Engineering Secure Software Systems

November 17, 2020: Crypto Protocols: Formal Model

Henning Schnoor

Institut für Informatik, Christian-Albrechts-Universität zu Kiel

Part I: Crypto Protocols

Overview

Part I: Crypto Protocols

- Foundations
- Cryptography
- An Example and an Attack
- More Examples

Formal Protocol Model

- Message Construction: Dolev-Yao Closure
- Algorithm: Computing the Dolev-Yao Closure
- Message Parsing and Delivery: Receive/Send Actions, Substitutions, Matching
- Protocol Specifications: Instances and Protocols
- Sessions and Scheduling: Execution Orders

Overview

Part I: Crypto Protocols

- Foundations
- Cryptography
- An Example and an Attack
- More Examples

Formal Protocol Model

- Message Construction: Dolev-Yao Closure
- Algorithm: Computing the Dolev-Yao Closure
- Message Parsing and Delivery: Receive/Send Actions, Substitutions, Matching
- Protocol Specifications: Instances and Protocols
- Sessions and Scheduling: Execution Orders

Dolev-Yao Closure Example

situation: PKI, shared keys, look at Charlie

$$S = \left\{ \hat{k}_C, k_A, k_B, k_C, k_{AC}, k_{BC}, N_C^1, N_C^2, \mathrm{enc}^{\mathrm{s}}_{k_{BC}} \left(\mathrm{enc}^{\mathrm{a}}_{k_C} \left(\mathrm{enc}^{\mathrm{s}}_{k_{AB}} (N_A) \right) \right) \right\}$$

derivable?

- $\mathrm{sig}_{k_C} \left(\mathrm{enc}^{\mathrm{s}}_{k_{AB}} (N_A) \right)$? yes
- $\mathrm{sig}_{k_C} (N_A)$? no
- $\mathrm{sig}_{k_A} \left(\mathrm{enc}^{\mathrm{s}}_{k_{AB}} (N_A) \right)$? no

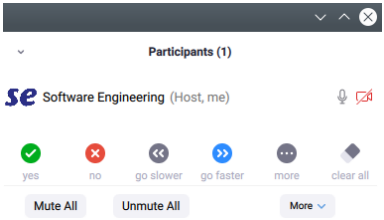
Dolev Yao Closure: Examples

initial adversary knowledge

$$I = \left\{A, B, \hat{k}_I, k_{AI}, k_{BI}, 0, 1, \text{yes}, \text{no}\right\},$$

knowledge grows with each message

can adversary derive terms?



messages

$\mathcal{A}$ receives	goal	derivable?
$\text{enc}^{a}_{k_A}$ (secret)	secret	✗
$\text{enc}^{a}_{k_I}$ (secret)	secret	✓
$\text{enc}^{s}_{k_{AB}}$ (yes)	yes	✓
$\text{enc}^{s}_{k_{AB}}$ (N_A)	N_A	✗
$\text{enc}^{s}_{k_{AI}}$ (k_{AB})	N_A	✓
$\text{enc}^{s}_{k_A}$ ([0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1])	[0, . . . , 1]	✓
$\text{enc}^{s}_{[0,1,1,0,0,1,1,\ldots,0,1,1]}$ (N_B)	N_B	✓

consequence

- arbitrarily long bit sequences always “known”
- do not model: adversary knows that **this message** contains “yes”
- ⇝ generalization later

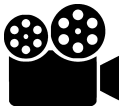
Overview

Part I: Crypto Protocols

- Foundations
- Cryptography
- An Example and an Attack
- More Examples

Formal Protocol Model

- Message Construction: Dolev-Yao Closure
- Algorithm: Computing the Dolev-Yao Closure
- Message Parsing and Delivery: Receive/Send Actions, Substitutions, Matching
- Protocol Specifications: Instances and Protocols
- Sessions and Scheduling: Execution Orders



Computing the Dolev-Yao Closure

`https://cloud.rz.uni-kiel.de/index.php/s/No4yD7SPJaYz4wn`

video content

- characterization of $\mathsf{DY}(S)$ with *derivation rules*
- properties of “minimal derivations”
- a fixpoint algorithm for “computing” $\mathsf{DY}(S)$

study

- watch video—feedback welcome!
- video slides contained in slide set (gray background), additional material in lecture notes
- next week: discussion of content (in small groups), bring questions!

