Collaborative Systems

Andre Scedrov

University of Pennsylvania, USA

To Share or not to Share

Examples: Administrative tasks, protocols

- Agents collaborate to achieve some common goal.
- No intruder can enter the system.
- However, an agent does not completely trust any other agent.
- Therefore, while collaborating, an agent might not want some confidential information to be leaked.

Agenda

- **Local State Transition Systems**

- Fresh Values
- Progressing Collaborative Systems
- Bounded Memory Adversary
- Timed Collaborative Systems

Collaborative Systems [Kanovich, Rowe, and Scedrov]

Model (LSTS)

- FOL signature
- Configurations are multisets of facts:

$$\{\text{Nurse}(\text{Tom}, \text{id1}, \text{blood}), \text{Nurse}(\text{Sam}, \text{id2}, \text{blood})\}$$

- Actions are rewrite rules:

$$\text{Nurse}(\text{X}, \text{Y}, \text{blood}) \rightarrow \text{Nurse}(\text{blank}, \text{Y}, \text{blood})$$

$$\text{Lab}(\text{id}, \text{blood}) \rightarrow \text{Lab}(\text{id}, \text{testResults})$$

- Goals are multisets of facts:

$$\{\text{Doctor}(\text{testResults}, \text{Tom})\}$$

- Critical configurations are configurations that have to be avoided

$$\{\text{Lab}(\text{testResults}, \text{Tom})\} \qquad \{\text{Nurse}(\text{Tom}, \text{id1}, \text{blood}), \text{Nurse}(\text{Sam}, \text{id1}, \text{blood})\}$$

Previous results [Kanovich, Rowe, and Scedrov, CSF'07, CSF'09, Rowe PhD Dissertation UPENN'09]

Previous results [Kanovich, Rowe, and Scedrov, CSF'07, CSF'09, Rowe PhD Dissertation UPENN'09]

The planning problem

Plan compliance: Is there a plan from an initial configuration to a configuration containing a goal such that no critical configuration is reached along the plan?

Medical scenario: the test results of a patient should not be publicly leaked with the patient's name.

Previous results [Kanovich, Rowe, and Scedrov, CSF'07, CSF'09, Rowe PhD Dissertation UPENN'09]

The planning problem

Plan compliance: Is there a plan from an initial configuration to a configuration containing a goal such that no critical configuration is reached along the plan?

Medical scenario: the test results of a patient should not be publicly leaked with the patient's name.

Assumption

Balanced actions, that is, actions that have the same number of facts in their pre and post conditions.

Along a plan, states have the same number of facts (intuitively, agents have collectively a bounded memory): different from the Dolev-Yao intruder.

(Closed Room)

Previous results [Kanovich, Rowe, and Scedrov, CSF'07, CSF'09, Rowe PhD Dissertation UPENN'09]

The planning problem

Plan compliance: Is there a plan from an initial configuration to a configuration containing a goal such that no critical configuration is reached along the plan?

Medical scenario: the test results of a patient should not be publicly leaked with the patient's name.

Assumption

Balanced actions, that is, actions that have the same number of facts in their pre and post conditions.

Along a plan, states have the same number of facts (intuitively, agents have collectively a bounded memory): different from the Dolev-Yao intruder.

(Closed Room)

Complexity Results

Balanced actions:

PSPACE-complete

Not necessarily balanced actions:

Undecidable

Systems with balanced actions

Problem

• Although checking for the existence of plan is in PSPACE, it turns out that to write down the entire plan may require exponential space because the plan might be exponentially long.

Systems with balanced actions

Problem

• Although checking for the existence of plan is in PSPACE, it turns out that to write down the entire plan may require exponential space because the plan might be exponentially long.

• The solution given in CSF'07 was by scheduling a plan in PSPACE.

Systems with balanced actions

Problem

• Although checking for the existence of plan is in PSPACE, it turns out that to write down the entire plan may require exponential space because the plan might be exponentially long.

• The solution given in CSF'07 was by scheduling a plan in PSPACE.

Example: Towers of Hanoi

$$Clear(x)\ On(x,y)\ Clear(z)\ S(x,z) \rightarrow Clear(x)\ Clear(y)\ On(x,z)\ S(x,z)$$

Given *n* disks plans must be of exponential length $2^n - 1$, at least.

PSPACE Upper Bound

- Must check both goal reachability and policy compliance.
- Rely on a non-deterministic algorithm which can be *determinized* by Savitch's Theorem.
- Also use the fact that PSPACE = COPSPACE.

PSPACE Upper Bound

Some Assumptions

- All actions are balanced.

- There are three functions, C, G, and T, which run in polynomial space and:

  - $C(Z) = 1$ if Z is a critical configuration, and $C(Z) = 0$, otherwise;
  - $G(Z) = 1$ if Z is a goal configuration, and $G(Z) = 0$, otherwise;
  - $T(\alpha) = 1$ if α is a valid transition, and $T(\alpha) = 0$, otherwise.

- Let W be the initial configuration.

PSPACE Upper Bound

Algorithm

- Use non-determinism to “guess” a compliant plan leading to a goal configuration
- Each intermediate configuration Z_i must check if $C(Z_i) = 1$ and if $G(Z_i) = 1$;
- Recording one step at a time, this is done in polynomial space;
- Goal reachability and plan compliance are check simultaneously in this algorithm

PSPACE Upper Bound

Algorithm

- Initialize $Z_0 = W$ and $i = 0$;
- If $C(Z_i) = 1$, output no;
- If $G(Z_i) = 1$, output yes;
- Otherwise use non-determinism to “guess” a compliant plan leading to a goal configuration. By using T, guess an action α applicable to Z_i resulting in the configuration Z';
- Set $Z_{i+1} = Z'$ and $i := i+1$;
- Repeat.

Record one step at a time, this can be done in polynomial space.

Agenda

- Local State Transition Systems
- **Fresh Values**
- Progressing Collaborative Systems
- Bounded Memory Adversary
- Timed Collaborative Systems

New feature: fresh values [Kanovich, Ban Kirigin, Nigam, and Scedrov, FAST'10]

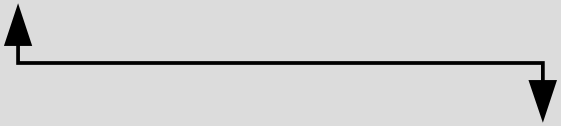
Motivation

New feature: fresh values [Kanovich, Ban Kirigin, Nigam, and Scedrov, FAST'10]

Motivation

Agents might need to create fresh values or *nonces*:

$$\text{nurse}(\text{Tom}, \text{blank}, \text{blood}) \rightarrow \exists\, \text{testNo}.\text{nurse}(\text{Tom}, \text{testNo}, \text{blood})$$



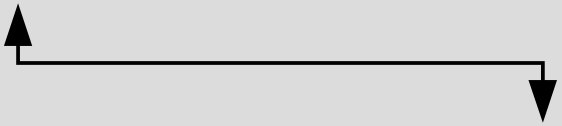
Each sample should have a different number assigned.

New feature: fresh values [Kanovich, Ban Kirigin, Nigam, and Scedrov, FAST'10]

Motivation

Agents might need to create fresh values or *nonces*:

$$\text{nurse}(\text{Tom}, \text{blank}, \text{blood}) \rightarrow \exists\, \text{testNo}.\text{nurse}(\text{Tom}, \text{testNo}, \text{blood})$$



Each sample should have a different number assigned.

