p \phi \land \varphi
\end{align*}
\]
Honesty Rule

∀ roles R of Q. ∀ initial segments A ⊆ R.

\[ \begin{align*}
Q & \vdash [A]_X \phi \\
Q & \vdash \text{Honest}(X) \supset \phi
\end{align*} \]

- **Finitary rule** (finite number of premises to choose from)
  - Typical protocol has 2-3 roles, typical role has 1-3 actions
- **Example:**
  - If Honest(X) ⊃ (Sent(X,m) ⊃ Received(X,m')) and Y receives a message from X, then Y can conclude Honest(X) ⊃ Received(X,m')
Correctness of Challenge-Response

\[ \text{InitCR}(A, X) = [ \]
\[
\text{new m;}
\] \[
\text{send A, X, \{m, A\};}
\] \[
\text{receive X, A, \{x, sig}_{X}\{m, x, A\};}
\] \[
\text{send A, X, sig}_{A}\{m, x, X\};
\]\[
]
\[ \]

\[ \text{RespCR}(B) = [ \]
\[
\text{receive Y, B, \{y, Y\};}
\] \[
\text{new n;}
\] \[
\text{send B, Y, \{n, sig}_{B}\{y, n, Y\};}
\] \[
\text{receive Y, B, sig}_{Y}\{y, n, B\};
\]\[
]
\[ \]

\[ \text{CR} \vdash [ \text{InitCR}(A, B) \]_A \text{ Honest}(B) \supset \text{ActionsInOrder} (\]
\[
\text{Send}(A, \{A,B,m\}),
\] \[
\text{Receive}(B, \{A,B,m\}),
\] \[
\text{Send}(B, \{B,A,\{n, sig}_{B}\{m, n, A\}\}),
\] \[
\text{Receive}(A, \{B,A,\{n, sig}_{B}\{m, n, A\}\})
\] )
1: A Reasons about Own Actions

\[
\text{InitCR}(A, X) = [ \\
\quad \text{new m;} \\
\quad \text{send } A, X, \{m, A\}; \\
\quad \text{receive } X, A, \{x, \text{sig}_X\{m, x, A\}\}; \\
\quad \text{send } A, X, \text{sig}_A\{m, x, X\}; \\
\]
\]

\[
\text{RespCR}(B) = [ \\
\quad \text{receive } Y, B, \{y, Y\}; \\
\quad \text{new n;} \\
\quad \text{send } B, Y, \{n, \text{sig}_B\{y, n, Y\}\}; \\
\quad \text{receive } Y, B, \text{sig}_Y\{y, n, B\}; \\
\]
\]

\[
\text{CR} \vdash [ \text{InitCR}(A, B) ]_A \\
\quad \diamond \text{Verify}(A, \text{sig}_B\{m, n, A\})
\]

If A completed a protocol session, it must have verified B’s signature at some point.
2: Properties of Signatures

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};
]

\[ CR \models [\text{InitCR}(A, B)]_A \text{ Honest}(B) \supset \exists \ t'(\Diamond \text{Send}(B, t') \land \text{Contains}(t', \text{sig}_B\{m, n, A\})) \]

If A completed protocol and B is honest, then B must have sent its signature as part of some message.
Honesty Invariant

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 |- Honest(X) ∧
   ◊ Send(X, t') ∧ Contains(t', sig_X{y, x, Y}) ∧
   ¬ ◊ New(X, y) ⊃
   ◊ Receive(X, {Y, X, {y, Y}})

Honest responder only sends his signature if he received a properly formed first message of the protocol.

This condition disambiguates sig_X(...) sent by responder from sig_A(...) sent by initiator.
Reminder: Honesty Rule

∀ roles R of Q. ∀ initial segments A ⊆ R.

\[ Q \mid - [ A ]_X \phi \]

\[ Q \mid - \text{Honest}(X) \supset \phi \]
3: Use Honesty Rule

\[
\text{InitCR}(A, X) = [ \\
\quad \text{new m;} \\
\quad \text{send } A, X, \{m, A\}; \\
\quad \text{receive } X, A, \{x, \text{sig}_X\{m, x, A\}\}; \\
\quad \text{send } A, X, \text{sig}_A\{m, x, X\}; \\
\]
\]

\[
\text{RespCR}(B) = [ \\
\quad \text{receive } Y, B, \{y, Y\}; \\
\quad \text{new n;} \\
\quad \text{send } B, Y, \{n, \text{sig}_B\{y, n, Y\}\}; \\
\quad \text{receive } Y, B, \text{sig}_Y\{y, n, B\}; \\
\]
\]

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

If A completed protocol and B is honest, then B must have received A’s first message.
4: Nonces Imply Temporal Order

\[\text{InitCR}(A, X) = [\]
\[\text{new } m;\]
\[\text{send } A, X, \{m, A\};\]
\[\text{receive } X, A, \{x, \text{sig}_X\{m, x, A\}\};\]
\[\text{send } A, X, \text{sig}_A\{m, x, X\};\]
\]

\[\text{RespCR}(B) = [\]
\[\text{receive } Y, B, \{y, Y\};\]
\[\text{new } n;\]
\[\text{send } B, Y, \{n, \text{sig}_B\{y, n, Y\}\};\]
\[\text{receive } Y, B, \text{sig}_Y\{y, n, B\};\]
\]

\[\text{CR} |- [ \text{InitCR}(A, B) ]_A \text{ Honest}(B) \supset\]
\[\text{ActionsInOrder}(\ldots)\]
Complete Proof