Goal: Compute Dolev-Yao Closure

Dolev-Yao

- proofs of insecurity (security): argue that adversary can (not) send message m
- need formal criterion of messages that adversary can send
- $\mathsf{DY}(S)$: set of messages the adversary can derive from S

long-term goal: automatic security analysis

need algorithm for $\mathsf{DY}(S)$

obstacle to computation

- $\mathsf{DY}(S)$ is infinite: ϵ, $[\epsilon, \epsilon]$, $[\epsilon, [\epsilon, \epsilon]]$, …
- algorithm cannot "write down" $\mathsf{DY}(S)$

way out

- we do not need to enumerate $\mathsf{DY}(S)$
- suffices to algorithmically answer question
  can adversary send m?

Result: Decision Procedure

decision problem

Problem:	DERIVE
Input:	set of terms S, term m
Question:	is $m \in \mathsf{DY}(S)$?

theorem

DERIVE can be decided in polynomial time.

reference

Michaël Rusinowitch and Mathieu Turuani. “Protocol insecurity with a finite number of sessions, composed keys is NP-complete”. In: Theoretical Computer Science 1-3.299 (2003), pp. 451–475

Technique: Proof Overview

steps

- characterization of Dolev-Yao Closure with **derivation rules**
- deciding whether $m \in \mathsf{DY}(S)$ is deciding whether there is a derivation of m from S
- issue: infinite search space of derivations
- solution:
  - if there is a derivation of m from S, then there is a shortest one
  - a shortest derivation contains no unnecessary steps
  - this restricts the search space

simplification

- to simplify case distinctions: only encryption, pairing, nonces, constants in this proof
- arguments suffice to also cover signatures, MACs and hash functions, …
- see exercise task for generalization

Tool: Derivation Rules

rules

for a message m, rule $L_d(m)/L_c(m)$ describes how m can be **decomposed**/**composed**

this potentially needs **prerequisites**:

- composing $\mathbf{sig}_{k_B}(m)$ needs m and $\hat{k}_B$
- decomposing $\mathbf{enc}^{s}_{k_{AB}}(m)$ needs k_{AB}

a rule consists of a set R of **required** and and a set O of **obtained** terms, written $R \to O$. In the specific rules, we omit set brackets.

composition rules

$L_c([a,b])$	$a, b \to [a,b]$
$L_c(\mathbf{enc}^{a}_{k_A}(m))$	$m \to \mathbf{enc}^{a}_{k_A}(m)$
$L_c(\mathbf{enc}^{s}_{t_k}(m))$	$m, t_k \to \mathbf{enc}^{s}_{t_k}(m)$

decomposition rules

$L_d([a,b])$	$[a,b] \to a, b$
$L_d(\mathbf{enc}^{a}_{k_A}(m))$	$\mathbf{enc}^{a}_{k_A}(m), \hat{k}_A \to m$
$L_d(\mathbf{enc}^{s}_{t_k}(m))$	$\mathbf{enc}^{s}_{t_k}(m), t_k \to m$

Application of Rules: Derivations

definition

derivation: sequence $S_0 \to_{L_0} S_1 \to_{L_1} \ldots S_{n-1} \to_{L_{n-1}} S_n$, such that, for all $i \in \{0, \ldots, n-1\}$,

- L_i is a rule of the form $S \to S'$
- $S \subseteq S_i$
- $S_{i+1} = S_i \cup S'$

intuition

S_{i+1} obtained from S_i with L_i

Characterization: Dolev-Yao Closure Captured by Derivation Rules

lemma & definition

If $m \in \mathsf{DY}(S)$ where $\mathsf{IDs} \cup \{k_a \mid a \in \mathsf{IDs}\} \cup \{\epsilon\} \subseteq S$, then there is a derivation of m from S:

$$S = S_0 \to_{L_0} S_1 \to_{L_1} \dots S_{n-1} \to_{L_{n-1}} S_n$$

with $m \in S_n$. We call n the length of the derivation.

proof

see exercise

definition

For $m \in \mathsf{DY}(S)$, let $D_S(m)$ be a (fixed) shortest derivation of m from S. We write $L \in D_S(m)$, if L is a rule applied in $D_S(m)$.

Exercise

Task (DY closure and derivations)

In the lecture, the following lemma was stated (without proof):

If S is a set with $\mathsf{IDs} \cup \{k_a \mid a \in \mathsf{IDs}\} \cup \{\epsilon\} \subseteq S$ and $m \in \mathsf{DY}(S)$, then there is a derivation of m from S: $S = S_0 \to_{L_0} S_1 \to_{L_1} \ldots S_{n-1} \to_{L_{n-1}} S_n$ with $m \in S_n$.

1. Prove the above lemma.
2. State and prove an appropriate converse of the lemma.

Note: As in the lecture, you can assume that both S and m do not contain applications of hash functions, message authentication codes (MACs), or signatures.