Other examples:

- opening a new bank account;
- changing a customer's password;
- creating a transaction number or a case number.

Actions that create fresh values

$$\text{nurse}(\text{Tom}, \text{blank}, \text{blood}) \rightarrow \exists\ \text{testNo}.\text{nurse}(\text{Tom}, \text{testNo}, \text{blood})$$

The fresh value uses the memory slot used previously by the updated value.

Actions that create fresh values

$$\text{nurse}(\text{Tom}, \text{blank}, \text{blood}) \rightarrow \exists\, \text{testNo}.\text{nurse}(\text{Tom}, \text{testNo}, \text{blood})$$

The fresh value uses the memory slot used previously by the updated value.

Agents have a bounded memory even when they can create fresh values.

Actions that create fresh values

$$\text{nurse}(\text{Tom}, \text{blank}, \text{blood}) \rightarrow \exists\, \text{testNo}.\text{nurse}(\text{Tom}, \text{testNo}, \text{blood})$$

The fresh value uses the memory slot used previously by the updated value.

Agents have a bounded memory even when they can create fresh values.

$$\rightarrow \exists n.A(n)$$

For example, whenever such an unbalanced rule is used, it requires an extra memory slot to store the nonce created. That is, agents possess an unbounded memory.

Systems with balanced actions

Problem

• Although checking for the existence of plan is in PSPACE, it turns out that to write down the entire plan may require exponential space and exponentially many mutually distinct nonces.

Example: Towers of Hanoi, suitably modified to have balanced actions that always creates fresh values.

• To cope with this problem we use the fact that the number of constants in a configuration is bounded. In particular, we will show how to reuse obsolete constants instead of updating with fresh constants.

Systems with balanced actions

Theorem: Given a local state transition system (LSTS) with balanced actions that may create fresh values, any plan leading from an initial configuration *W* to a partial goal *Z* can be transformed into another plan also leading from *W* to *Z* that uses only a polynomial number of nonces with respect to the number of facts in the initial configuration and the upper bound on the size of facts.

Proof outline

Proof outline

- The number of facts, m, of any configuration in a plan does not change.

Proof outline

- The number of facts, *m*, of any configuration in a plan does not change.
- We assume that the size of facts, *k*, is bounded, where the size of facts is the number of symbols it contains.

$$|P(x,y)| = 3 \qquad |P(h(x,y),z)| = 5$$

Proof outline

- The number of facts, *m*, of any configuration in a plan does not change.
- We assume that the size of facts, *k*, is bounded, where the size of facts is the number of symbols it contains.

$$|P(x,y)| = 3 \qquad |P(h(x,y),z)| = 5$$

In any configuration there are at most *mk* occurrences of constants.

Proof outline

Alpha-equivalence among configurations
inspired by a similar notion from logic

$$\{A(t_1, n_1), B(n_2, n_1), C(n_3, t_2)\}$$

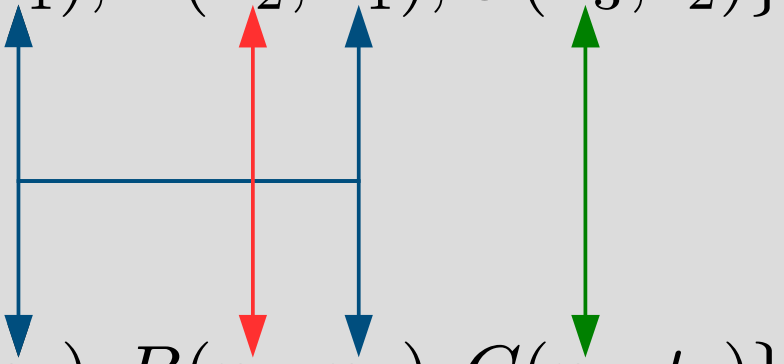
$$\{A(t_1, n_4), B(n_5, n_4), C(n_6, t_2)\}$$

Proof outline

Alpha-equivalence among configurations

inspired by a similar notion from logic

$$\{A(t_1, n_1), B(n_2, n_1), C(n_3, t_2)\}$$



$$\{A(t_1, n_4), B(n_5, n_4), C(n_6, t_2)\}$$

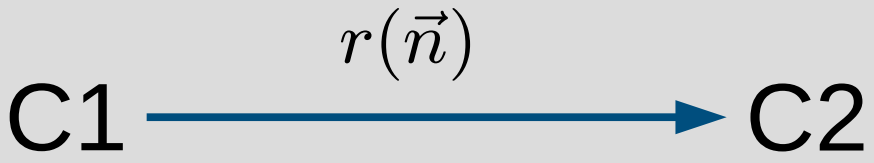
These configurations only differ in the names of the nonces used. Intuitively, they represent the same information.