Table 8. Deductions of A executing Init role of CR
Properties of Proof System

◆ Soundness
  • If $\phi$ is a theorem, then $\phi$ is a valid formula
    - $Q |- \phi$ implies $Q |= \phi$
  • Informally: if we can prove something in the logic, then it is actually true

◆ Proved formula holds in any step of any run
  • There is no bound on the number of sessions!
  • Unlike finite-state checking, the proved property is true for the entire protocol, not for specific session(s)
Weak Challenge-Response

InitWCR(A, X) = [
  new m;
  send A, X, {m};
  receive X, A, {x, sigX{m, x}};
  send A, X, sigA{m, x};
]

RespWCR(B) = [
  receive Y, B, {y};
  new n;
  send B, Y, {n, sigB{y, n}};
  receive Y, B, sigY{y, n};
]
1: A Reasons about Own Actions

\[
\text{InitWCR}(A, X) = [ \\
\quad \text{new } m; \\
\quad \text{send } A, X, \{m\}; \\
\quad \text{receive } X, A, \{x, \text{sig}_X\{m, x\}\}; \\
\quad \text{send } A, X, \text{sig}_A\{m, x\}; \\
\]
\]

\[
\text{RespWCR}(B) = [ \\
\quad \text{receive } Y, B, \{y\}; \\
\quad \text{new } n; \\
\quad \text{send } B, Y, \{n, \text{sig}_B\{y, n\}\}; \\
\quad \text{receive } Y, B, \text{sig}_Y\{y, n\}; \\
\]
\]

\[
\text{WCR} \mid - [ \text{InitWCR}(A, B) ]_A \\
\quad \Diamond \text{Verify}(A, \text{sig}_B\{m, n\})
\]
2: Properties of Signatures

InitWCR(A, X) = [ 
    new m;
    send A, X, {m};
    receive X, A, {x, sig_x{m, x}};
    send A, X, sig_A{m, x};;
]

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

WCR |- [ InitWCR(A, B) ]_A Honest(B) ⊃
    ∃ t' (◊ Send(B, t') ∧
    Contains(t', sig_B{m, n}))
Honesty Invariant

\[ \text{InitWCR}(A, X) = [ \]
\[ \text{new m;} \]
\[ \text{send } A, X, \{m\}; \]
\[ \text{receive } X, A, \{x, \text{sig}_X\{m, x\}\}; \]
\[ \text{send } A, X, \text{sig}_A\{m, x\}; \]
\[ ] \]
\[ \text{RespWCR}(B) = [ \]
\[ \text{receive } Y, B, \{y\}; \]
\[ \text{new n;} \]
\[ \text{send } B, Y, \{n, \text{sig}_B\{y, n\}\}; \]
\[ \text{receive } Y, B, \text{sig}_Y\{y, n\}; \]
\[ ] \]
\[ \text{WCR} |- \text{Honest}(X) \land \]
\[ \text{◇Send}(X, t') \land \text{Contains}(t', \text{sig}_X\{y, x\}) \land \]
\[ \neg \text{◇New}(X, y) \supset \]
\[ \text{◇Receive}(X, \{Y, X, \{y\}\}) \]

In this protocol, \text{sig}_X\{y, x\} does not explicitly include identity of intended recipient Y
3: Use Honesty Rule

\[ \text{InitWCR}(A, X) = [ \]
\[
\begin{align*}
    &\text{new } m; \\
    &\text{send } A, X, \{m\}; \\
    &\text{receive } X, A, \{x, \text{sig}_X\{m, x\}\}; \\
    &\text{send } A, X, \text{sig}_A\{m, x\}; \\
\end{align*}
\]

\[ ] \]

\[ \text{RespWCR}(B) = [ \]
\[
\begin{align*}
    &\text{receive } Y, B, \{y\}; \\
    &\text{new } n; \\
    &\text{send } B, Y, \{n, \text{sig}_B\{y, n\}\}; \\
    &\text{receive } Y, B, \text{sig}_Y\{y, n\}; \\
\end{align*}
\]

\[ ] \]

\[ \text{WCR} |- [ \text{InitWCR}(A, B) ]_A \ \text{Honest}(B) \supset \]
\[ \Diamond \text{Receive}(B, \{Y, B, \text{sig}_Y\{y, n\}\}) \]

B receives 3\textsuperscript{rd} message from someone, not necessarily A
Failed Proof and Counterexample

◆ WCR does not provide the strong authentication property for the initiator

◆ Counterexample: intruder can forge sender’s and receiver’s identity in first two messages

• A  ->  X(B)  A, B, m
• X(C) ->  B  C, B, m  [X pretends to be C]
• B  ->  X(C)  n, sig_B(m, n)
• X(B) ->  A  n, sig_B(m, n)
Further Work on Protocol Logic

◆ See papers by Datta, Derek, Mitchell, and Pavlovic on the course website
  • With a Diffie-Hellman primitive, prove authentication and secrecy for key exchange (STS, ISO-97898-3)
  • With symmetric encryption and hashing, prove authentication for ISO-9798-2, SKID3

◆ Work on protocol derivation
  • Build protocols by combining standard parts
    - Similar to the derivation of JFK described in class
  • Reuse proofs of correctness for building blocks
    - Compositionality pays off!