Separating Search from Algebra in CPSA

Joshua D. Guttman
Jonathan K. Millen
John D. Ramsdell
Ariel Segall

Supported by the National Security Agency

http://www.ccs.neu.edu/home/guttman
CPSA Algorithm

\[ \mathcal{F} := \{A_0\}; \quad \text{shapes} := \emptyset; \quad \text{seen} := \mathcal{F}; \]

while \( \mathcal{F} \neq \emptyset \) begin

\[ A := \text{select}(\mathcal{F}); \quad \mathcal{F} := \mathcal{F} \setminus \{A\}; \]

if redundant_strand(\(A\)) then continue
else if (n:=unsolved_node(\(A\)))

then begin

let new = targets(get_cohort(n, A)) \ seen in \( \mathcal{F} := \mathcal{F} \cup \text{new}; \)
\( \mathcal{F} := \mathcal{F} \setminus (\text{filter dead } \mathcal{F}); \)
\( \text{seen} := \text{seen} \cup \text{new} \)

end
else /* \(A\) realized */

\( \text{shapes} := \text{shapes} \cup \text{min_real}_{A_0}(A) \)

end;

return \( \text{shapes} \)
Yahalom Protocol, 1

\[
\begin{align*}
\text{Init} & \quad \rightarrow \\
A, \ N_a & \\
\downarrow & \\
\{B, \ K, \ N_a, \ N_b\}_{\text{ltk}(A)} & \\
\downarrow & \\
\bullet & \\
\downarrow & \\
\{N_b\}_K & \\
\end{align*}
\]

\[
\begin{align*}
\text{Resp} & \\
A, \ N_a & \\
\downarrow & \\
\{A, \ N_a, \ N_b\}_{\text{ltk}(B)} & \\
\downarrow & \\
\bullet & \\
\downarrow & \\
\{A, \ K\}_{\text{ltk}(B)} & \\
\downarrow & \\
\bullet & \\
\downarrow & \\
\{N_b\}_K & \\
\end{align*}
\]
Yahalom Protocol, 2

\[ \{ B, K, N_a, N_b \}_{\text{ltk}(A)} \]

\[ \{ A, N_a, N_b \}_{\text{ltk}(B)} \]

\[ \{ A, K \}_{\text{ltk}(B)} \]
Partial Information about Executions

\[ a, n_a \rightarrow \text{Resp } b \]

\[ b, \{a, n_a, n_b\}_{\text{ltk}(b)} \rightarrow \cdot \]

\[ \{a, k\}_{\text{ltk}(b)} \rightarrow \cdot \]

\[ \{n_b\}_k \rightarrow \cdot \]
Partial Information about Executions

\[ a, n_a \rightarrow \text{Resp } b \]
\[ b, \{a, n_a, n_b\}_{ltk(b)} \]
\[ \{a, k\}_{ltk(b)} \]
\[ \{n_b\}_k \]

- \( ltk(a), ltk(b) \) uncompromised
- \( n_b \) freshly chosen
Partial Information about Executions

- $\operatorname{ltk}(a)$, $\operatorname{ltk}(b)$ uncompromised  \hspace{1cm} keys used only
- $n_b$ freshly chosen  \hspace{1cm} according to protocol
$n_b$ Transformed \hspace{1cm} \text{ltk}(a), \text{ltk}(b) \in \text{non}, \hspace{0.5cm} n_b \in \text{unique}$

$n_b$ occurs only within $\{|a, \ n_a, \ n_b\}_{\text{ltk}(b)}$ above line

\begin{center}
\begin{tikzpicture}
\node (a) at (0,0) {$a, \ n_a$}; \node (b) at (1.5,-1) {$b, \ \{|a, \ n_a, \ n_b\}_{\text{ltk}(b)}$}; \node (c) at (3.5,-1) {$\{|a, \ k\}_{\text{ltk}(b)}$}; \node (d) at (6,-1) {$\{|n_b\}_k$}; \node (resp) at (3,0) {Resp $b$};
\draw (a) -- (b); \draw (b) -- (c); \draw (c) -- (d); \draw (a) -- (resp); \draw (b) -- (resp); \draw (c) -- (resp); \draw (d) -- (resp);
\end{tikzpicture}
\end{center}
Either Regular Transformer... \( \text{ltk}(a), \text{ltk}(b) \in \text{non}, \ k', n_b \in \text{unique} \)