Proof outline

Observational equivalence among plans

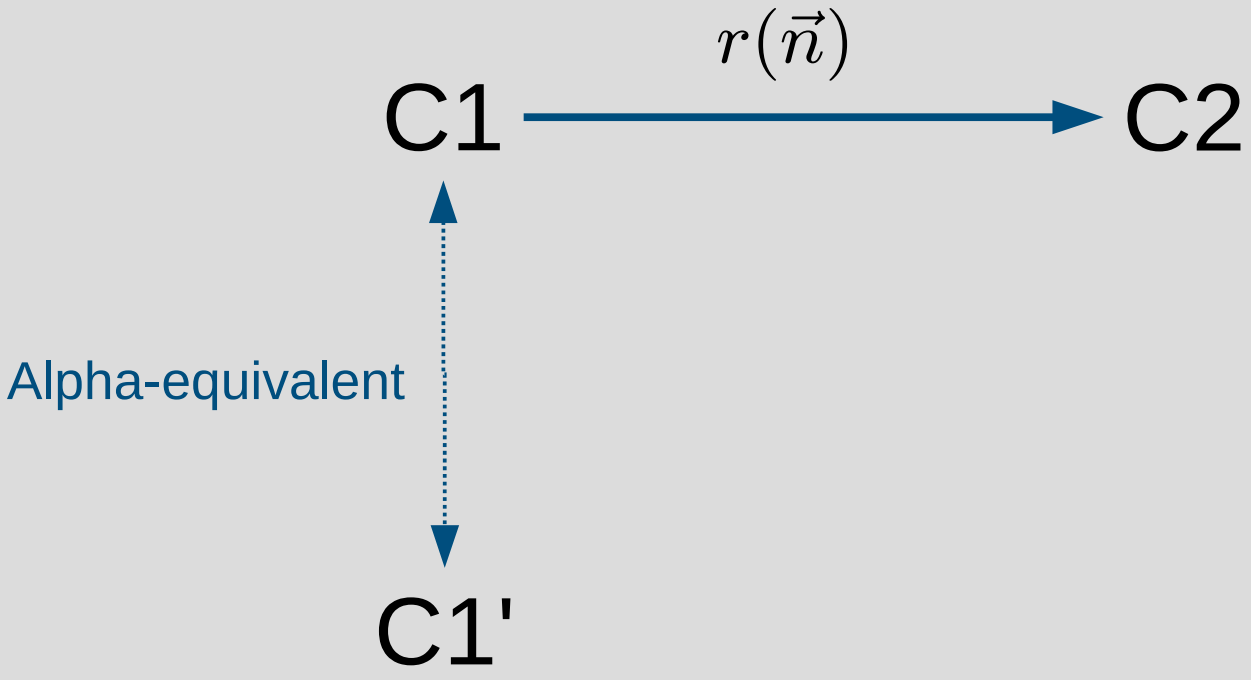
Proof outline

Observational equivalence among plans



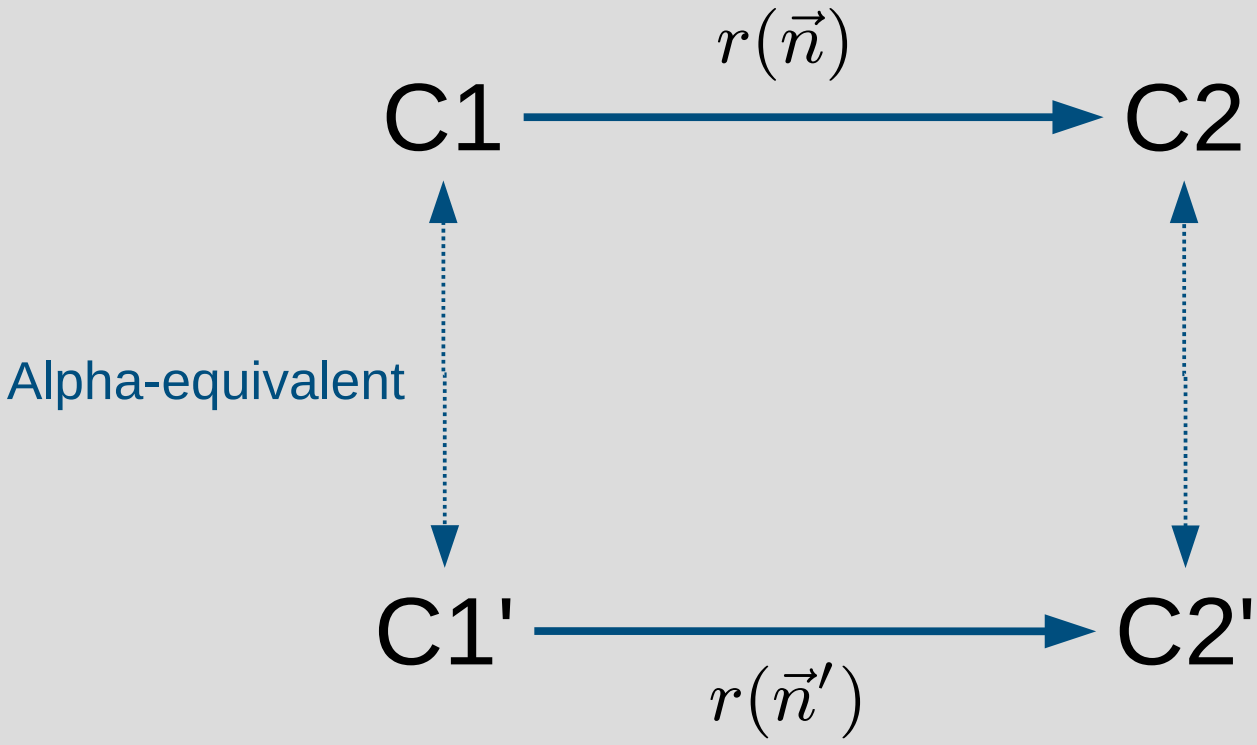
Proof outline

Observational equivalence among plans



Proof outline

Observational equivalence among plans



Where all nonces in C1' and C2' including the nonces $\vec{n}'$ are taken from a pre-defined set of ***2mk*** nonces.

Systems with balanced actions

Theorem: Given an LSTS system with balanced actions that can create fresh values, the plan compliance problem is PSPACE-complete.

Agenda

- Local State Transition Systems
- Fresh Values
- **Progressing Collaborative Systems**
- Bounded Memory Adversary
- Timed Collaborative Systems

Progressing Collaborative Systems [Kanovich, Ban Kirigin, Nigam, and Scedrov, FCS-PrivMod'10]

Progressing is inspired by the nature of security protocols, as well as many administrative and business processes: **once one step of a protocol session is taken, the same step is not repeated.**

Progressing Plans

A plan is progressing if an instance of an action appears at most once.

Note that this implies that the length of progressing traces are of polynomial due to the assumption of size of facts.

Progressing Collaborative Systems [Kanovich, Ban Kirigin, Nigam, and Scedrov, FCS-PrivMod'10]

This notion of progressing reflects the requirement that progressing processes are efficient, as one needs to consider only **traces of polynomial length** to check whether a process can be completed or not.

For instance, it is **not** possible to solve the Towers of Hanoi problem with a progressing plan.

Complexity [FCS-Privmod'10]

Assuming that one can check in poly-time whether a state is an initial or goal state, then the reachability problem for progressing plans is NP-complete when actions cannot create fresh values.

Progressing Collaborative Systems [Kanovich, Ban Kirigin, Nigam, and Scedrov, new]

However, extending this notion of progressing to systems that can create fresh values has turned out to be quite challenging.

One of the reasons is that with **the current definition of progressing,** progressing plans do not have necessarily polynomial length when one allows fresh values.

One can transform any problem into another problem whose solution is progressing, even problems that require exponential plans.

For instance, we can adapt the encoding of the Towers of Hanoi, so that each move creates a new nonce.

Progressing Collaborative Systems [Kanovich, Ban Kirigin, Nigam, and Scedrov, new]

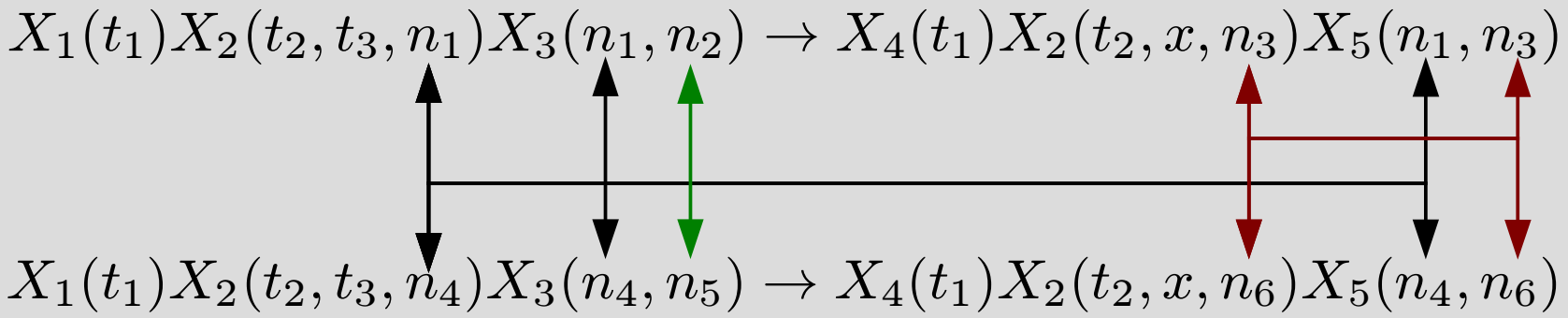
In order to extend the notion of progressing to the case where actions may create nonces, we shouldn't allow unbounded nonce generation. Instead we need to somehow limit the use of nonces, but how many nonces is enough?

For balanced systems, we know that it is enough to fix a polynomial number of nonce names with respect to the upper bound on the size of facts and the number of facts in the initial configuration.

Progressing Collaborative Systems [Kanovich, Ban Kirigin, Nigam, and Scedrov, new]

We extend the notion of **alpha-equivalence** to instances of actions.

Two instances, *r1* and *r2*, of the same action are equivalent if there is a bijection *σ* that maps the set of all nonce names appearing in one instance to the set of all nonce names appearing in the other instance, such that *(r1 σ) = r2*.



Progressing Collaborative Systems [Kanovich, Ban Kirigin, Nigam, and Scedrov, **new**]

We extend the definition of Progressing to plans containing nonces.

Given a balanced multiset rewrite system R whose actions may create fresh values, an initial configuration W and a polynomial *f(m,k)*, we say that a sequence of actions is progressing if it contains at most *f(m,k)* equivalent instances of any action, where m is the number of facts in the configuration W and k is the upper bound on size of facts.

With this new definition, it is not possible to solve the modified Towers of Hanoi problem using a progressing plan.

Progressing Collaborative Systems

Complexity [*new*]

Assuming that one can check in polynomial-time whether a state is an initial or goal state, then the reachability problem for progressing plans is NP-complete when actions are balanced and can create fresh values up to a polynomial number of times.

Summary of Results

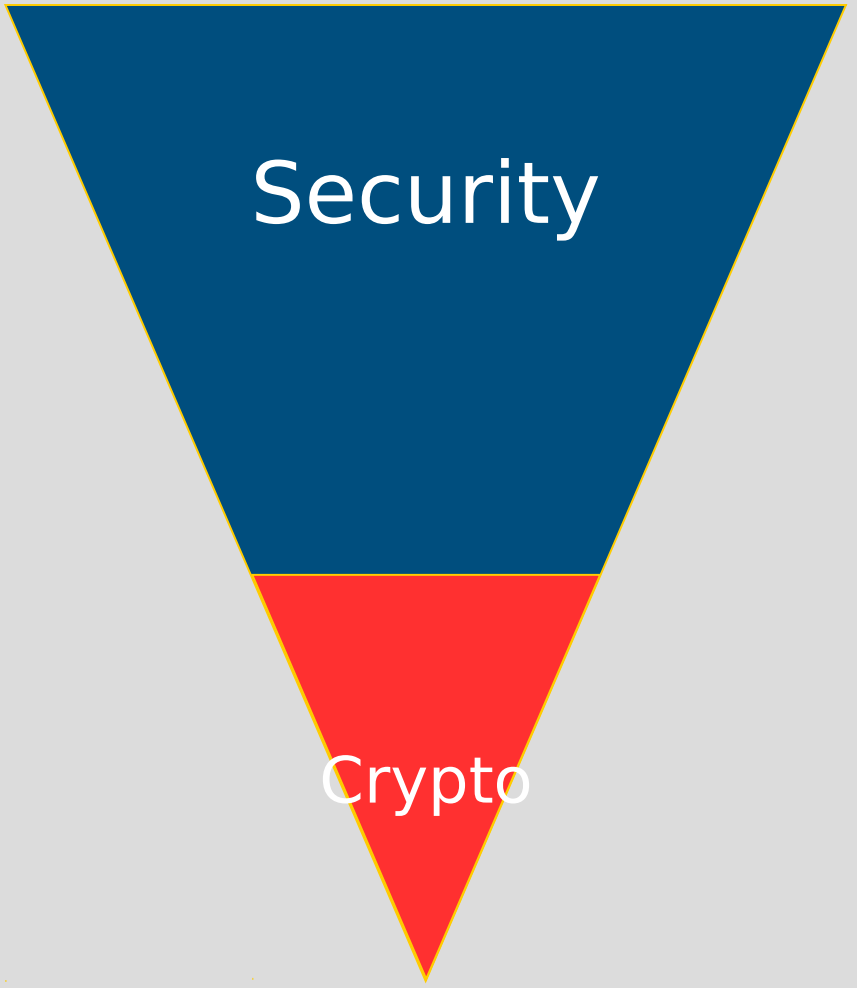
<table>
<tr><th colspan="4">Plan Compliance Problem</th></tr>
<tr><td rowspan="4">Balanced Actions</td><td rowspan="2">Nonces are not allowed</td><td>Progressing</td><td>NP-complete
[Kanovich et al. FCS-Privmod'10]</td></tr>
<tr><td>Not necessarily Progressing</td><td>PSPACE-complete
[Kanovich et al. CSF'07]</td></tr>
<tr><td rowspan="2">Nonces are allowed</td><td>Progressing</td><td>NP-Complete
[new]</td></tr>
<tr><td>Not Necessarily Progressing</td><td>PSPACE-complete
[Kanovich et al., FAST'10]</td></tr>
<tr><td colspan="3">Actions not necessarily balanced</td><td>Undecidable
[Kanovich et al., CSF'09]</td></tr>
</table>

Agenda

- Local State Transition Systems
- Fresh Values
- Progressing Collaborative Systems
- **Bounded Memory Adversary**
- Timed Collaborative Systems

Goal: protection of computer systems and digital information

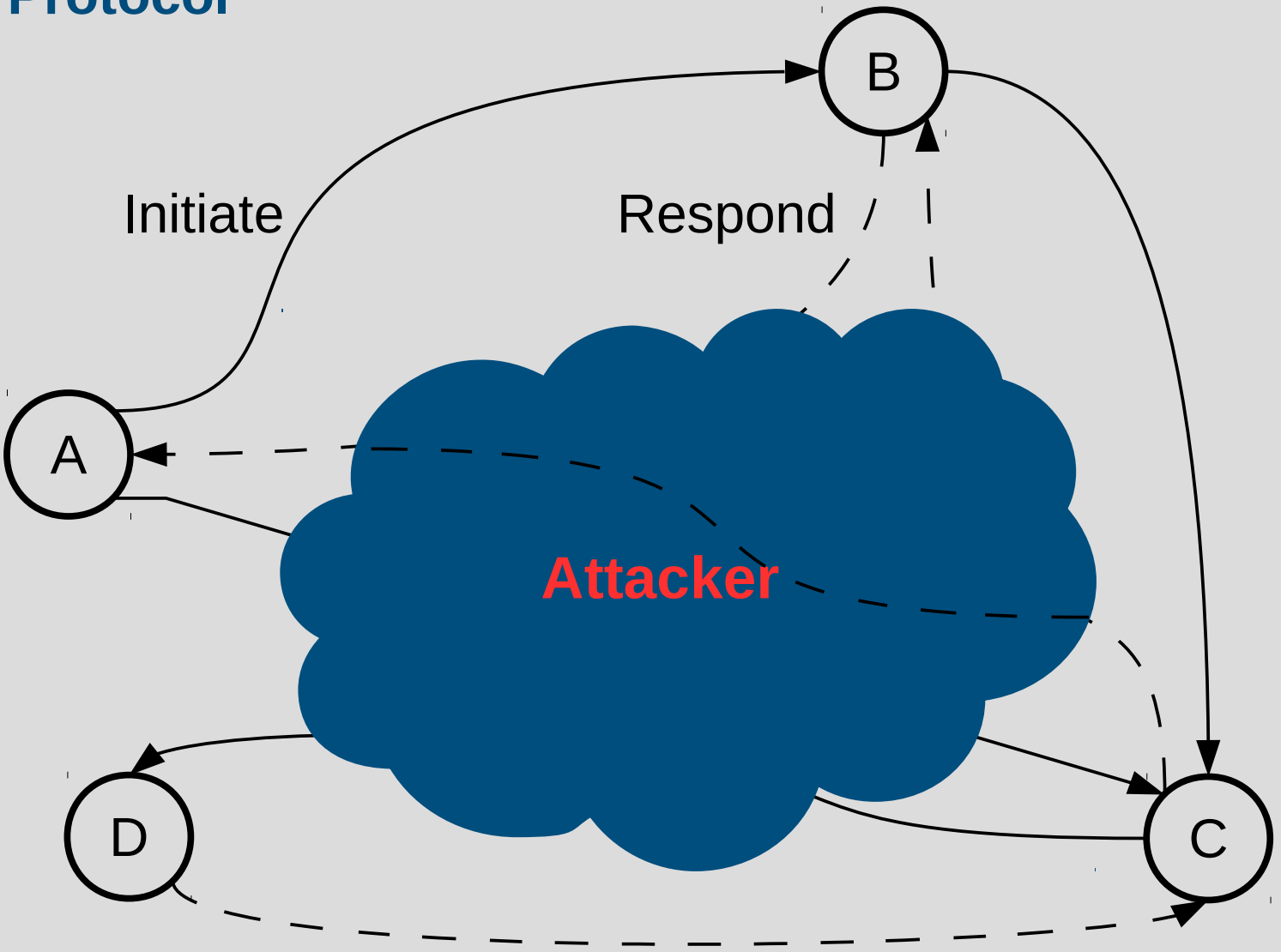
- Access control
- OS security
- Network security
- Cryptography
- …