\( n_b \) occurs only within \( \{A, \ N_a, \ N_b\}_{\text{ltk}(B)} \) above line
... Or $\text{ltk}(b)$ is Compromised

- $\text{ltk}(b)$ documents assumed compromise

\[ \begin{align*}
\text{ltk}(b) & \leftrightarrow \text{Resp } b \\
\{a, n_a\} & \rightarrow \text{ltk}(b) \\
\{a, n_a, n_b\} & \rightarrow \text{ltk}(b) \\
\{a, k\} & \rightarrow \text{ltk}(b) \\
\{n_b\} & \rightarrow \text{ltk}(b) \\
\end{align*} \]

$ltk(a), ltk(b) \in \text{non}, \quad n_b \in \text{unique}$
... Or $\text{ltk}(b)$ is Compromised

- $\text{ltk}(b)$ documents assumed compromise

Impossible, because $\text{ltk}(b)$ assumed uncompromised
So: Only Regular Transformer

$ltk(a), ltk(b) \in \text{non}, \quad k', n_b \in \text{unique}$

$n_b$ occurs only within $\{a, n_a, n_b\}_{ltk(b)}$ above line
So: Only Regular Transformer

$ltk(a), ltk(b) \in \text{non}, \ k', n_b \in \text{unique}$

$n_b$ occurs only within $\{a, n_a, n_b\}_{ltk(b)}$ above line

---

How did we find $Serv$ with parameters $a, b, n_a, n_b, k'$?
So: Only Regular Transformer \( \text{ltk}(a), \text{ltk}(b) \in \text{non}, \ k', n_b \in \text{unique} \)

\( n_b \) occurs only within \( \{a, n_a, n_b\}_{\text{ltk}(b)} \) above line

How did we find \( \text{Serv} \) with parameters \( a, b, n_a, n_b, k' \)?

By unification
Unify Against Roles, 1

Role must receive msg unifying with \{a, n_a, n_b\}_{ltk(b)}
then emit \(n_b\) transformed
Role must receive msg unifying with \{a, n_a, n_b\}_{ltk(b)}
then emit $n_b$ transformed
Unify Against Roles, 2

Role must **receive** msg unifying with $\{a, n_a, n_b\}_{\text{ltk}(b)}$
then **emit** $n_b$ transformed
Succeeds with $[A \mapsto a, B \mapsto b, N_a \mapsto n_a, N_b \mapsto n_b]$
Unify Against Roles, 2

Role must receive msg unifying with \[\{a, n_a, n_b\}\text{ltk}(b)\]
then emit \(n_b\) transformed
Succeeds with \([A \mapsto a, B \mapsto b, N_a \mapsto n_a, N_b \mapsto n_b]\)

Note that \(K\) is unconstrained
$k'$: New Variable  \[\text{ltk}(a), \text{ltk}(b) \in \text{non}, \quad k', n_b \in \text{unique}\]

$n_b$ occurs only within \(\{a, n_a, n_b\}\) above line

\[\begin{align*}
\{b, k', n_a, n_b\}_{\text{ltk}(a)} & \quad \text{Serv} \quad \{a, n_a, n_b\}_{\text{ltk}(b)} \\
\{a, k\}_{\text{ltk}(b)} & \quad \{n_b\}_k
\end{align*}\]
\( n_b \) Transformed Again

\[ \text{ltk}(a), \text{ltk}(b) \in \text{non}, \quad k', n_b \in \text{unique} \]

\( n_b \) occurs only within \( \{a, n_a, n_b\}_{\text{ltk}(b)}, \{b, k', n_a, n_b\}_{\text{ltk}(a)} \)

above line

Either \( \text{ltk}(a) \) is compromised, or must unify again
Branch: $\text{ltk}(a)$ Compromised

$l\text{tk}(a), l\text{tk}(b) \in \text{non}, \quad k', n_b \in \text{unique}$

$n_b$ occurs only within $\{a, n_a, n_b\}_{l\text{tk}(b)}, \{b, k', n_a, n_b\}_{l\text{tk}(a)}$

above line

This branch is also a dead end, since $l\text{tk}(a)$ assumed uncompromised
Unify Against Roles, 2

Role must receive msg unifying with

\[
\{ a, n_a, n_b \}_{ltk(b)}, \{ b, k', n_a, n_b \}_{ltk(a)}
\]

then emit \( n_b \) outside those messages
Unify Against Roles, 2

Role must receive msg unifying with

\[ S = \{ \{a, n_a, n_b\}_{ltk(b)}, \{b, k', n_a, n_b\}_{ltk(a)} : k' \in \text{keys} \} \]

then emit \( n_b \) outside \( S \)
Unify Against Roles, 2

Role must receive msg unifying with

\[ S = \{ \{ a, n_a, n_b \}_{\text{ltk}(b)}, \{ b, k', n_a, n_b \}_{\text{ltk}(a)} : k' \in \text{keys} \} \]

then emit \( n_b \) outside \( S \)

So Serv does not qualify
Unify Against Roles, 1

Role must receive msg unifying with

\[ S = \{ \{ a, n_a, n_b \}_ltk(b), \{ b, k', n_a, n_b \}_ltk(a) : k' \in \text{keys} \} \]

then emit \( n_b \) outside \( S \)
Role must receive msg unifying with

\[ S = \{ \{a, n_a, n_b\}_{\text{ltk}(b)}, \{b, k', n_a, n_b\}_{\text{ltk}(a)} : k' \in \text{keys} \} \]

then emit \( n_b \) outside \( S \)

So Init is only candidate
Only Yahalom Shape for Responder

\[ a : s_i \quad \text{\longrightarrow} \quad b : s_r \]

\[ s : s_s \quad \text{\longleftarrow} \quad n_0 \]

\[ s : s_s \quad \text{\longleftarrow} \quad n_0 \]

\[ m_1 \quad \text{\longleftarrow} \quad \bullet \]

\[ m_2 \quad \text{\longleftarrow} \quad \bullet \]

\[ n_0 \quad \text{\longleftarrow} \quad n_1 \]

\[ n_0 \quad \text{\longleftarrow} \quad n_2 \]

\[ a, b, n_a, n_b, k \quad \text{\longleftarrow} \quad a, b, n_a, n_b, k \quad \text{\longleftarrow} \quad a, b, n_a, n_b, k \]

Authenticates \( a, s \) to \( b \)
Reading Off Secrecy

\[ \text{ltk}(a), \text{ltk}(b) \in \text{non}, \quad k', n_b \in \text{unique} \]

---

Dead end: validates secrecy of \( k \)
Augmentation Theorem

If a state requires transformation, one of the following applies:

- A transforming instance of some role must be added
  e.g. unifying instances of Serv, Init ;
Augmentation Theorem

If a state requires transformation, one of the following applies:

- A transforming instance of some role must be added
  e.g. unifying instances of Serv, Init;
- A compromised key must be recorded
  e.g. compromised keys ltk(b) ← , ltk(a) ←; or
Augmentation Theorem

If a state requires transformation, one of the following applies:

- A transforming instance of some role must be added
e.g. unifying instances of Serv, Init;

- A compromised key must be recorded
e.g. compromised keys $\text{ltk}(b), \text{ltk}(a)$; or

- Variables must be fused
e.g. session keys $k, k'$
Augmentation Theorem

If a state requires transformation, one of the following applies:

- A transforming instance of some role must be added
  e.g. unifying instances of Serv, Init;
- A compromised key must be recorded
  e.g. compromised keys $\leftrightarrow$, $\leftrightarrow$; or
- Variables must be fused
  e.g. session keys $k, k'$

Set of possible solutions
for one required transformation: cohort
Augmentation Theorem

If a state requires transformation, one of the following applies:

■ A transforming instance of some role must be added
e.g. unifying instances of Serv, Init ;

■ A compromised key must be recorded
e.g. compromised keys \( \text{ltk}(b) \leftarrow, \text{ltk}(a) \leftarrow \); or

■ Variables must be fused
e.g. session keys \( k, k' \)