Protocol Security

- Cryptographic Protocol
  - Program distributed over network
  - Use cryptography to achieve goal
- Attacker
  - Read, intercept, replace messages, and remember their contents
- Correctness
  - Attacker cannot learn protected secret or cause incorrect protocol completion

Run of Protocol

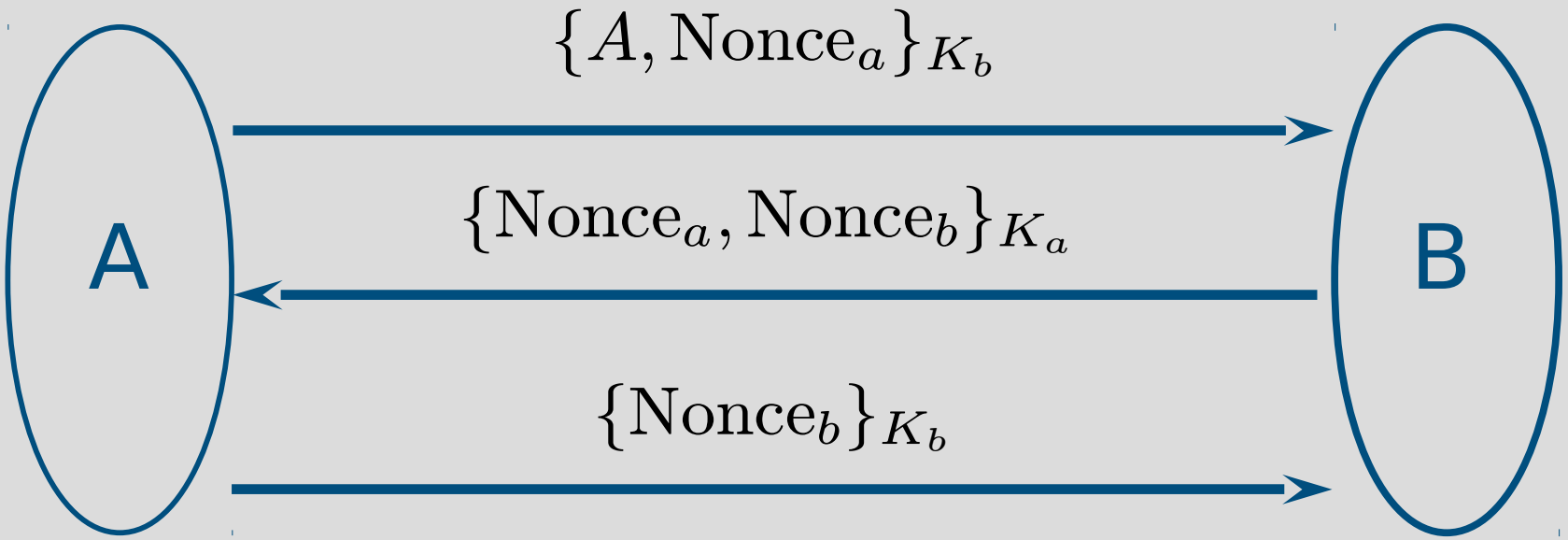


Correct if no security violation in any run.

Correctness vs Security

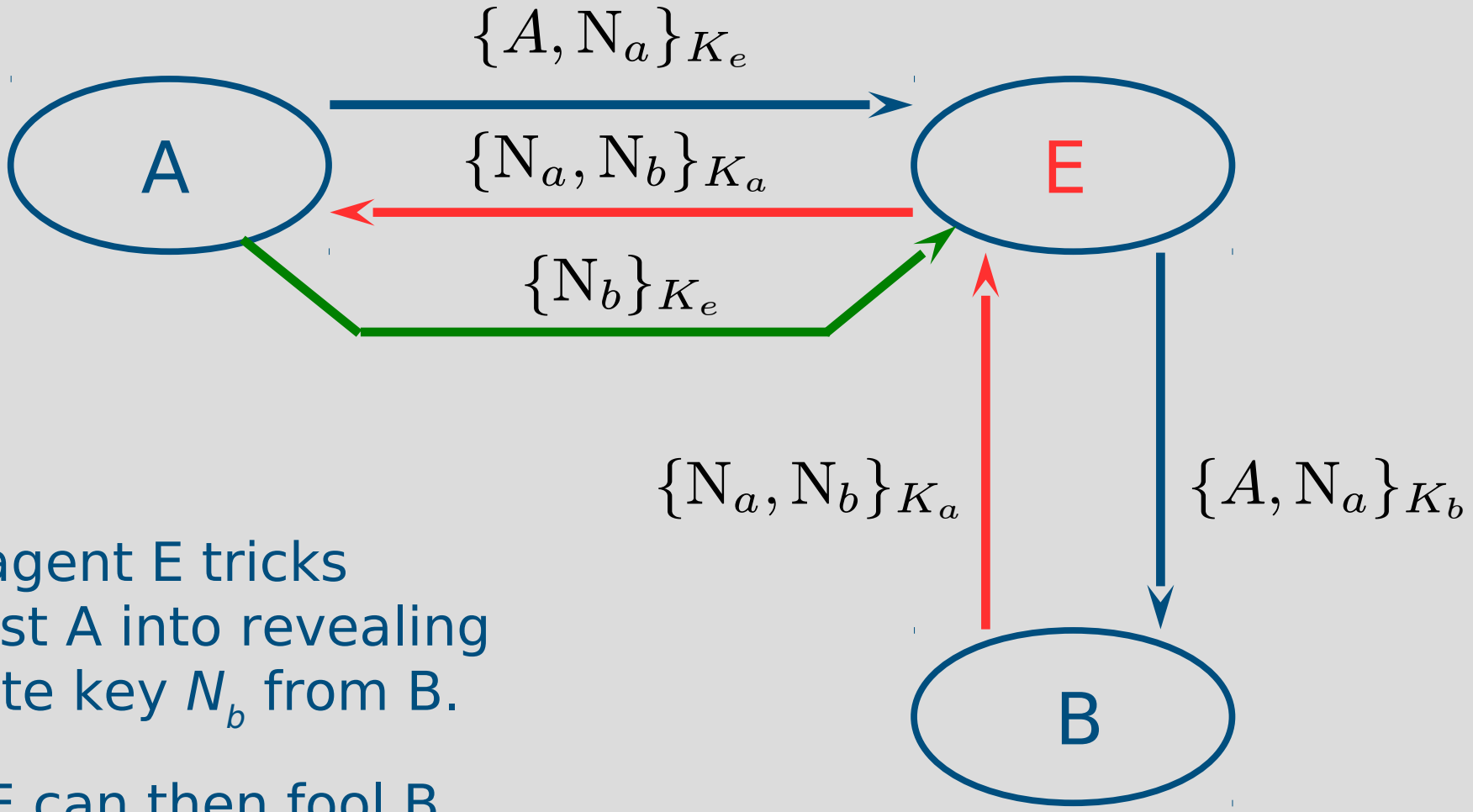
- Program or System Correctness
  - Program satisfies specification
    - For reasonable input, get reasonable output
- Program or System Security
  - Program resists attack
    - For unreasonable input, output not completely disastrous
- Main differences
  - Active interference from environment
  - Refinement techniques may fail