Set of possible solutions
for one required transformation: cohort

If state does not require transformation, it’s realized
CPSA Algorithm

\[ \mathcal{F} := \{A_0\}; \quad \text{shapes} := \emptyset; \quad \text{seen} := \mathcal{F}; \]
while \( \mathcal{F} \neq \emptyset \) begin
\[ A := \text{select}(\mathcal{F}); \quad \mathcal{F} := \mathcal{F} \setminus \{A\}; \]
if redundant_strand(\( A \)) then continue
else if \((n:=\text{unsolved_node}(A))\)
then begin
let new = targets(get_cohort\((n, A)\)) \( \setminus \text{seen} \) in
\[ \mathcal{F} := \mathcal{F} \cup \text{new}; \]
\[ \mathcal{F} := \mathcal{F} \setminus (\text{filter dead } \mathcal{F}); \]
\text{seen} := \text{seen} \cup \text{new}
end
else /* \( A \) realized */
\[ \text{shapes} := \text{shapes} \cup \text{min_real}_{A_0}(A); \]
end;
return \text{shapes}
Unsolved Outgoing Node

Negative node $n$ is an **unsolved outgoing** node for $a, S$ in $\mathbb{A}$ if:

$$a \in \text{unique}_\mathbb{A} \text{ occurs outside } S \text{ in } \text{msg}(n),$$

but for all $m \prec_\mathbb{A} n$, $a$ occurs only within $S$ in $\text{msg}(m)$,

and $\text{used}(S)^{-1} \subseteq \text{non}_\mathbb{A} \cup \text{unique}_\mathbb{A}$

where

$$S \text{ is a set of encryptions } \{|t|\}_K$$

$$\text{used}(S)^{-1} = \{K^{-1} : \{|t|\}_K \in S, \text{ for some } t\}$$
Solving an Outgoing Node

If $n$ unsolved, outgoing for $a, S$ in $A$, the solutions for $n, a, S$ are:

1. Regular strands that receive $a$ only within $S$, then transmit $a$ outside $S$;
2. Listener strands that “hear” some key $K \in \text{used}(S)^{-1} \cap \text{unique}_A$
3. Contractions $\alpha$ such that $t \cdot \alpha$ no longer occurs outside $S \cdot \alpha$ in $\text{msg}(n) \cdot \alpha$
Unsolved Incoming Node

Negative node $n$ is an unsolved incoming node for $t, S$ in $A$ if:

- $t$ occurs outside $S$ in $\text{msg}(n)$,
- but for all $m \prec_A n$, $t$ occurs only within $S$ in $\text{msg}(m)$,
- and $K_0, \text{used}(S)^{-1} \subseteq \text{non}_A \cup \text{unique}_A$

where $t = \{t_0\}_{K_0}$
Solving an Incoming Node

If $n$ unsolved, incoming for $t, S$ in $\mathbb{A}$, the solutions for $n, t, S$ are:

1. Regular strands that receive $t$ only within $S$ (if at all), then transmit $t$ outside $S$;

2. Listener strands that “hear” some key $K \in \text{used}(S)^{-1} \cap \text{unique}_\mathbb{A}$ or else hear $K_0$

3. Contractions $\alpha$ such that $t \cdot \alpha$ no longer occurs outside $S \cdot \alpha$ in $\text{msg}(n) \cdot \alpha$

where $t = \{t_0\}_{K_0}$
Get Cohort

1. Unify with roles, finding all transforming edges receiving $t$ only within $S$, transmitting $t$ outside $S$

2. Add listener strands for all $K \in \text{used}(S)^{-1} \cap \text{unique}_{\Delta}, K_0$ if incoming

3. Apply contractions that destroy transformation

These embeddings, contractions form a cohort
Dependency on Algebra

- Algebra provides notion of unification
- Algebra provides set of contractions (parameter-reducing replacements)
Dependency on Algebra

- Algebra provides notion of unification
- Algebra provides set of contractions (parameter-reducing replacements)
- Algebra plus adversary model justifies test theorems
  - Every “penetrator web” based on msgs with $t$ only within $S$
    that produces msg with $t$ outside $S$
    consumes some key $K \in \text{used}(S)^{-1} \cap \text{unique}_A$ or $K_0$
Protocol Analysis

- Cryptographic protocols: (e.g. TLS/SSL, SSH, ...)
  - Authenticated agreement on data values
  - Secrecy for some shared values
  - Freshness (loose synchronization)

short sequences of messages using crypto for the goals:

despite a powerful adversary
Protocol Analysis

- Cryptographic protocols: (e.g. TLS/SSL, SSH, ...) short sequences of messages using crypto for the goals:
  - Authenticated agreement on data values
  - Secrecy for some shared values
  - Freshness (loose synchronization) despite a powerful adversary

- Central mechanism for coordination in distributed systems
Protocol Analysis

- Cryptographic protocols: (e.g. TLS/SSL, SSH, ...)
  short sequences of messages using crypto for the goals:
  - Authenticated agreement on data values
  - Secrecy for some shared values
  - Freshness (loose synchronization)
  despite a powerful adversary

- Central mechanism for coordination in distributed systems

- Protocol analysis tasks:
  - Given a protocol and a goal, check success
  - Given goals, design protocol
  - Match protocol structure with crypto primitives
Current Research State

- Immense body of research using e.g.
  - Logics and rewriting
  - State-based techniques, Hoare logics
  - Process algebras, type systems or model checking
  - Special purpose methods, e.g. **strand spaces**

Secrecy, authentication **undecidable** in general
Current Research State

- Immense body of research using e.g.
  - Logics and rewriting
  - State-based techniques, Hoare logics
  - Process algebras, type systems or model checking
  - Special purpose methods, e.g. strand spaces

Secrecy, authentication undecidable in general

- Tools
  - Given protocol and goal, ensure satisfied
    - Bounded number of sessions or safe approx.
  - Given protocol and goal, find counterexample ("attack")
    - Bounded number of sessions
  - Not much protocol design support
Current Research State

- Immense body of research using e.g.
  - Logics and rewriting
  - State-based techniques, Hoare logics
  - Process algebras, type systems or model checking
  - Special purpose methods, e.g. strand spaces
  Secrecy, authentication undecidable in general

- Tools
  - Given protocol and goal, ensure satisfied
    - Bounded number of sessions or safe approx.
  - Given protocol and goal, find counterexample ("attack")
    - Bounded number of sessions
  - Not much protocol design support

- Intense focus: structure vs. crypto primitives (FCC’07)
Our Contribution

- Cryptographic Protocol Shape Analyzer (CPSA)
  - Given a protocol, tells you what can happen
  - You can read off authentication, secrecy goals
    - Goals achieved
    - Counterexamples/attacks
Our Contribution

- Cryptographic Protocol Shape Analyzer (CPSA)
  - Given a protocol, tells you **what can happen**
  - You can read off **authentication, secrecy** goals
    - Goals achieved
    - Counterexamples/attacks

- But: infinitely many things can happen
Our Contribution

- Cryptographic Protocol Shape Analyzer (CPSA)
  - Given a protocol, tells you what can happen
  - You can read off authentication, secrecy goals
    - Goals achieved
    - Counterexamples/attacks

- But: infinitely many things can happen

- CPSA enumerates the shapes of a protocol: the
  - essentially different
  - minimal
  executions possible for it

- Many protocols have very few shapes
  - E.g. 1 or 2 per initial trust assumption
In Particular

- CPSA search algorithm enumerates all shapes
  - Never represents adversary behavior
  - Manipulates states of partial information about regular behavior

- Each step is information-preserving
  - Maintain or reduce set of executions described
  - Information-preserving maps called homomorphisms
In Particular

- CPSA search algorithm enumerates all shapes
  - Never represents adversary behavior
  - Manipulates states of partial information about regular behavior

- Each step is information-preserving
  - Maintain or reduce set of executions described
  - Information-preserving maps called homomorphisms

- Progress in search means splitting
  - “Every execution described by $A_0$ is described by either $A_1$ or $A_2$ . . . ”
  - Motivated by protocol defn.
  - Progress steps are information-increasing maps, i.e. special homomorphisms $A_0 \mapsto A_1$ and $A_0 \mapsto A_2$, . . .