Needham-Schroeder Key Exchange



Result: A and B share two private numbers not known to any observer without K_a^{-1}, K_b^{-1}

Anomaly in Needham-Schroeder [Lowe]



Evil agent E tricks honest A into revealing private key N_b from B.

Evil E can then fool B.

Dolev-Yao intruder, e.g., as formalized in MSR [CSFW'99]

Dolev-Yao intruder, e.g., as formalized in MSR [CSFW'99]

Intercept/send messages:

$$N_S(x) \rightarrow M(x)$$

$$M(x) \rightarrow N_R(x)$$

Decompose messages:

$$M(\langle x, y \rangle) \rightarrow M(x), M(y)$$

Compose messages:

$$M(x), M(y) \rightarrow M(\langle x, y \rangle)$$

Create nonces:

$$\rightarrow \exists z.M(z)$$

Among other rules, *e.g.*, rules involving encryption/decryption.

Some of these rules are not balanced. In particular, the intruder has an unbounded memory, *i.e.*, he can remember as many facts as he needs.

The secrecy/planning problem is undecidable.

Memory Bounded Dolev-Yao intruder

How much adversarial behavior can be done by some insiders in a collaborative system?

Since insiders have a bounded memory, we need to consider a memory bounded Dolev-Yao intruder.

Memory Bounded Dolev-Yao intruder, sample rules

We use private facts of the form *R(*)* to denote a free memory slot available only to the intruder and public facts of the form *P(*)* to denote a memory slot available to all agents.

Memory Bounded Dolev-Yao intruder, sample rules

Intercept/send messages:

$$R(*), N_S(x) \rightarrow M(x), P(*)$$

$$P(*), M(x) \rightarrow N_R(x), R(*)$$

Decompose messages:

$$R(*), M(\langle x, y \rangle) \rightarrow M(x), M(y)$$

Compose messages:

$$M(x), M(y) \rightarrow M(\langle x, y \rangle), R(*)$$

Create nonces:

$$R(*) \rightarrow \exists z.M(z)$$

Intruder might need to forget information:

$$M(x) \rightarrow R(*)$$

We use private facts of the form *R(*)* to denote a free memory slot available only to the intruder and public facts of the form *P(*)* to denote a memory slot available to all agents.

Memory Bounded Dolev-Yao intruder

Memory management

Memory Bounded Dolev-Yao intruder

Memory management

Protocol roles may be created/deleted while other protocols are running.

$$A_k \rightarrow P(*)$$

$$Q_1(\vec{x}_1) \cdots Q_n(\vec{x}_n) P(*) \rightarrow Q_1(\vec{x}_1) \cdots Q_n(\vec{x}_n) A_0(\vec{x})$$

Well-founded theories [Cervesato *et al.,* CSFW'99] prohibit this. They only allow protocol roles to be created before any protocol runs take place. Hence they only allow for a bounded number of roles if actions are balanced.

PSPACE lower-bound using protocol theories

We rely upon the fact that NPSPACE = PSPACE.

We encode a deterministic Turing machine, *TM*, that accepts in space n^2.

Assume w.l.o.g. that the machine has only one accepting configuration.

We encode *TM* by using two participants *A* and *B*. *A* initiates the protocol, while *B* encodes the actions of the Turing machine *M* and also checks whether the current state is the accepting configuration of *TM*.

Theorem: Let *P(I,TM)* be a protocol theory encoding *TM* with initial configuration *I*. Let *M* be a balanced intruder theory. A run of theory *P(I,TM) + M* can lead to a state containing *M(secret)* if and only if the machine *TM* can reach the accepting configuration starting from *I*.

PSPACE lower-bound using protocol theories

Encoding *TM*'s configurations as messages

$$\langle \$\, \xi_1 \xi_2 \ldots \xi_i \ldots \xi_{n^2}\, \#, q, i \rangle \quad \text{or} \quad \langle \tau, q, i \rangle$$

$\$$ and $\#$ mark the beginning and the end of the tape.

ξ_j contains the symbol at the j[th] position in the tape.

q is the state of *TM*.

i is the position in the tape that *TM* is scanning.

We assume that no instruction leads *TM* to scan a position to the left of $ or to the right of #.

PSPACE lower-bound using protocol theories

Normal Run

$$
\begin{aligned}
A \longrightarrow B: &\quad \langle update, \{\langle \tau, q, i \rangle\}_k \rangle \\
B \longrightarrow A: &\quad \langle done, \{\langle \tau', q', i' \rangle\}_k \rangle \\
A \longrightarrow B: &\quad \langle check, \{\langle \tau', q', i' \rangle\}_k \rangle \\
B \longrightarrow A: &\quad result
\end{aligned}
$$

In the first two actions, *B* executes the unique *TM*'s instructions that changes the state from *q to q',* changing the contents of the tape, and *i' = i + 1* if the instruction moves *TM*'s head to the right, *or i' = i - 1* if the instruction moves *TM*'s head to the left, otherwise *i' = i .*

In the last two actions, *B* checks whether *q'* is the accepting state is reached. If it is then *result* is the *secret,* otherwise *result* is *no.*

Anomaly

First Session of the Anomaly

$$
\begin{aligned}
A \longrightarrow M \longrightarrow B: &\quad \langle update, \{\langle \tau, q, i\rangle\}_k\rangle \\
B \longrightarrow M \longrightarrow A: &\quad \langle done, \{\langle \tau', q', i'\rangle\}_k\rangle \\
A \longrightarrow M \longrightarrow B: &\quad \langle check, \{\langle \tau', q', i'\rangle\}_k\rangle \\
B \longrightarrow M \longrightarrow A: &\quad result
\end{aligned}
$$

Intruder is the man in the middle. He learns the initial state.

Later Sessions of the Anomaly

$$
\begin{aligned}
M(A) \longrightarrow B: &\quad \langle update, \{\langle \tau, q, i\rangle\}_k\rangle \\
B \longrightarrow M(A): &\quad \langle done, \{\langle \tau', q', i'\rangle\}_k\rangle \\
M(A) \longrightarrow B: &\quad \langle check, \{\langle \tau', q', i'\rangle\}_k\rangle \\
B \longrightarrow M(A): &\quad result
\end{aligned}
$$

Intruder impersonates *A*. After each session, the message exchanged encodes the next state of the Turing machine.