- Shape notion also defined via homomorphisms
Constructive Penetrator Strands
Destructive Penetrator Strands
Initial Penetrator Strands

Atom

\[ a \]
Executions are Bundles

- Causally well founded directed graphs made of
  - Instances of the protocol roles
  - Adversary behaviors

- “Causally well founded” means:
  - Every reception node has in-degree 1
  - Each strand starts at beginning
  - Graph is finite and acyclic
CPSA Algorithm

\[ \mathcal{F} := \{ A_0 \}; \quad \text{shapes} := \emptyset; \quad \text{seen} := \mathcal{F}; \]

while \( \mathcal{F} \neq \emptyset \) begin
\[ A := \text{select}(\mathcal{F}); \quad \mathcal{F} := \mathcal{F} \setminus \{ A \}; \]

if \( \text{realized}(A) \)
\[ \text{then shapes} := \text{shapes} \cup \min_{A_0} \text{real}_A(A); \]
else if \( \text{redundant_strand}(A) \) then continue
else if \( \text{transform_needed}(A) \)
\[ \text{then begin} \]
\[ \text{let new} = \text{targets} \left( \text{get_cohort}(A) \right) \setminus \text{seen} \in \]
\[ \mathcal{F} := \mathcal{F} \cup \text{new}; \]
\[ \mathcal{F} := \mathcal{F} \setminus (\text{filter dead } \mathcal{F}); \]
\[ \text{seen} := \text{seen} \cup \text{new} \]
\[ \text{end} \]
else fail “Impossible.”
\[ \text{end}; \]

return shapes
Only Yahalom Shape for Responder

\[ b : s_r, \text{ non} = \{\text{ltk}(a), \text{ltk}(b)\}, \text{ unique} = \{n_b\} \]
Protocols and their Shapes

- The shapes of a protocol: the
  - essentially different
  - minimal
eexecutions possible for it
- Shapes contain only regular behavior, not adversary behavior
- Many protocols have very few shapes:
  - Frequently just one
  - Possibly infinitely many
  - From shapes, read off:
    - Authentication and secrecy properties
    - Anomalies (too many/unexpected peers, etc.)
- Implementation: Cryptographic Protocol Shape Analyzer
  
  http://www.ccs.neu.edu/home/guttman
A Skeleton

\[ \{ b, \; k', \; n_a, \; n_b \} \lt k(a) \rightarrow m_0 \]

\[ m_0 \rightarrow m_1 \]

\[ b, \{ a, \; n_a, \; n_b \} \lt k(b) \rightarrow n_0 \]

\[ a, \; n_a \rightarrow \text{Resp} \; b \]

\[ \{ a, \; k \} \lt k(b) \rightarrow \bullet \]

\[ \{ n_b \} \lt k \rightarrow n_1 \]

\[ n_0 \lt m_0, \; m_1 \lt n_1 \]

\[ \lt k(a), \; \lt k(b) \in \text{non} \]

\[ n_b, \; k' \in \text{unique} \]

(plus strand ordering)

(non-compromised)

(fresh)
**Skeleton** 

1. $\text{nodes}_A$, finite set of regular nodes
2. $\preceq_A$, reflexive partial order on $\text{nodes}_A$ representing causal accessibility
3. $\text{non}_A$, set of keys assumed non-originating (uncompromised, because used but not sent)
4. $\text{unique}_A$, set of atoms assumed uniquely originating (like nonces, session keys)

When $n \Rightarrow^* n'$ and $n' \in \text{nodes}_A$, we require

\[ n \in \text{nodes}_A \text{ and } n \preceq_A n' \]
Homomorphism $H : A_0 \leftrightarrow A_1$

$H = \phi, \alpha$ where

$\phi : \text{nodes}_{A_0} \leftrightarrow \text{nodes}_{A_1}$
$\alpha$ maps atoms to atoms

such that

1. $\phi$ respects strand structure, and

\[ \text{msg}(n) \cdot \alpha = \text{msg}(\phi(n)) \]

for all $n \in A_0$

2. $m \preceq_{A_0} n$ implies $\phi(m) \preceq_{A_1} \phi(n)$

3. $(\text{non}_{A_0}) \cdot \alpha \subset \text{non}_{A_1}$

4. $(\text{unique}_{A_0}) \cdot \alpha \subset \text{unique}_{A_1}$

---

$^a x \cdot \alpha$ means $\alpha$ applied to all atoms throughout $x$
Homomorphisms Preserve Information