Formally in our system:

Protocol Theory for *A*

ROLA: $Guy(G,k)Init(I)P(*) \rightarrow_A Guy(G,k)Init(I)A_0(I,k)$

UPDA: $A_0(X,k)P(*) \rightarrow_A A_1(X,k)N_S(\langle update, enc(k,X)\rangle)$

CHKA: $A_1(X,k)N_R(\langle done, enc(k,Y)\rangle) \rightarrow_A A_2(Y,k)N_S(\langle check, enc(k,Y)\rangle)$

RESA: $A_2(X,k)N_R(Res) \rightarrow_A A_3(X,Res,k)P(*)$

ERASEA: $A_3(X,Res,k) \rightarrow_A P(*)$

Formally in our system:

Protocol Theory for *B*

$$
\begin{array}{ll}
\text{ROLB:} & Guy(G,k)Secret(s)P(*) \rightarrow Guy(G,k)Secret(s)B_0(k,s) \\
\text{UPDB:} & B_0(k,s)N_R(\langle update, enc(k,\langle x_0,\ldots,x_{i-1},\xi,x_{i+1},\ldots,x_{n^2+1},q,i\rangle)\rangle) \\
 & \rightarrow B_1(\langle x_0,\ldots,x_{i-1},\eta,x_{i+1},\ldots,x_{n^2+1},q',i'\rangle,k,s) \\
 & N_S(\langle done, enc(k,\langle x_0,\ldots,x_{i-1},\eta,x_{i+1},\ldots,x_{n^2+1},q',i'\rangle)\rangle) \\
\text{CHKB:} & B_1(X,k,s)N_R(\langle check, enc(k,X)\rangle) \rightarrow B_2(X,k,s)N_S(result) \\
\text{ERASEB:} & B_2(X,k,s) \rightarrow P(*)
\end{array}
$$

For each instruction in *TM* of the form:

$$q\xi \rightarrow q'\eta D$$

there are n^2 UPDB rules where $0 < i < n^2+1$ is the position *TM*'s head and the action above denotes that "if in state *q* and looking at ξ then replace it by η, move in the direction *D* and go to the state *q'*."

Memory Bounded Dolev-Yao intruder

Since all actions are balanced, the secrecy problem is PSPACE-complete.

This is one theoretical explanation of the successful use of model-checkers in the verification of security protocols. Our PSPACE upper bound can have some impact on practical aspects of protocol verification.

Analysis of the intruder's memory for known anomalies

Protocol	Needham-Schroeder	Yahalom	Otway-Rees	Woo-Lam	Kerberos 5	PKINIT
No intruder	Facts: 9	Facts: 8	Facts: 8	Facts: 7	Facts: 15	Facts: 18
With Anomaly	Facts: 19 R(*): 7	Facts: 15 R(*): 9	Facts: 11/17 R(*): 5/9	Facts: 8 R(*): 2	Facts: 22/20 R(*): 9/4	Facts: 31 R(*): 10
Size of Facts	6	16	26	6	16	28

No Upper Bound for the Dolev-Yao's Memory

Some known anomalies can be carried out by the Bounded Memory Dolev-Yao intruder if one gives him enough memory.

In particular, we considered all protocols to be bounded. That is, the agents participating cannot remember an unbounded number of facts. This is different from the setting in [*Cervesato et al., CSFW'99*]. In particular, for bounded protocols there in only a bounded number concurrent sessions.

No Upper Bound for the Dolev-Yao's Memory

This leads to the following question:

Is it possible to infer an upper-bound on the memory required by the Standard Dolev-Yao adversary to carry out an anomaly from the memory bound of the bounded protocol?

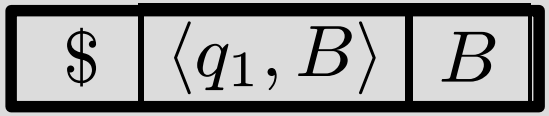
We answer this question negatively, confirming the hardness of protocol verification.

Encoding Turing Machines

In particular, we provide a sound and faithful encoding of Turing Machines using bounded memory protocols and the Standard Dolev-Yao adversary.

Assumptions (w.l.o.g.) about the machine *M*

- Only one tape, which is one-way unbounded to the right. The leftmost cell (numbered by 0) contains the marker $ unerased;
- The initial 3-cell configuration is of the following form, where **B** stands for the blank symbol:



- We assume that all instructions are “move”' instructions. The head of the machine cannot move to the leftmost cell marked with $.
- Only one accepting state q_0

Encoding Turing Machines

We use assume to principal, Alice and Bob, which share a symmetric key *K*;

Encoding of the Tape

• An unscanned cell that contains symbol ξ_0 is encoded by a term encrypted with the key K;

$$E_K(\langle t_0, \xi_0, e_0, t_1 \rangle)$$

where t_0 and t_1 are nonces, and $e_0 = 1$, if the cell is the last cell in a configuration.

• The cell that contains symbol ξ and is scanned by the machine *M* in state q is also encoded by a term encrypted with the key K:

$$E_K(\langle t_1, \langle q, \xi \rangle, 0, t_2 \rangle)$$

Encoding Turing Machines

The nonces t_0 and t_1 are used as “timestamps” and to specify the adjacency of cells.

Initial Configuration

$$\langle E_K(\langle t_0, \$, 0, t_1\rangle), E_K(\langle t_1, \langle q_1, B\rangle, 0, t_2\rangle), E_K(\langle t_2, B, 1, t_3\rangle)\rangle$$

Encoding Machine's Actions

Alice's Role – Alice is the initiator and her initial state is:

$$\langle E_K(\langle t_0, \$, 0, t_1\rangle), E_K(\langle t_1, \langle q, B\rangle, 0, t_2\rangle), E_K(\langle t_2, B, 1, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

Alice updates all nonces t_i to t'_i, and sends the following updated message to Bob:

$$\langle E_K(\langle t'_0, \$, 0, t'_1\rangle), E_K(\langle t'_1, \langle q, B\rangle, 0, t'_2\rangle), E_K(\langle t'_2, B, 1, t'_3\rangle), E_K(\langle t'_4, B, 1, t'_5\rangle)\rangle$$

Alice waits a response from Bob of the form:

$$\langle E_K(\langle t_0, \alpha_0, 0, t_1\rangle), E_K(\langle \widetilde{t_1}, \alpha_1, 0, \widetilde{t_2}\rangle), E_K(\langle t_2, \alpha_2, e_2, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

Alice checks whether $t_1 = \widetilde{t_1}$ and $\widetilde{t_2} = t_2$.

Moreover, if α_i is of the form $\langle q_0, \xi\rangle$ then, she releases the secret.

Encoding Turing Machines

Encoding Machine's Actions

Bob's Role – Bob transforms a message received with the help of an instruction from the given Turing machine. He expects a message of the form:

$$\langle E_K(\langle t_0, \xi_0, 0, t_1\rangle), E_K(\langle \widetilde{t_1}, \langle q, \xi\rangle, 0, \widetilde{t_2}\rangle), E_K(\langle t_2, \xi_2, e_2, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

If $t_1 = \widetilde{t_1}$ and $\widetilde{t_2} = t_2$, then he performs one of the following three actions:

1) Extends the tape – if $e_2 = 1$ Bob updates nonces t_i to t_i' , and sends the following updated message to Alice, which provides the chain of four cells with an updated last cell:

$$\langle E_K(\langle t_0, \xi_0, 0, t_1'\rangle), E_K(\langle t_1', \langle q, \xi\rangle, 0, t_2'\rangle), E_K(\langle t_2', \xi_2, 0, t_3'\rangle), E_K(\langle t_3', B, 1, t_4'\rangle)\rangle$$

Encoding Turing Machines

Encoding Machine's Actions

Message received by Bob:

$$\langle E_K(\langle t_0, \xi_0, 0, t_1\rangle), E_K(\langle \widetilde{t_1}, \langle q, \xi\rangle, 0, \widetilde{t_2}\rangle), E_K(\langle t_2, \xi_2, e_2, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

2) Moving the Head of the Machine to the Right – if $e_2 = 0$, for an instruction of the form

$$q\xi \rightarrow q'\eta R$$

Denoting "if in state q looking at symbol ξ, replace it by η, move the tape head one cell to the right, and go into state q "

Bob updates some nonces t_i to t'_i, and sends the following updated message to Alice:

$$\langle E_K(\langle t_0, \xi_0, 0, t'_1\rangle), E_K(\langle t'_1, \eta, 0, t'_2\rangle), E_K(\langle t'_2, \langle q', \xi_2\rangle, 0, t_3\rangle), E_K(\langle t_4, B, 1, t_5\rangle)\rangle$$

Encoding Turing Machines

Encoding Machine's Actions

Message received by Bob:

$$\langle E_K(\langle t_0, \xi_0, 0, t_1 \rangle), E_K(\langle \widetilde{t_1}, \langle q, \xi \rangle, 0, \widetilde{t_2} \rangle), E_K(\langle t_2, \xi_2, e_2, t_3 \rangle), E_K(\langle t_4, B, 1, t_5 \rangle) \rangle$$

3) Moving the Head of the Machine to the Left – if $e_2 = 0$, for an instruction of the form

$$q\xi \rightarrow q'\eta L$$

Denoting "if in state q looking at symbol ξ, replace it by η, move the tape head one cell to the left, and go into state q"

Bob updates some nonces t_i to t'_i, and sends the following updated message to Alice:

$$\langle E_K(\langle t_0, \langle q', \xi_0 \rangle, 0, t'_1 \rangle), E_K(\langle t'_1, \eta, 0, t'_2 \rangle), E_K(\langle t'_2, \xi_2, 0, t_3 \rangle), E_K(\langle t_4, B, 1, t_5 \rangle) \rangle$$

Man-in-the-Middle-Attack by the Intruder (Mallory)

Notice that by eavesdropping, Mallory can collect messages of the form:

$$E_K(\langle t_1, \alpha_1, e_1, t_2 \rangle)$$

Attack

• For the first run, Mallory intercepts the initial message from Alice, stores it, and resends it to Bob. While Bob responds, Mallory intercepts the message from Bob, stores it, and resends it to Alice.

• For each of the next runs, Mallory intercepts the initial message from Alice. Taking non-deterministically messages stored in his memory and composing the following message below, Mallory sends it to Bob:

$$\langle E_K(\langle t_0, \alpha_0, 0, t_1 \rangle), E_K(\langle \widetilde{t_1}, \alpha_1, 0, \widetilde{t_2} \rangle), E_K(\langle t_2, \alpha_2, e_2, t_3 \rangle), E_K(\langle t_4, B, 1, t_5 \rangle)$$

• If Bob accepts this message and responds with a transformed one as described in the protocol, then Mallory intercepts this new message from Bob, stores it, and resends it to Alice.

Man-in-the-Middle-Attack by the Intruder (Mallory)

Lemma: Suppose that a term of the form below appears in the intruder memory by active eavesdropping.

$$E_K(\langle t, \langle q, \xi \rangle, 0, t' \rangle)$$

Then there is a unique sequence of nonces $t_0, t_1, \ldots, t_{n+2}$ and a chain of terms from the adversary's memory:

$$E_K(\langle t_0, \$, 0, t_1 \rangle),\ E_K(\langle t_1, x_1, 0, t_2 \rangle), \ldots .\ E_K(\langle t_{j-1}, x_{j-1}, 0, t_j \rangle),$$
$$E_K(\langle t_j, \langle q, x_j \rangle, 0, t_{j+1} \rangle),\ E_K(\langle t_{j+1}, x_{j+1}, 0, t_{j+2} \rangle), \ldots,\ E_K(\langle t_n, x_n, 0, t_{n+1} \rangle),$$
$$E_K(\langle t_{n+1}, B, 1, t_{n+2} \rangle)$$

such that

$$t_j = t,\ \ x_j = \xi, \text{ and } \ t_{j+1} = t'$$

and *M* leads from the empty initial configuration to the configuration where the string $x_1 x_2 \ldots x_j \ldots x_n$ is written in cells 1,…,n on the tape, where the j-th cell is scanned by *M* in state q.

Theorem: There is a Dolev-Yao attack on the above protocol if and only if the machine *M* terminates on the empty input.

Comparison with Related Work

	Bound on the size of facts	Bound on the number of protocol sessions	Bound on the number of nonces	Bound on the number of parallel protocol sessions	Bounded Memory Intruder	Protocol Theories
PSPACE-complete [Kanovich et al]	Yes	No	No	Yes	Yes	No
DEXPTIME-complete [Durgin et al]	Yes	No	Yes	No	No	Yes
NP-complete [Amadio and Lugiez]	No	Yes	Yes	Yes	No	Yes
NP-complete for Progressing [new]	Yes	Yes	Yes	Yes	Yes	No

Conclusions

- Balanced systems provide an intuitive restriction to the memory capabilities of agents: each agent can store at any moment a bounded number of facts of bounded size;
- We provide a formalization for notion of freshness for balanced systems with balanced actions that can create fresh values: a nonce uses the space previously used by the updated value;
- We prove that in such systems the planning problem is PSPACE-complete;
- Returning to protocol security, we show that known protocol anomalies can also occur when the intruder has bounded memory and that the secrecy becomes PSPACE-complete;
- We showed that it is not possible to infer a computable upper bound on the Dolev-Yao's memory from the memory bound of protocols, confirming thus the hardness of protocol verification. This was done by a novel undecidability proof for the secrecy problem;
- We proposed a novel definition of Progressing Collaborative Systems for systems that may create fresh values;
-
- We showed that the reachability problem for balanced Progressing Collaborative Systems that can create fresh values is NP-complete.

Future work

• How to include Real Time into our model, and in particular find decidable fragments for the reachability problem. Many distance bounding protocols mention real time and are of great interest to protocol security community.

• Can our complexity results help the design of protocol verification tools? We are currently using Maude.

• Enrich our intruder model to include new parameters, such as the number of active concurrent protocol sessions, to provide richer quantitative measures of security of a protocol;

• Investigate ways to lift the assumption that the size of facts is bounded;

Related Work

- A. W. Roscoe. Proving security protocols with model checkers by data independence techniques, 1998.
- Harrison, Ruzzo, Ullman. On protection in operating systems, 1975.
- Amadio, Lugiez. On the reachability problem in cryptographic protocols, 2000.
- Amadio, Lugiez, Vanackere. On the symbolic reduction of processes with cryptographic functions, 2003.
- Rusinowitch, Turuani. Protocol insecurity with a finite number of sessions and composed keys is NP-complete, 2003.
- Chevalier, Kusters, Rusinowitch,Turuani. An NP decision procedure for protocol insecurity with xor, 2003.
- Comon-Lundh, Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or, 2003.
- Lam, Mitchell, Sundaram. A formalization of HIPAA for a medical messaging system, 2009.
- Esparza, Nielsen. Decidability issues for Petri nets - a survey, 1994.