- Homomorphisms are information-preserving maps
- A skeleton $A_0$ describes all the realized $A'$ such that $H : A_0 \leftrightarrow A'$
- Information preserved by $H_0 : A_0 \leftrightarrow A_1$: $A_1$ describes a subset
  
  If $H_1 : A_1 \leftrightarrow A'$ then $H_1 \circ H_0 : A_0 \leftrightarrow A'$

so $A_0$ describes $A'$ if its image $A_1$ does
Homomorphisms Preserve Information

- Homomorphisms are information-preserving maps
- A skeleton $\mathbb{A}_0$ describes all the realized $\mathbb{A}'$ such that $H : \mathbb{A}_0 \rightarrow \mathbb{A}'$
- Information preserved by $H_0 : \mathbb{A}_0 \leftrightarrow \mathbb{A}_1$:
  - $\mathbb{A}_1$ describes a subset
  - If $H_1 : \mathbb{A}_1 \leftrightarrow \mathbb{A}'$
    then $H_1 \circ H_0 : \mathbb{A}_0 \leftrightarrow \mathbb{A}'$
  - so $\mathbb{A}_0$ describes $\mathbb{A}'$ if its image $\mathbb{A}_1$ does
- Non-trivial homomorphisms increase information
  - Embedding $H_0$ adds info about additional events
  - $H_0$ identifying $B$, $C$ excludes cases where they differ
  - $H_0$ identifying nodes . . .
Shapes are Really Homomorphisms

- $H = (\phi, \alpha)$ is nodewise injective iff $\phi$ is injective
- $H \leq_n J$ means some nodewise injective $H'$, composed with $H$, yields $J$

$$J = H' \circ H$$

$\leq_n$ is a partial order to within isomorphism

Definition: $H : A_0 \rightarrowrightarrow A_1$ is a shape if

1. $A_1$ is realized
2. $H$ is $\leq_n$-minimal among homomorphisms to realized skeletons
NS Shape: Responder

At least this much must be present, assuming:

- Responder strand occurred
- \( \text{privk}(A) \) uncompromised
- \( N_b \) freshly chosen
NS Shape: Responder

At least this much must be present, assuming:

- Responder strand occurred
- \( \text{privk}(A) \) uncompromised
- \( N_b \) freshly chosen

Two realizations: \( \text{privk}(C) \) compromised, or \( C \leftrightarrow B \).
$C \leftrightarrow B$ yields NS intended run

\[\begin{array}{c}
A \xrightarrow{\{N_a, A\}_{\text{pubk}(B)}} B \\
\downarrow \{N_a, N_b\}_{\text{pubk}(A)} \leftrightsquigarrow \{N_a, N_b\}_{\text{pubk}(A)} \\
\downarrow \{N_b\}_{\text{pubk}(B)} \leftrightsquigarrow \{N_b\}_{\text{pubk}(B)}
\end{array}\]
$C \leftrightarrow B$ yields NS intended run

Less general than the shape on last slide, which didn’t force $C = B$
$C \mapsto B$ yields NS intended run

Less general than the shape on last slide, which didn’t force $C = B$

Shapes are minimal:

- Minimal set of nodes,
- which identify variables minimally
NS Unique Shape Allows Attack

- Responder strand occurred
- privk(A) uncompromised
- \( N_b \) freshly chosen
NS Unique Shape Allows Attack

- Responder strand occurred
- $\text{privk}(A)$ uncompromised
- $N_b$ freshly chosen

Two realizations: $\text{privk}(C)$ compromised, or $C \rightarrow B$. 
$C \leftrightarrow B$ yields NS intended run
\( C \leftrightarrow B \) yields NS intended run

Less general than the shape on last slide, which didn’t force \( C = B \)
$C \leftrightarrow B$ yields NS intended run

Less general than the shape on last slide, which didn’t force $C = B$

Shapes are minimal:

Minimal set of nodes, which identify variables minimally