Derivation Rules Property: One-Step Effects Only

composition

$L_c([a,b])$	$a, b \rightarrow [a,b]$
$L_c(\mathrm{enc}^{\mathrm{a}}_{k_A}(m))$	$m \rightarrow \mathrm{enc}^{\mathrm{a}}_{k_A}(m)$
$L_c(\mathrm{enc}^{\mathrm{s}}_{t_k}(m))$	$m, t_k \rightarrow \mathrm{enc}^{\mathrm{s}}_{t_k}(m)$

decomposition

$L_d([a,b])$	$[a,b] \rightarrow a, b$
$L_d(\mathrm{enc}^{\mathrm{a}}_{k_A}(m))$	$\mathrm{enc}^{\mathrm{a}}_{k_A}(m), \hat{k}_A \rightarrow m$
$L_d(\mathrm{enc}^{\mathrm{s}}_{t_k}(m))$	$\mathrm{enc}^{\mathrm{s}}_{t_k}(m), t_k \rightarrow m$

notation

$t \in \mathcal{T}$, then $\mathbf{Sub}^1(\mathrm{t})$ set of **direct subterms** of t

- $\mathbf{Sub}^1([\mathrm{t}_1, \mathrm{t}_2]) = \{t_1, t_2\}$
- $\mathbf{Sub}^1\left(\mathrm{enc}^{\mathrm{s}}_{\mathrm{k}}(\mathrm{t})\right) = \{k, t\}$
- $\mathbf{Sub}^1(\mathrm{hash}(\mathrm{t})) = \{t\}$
- …

direct successors in tree representation

observation

rules only work on $\mathbf{Sub}^1(.)$-level

- **composition rule** $L_c(m)$ has all terms from $\mathbf{Sub}^1(\mathrm{m})$ as prerequisites
- **decomposition rules** $L_d(m)$ obtains only terms from $\mathbf{Sub}^1(\mathrm{m})$

Reduce Search Space: Properties of Shortest Derivation

lemma

$D_S(m)$ shortest derivation of m from S, then:

1. If $L_d(t) \in D_S(m)$, then $t \in \mathsf{Sub}(\mathsf{S})$.
2. If $L_c(t) \in D_S(m)$, then $t \in \mathsf{Sub}(\mathsf{S} \cup \{\mathsf{m}\})$.

relevance

to derive m from S, we only need

1. decompositions of subterms from S
2. compositions of subterms of S or subterms of m

let's prove this!

written proof also contained in lecture notes

Exercise

Task (minimal derivation properties)

In the video lecture on the computation of the Dolev-Yao closure, we proved a lemma characterizing shortest derivations.

1. Can you generalize this result to handle signatures, MACs, and hash functions?
2. Which properties does the modeling of cryptographic primitives have to satisfy for an analog of this result to hold?
3. Can you come up with a modeling of cryptographic primitives where this property does not hold?

Algorithm for DERIVE

Input: set $S \neq \emptyset$ of messages, message m
  $S_{old} = \emptyset$
  while $S_{old} \neq S$ do
    $S_{old} = S$
    if ex. rule $S \rightarrow_L S \cup \{t\}$, $t \in \mathrm{Sub}(S \cup \{m\}) \setminus S$ then
      $S = S \cup \{t\}$
    end if
  end while
  if $m \in S$ then
    accept
  end if
  reject

algorithm uses previous results

- uses result on steps appearing in minimal derivations
- fixpoint algorihm: expands set S until fix point reached
- terminates in polynomial time since there are only polynomially many choices for t

covered in exercise

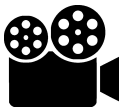
- algorithm correctness
- cannot “decompose first, compose later”

Exercise

Task (DY algorithm correctness)

Prove that the algorithm for computing the DY closure (in its decisional variant **DERIVE**) as stated in the lecture is correct and runs in polynomial time. As in the lecture, restrict yourself to terms without applications of hash functions, signatures, or message authentication codes (MACs).

Video Lecture: Feedback wanted



questions

- audio/video quality?
- proof presentation as screenshots, or “live writing?”
- better as video or “live Zoom session?”
- any suggestions?

feedback crucial

- your perspective very different from mine!
- constructive criticism always welcome
- review after week 6!

remember

- we’re all still learning this
- new tools, concepts
- big playground :-)

Plan for Review Sessions

purpose, timing

- used after self-study material (videos)
- purpose: discussions / questions about content (usually proofs)
  - mainly: your questions
  - some: review questions
  - **no prepared material**, that's the point!
- length/time: full or partial next session
  - synchronize schedule with last course iteration

„Wilke model": meet in smaller groups

- 2-3 groups, depending on number of participants (OLAT registration)
- groups for strong theory background / more basic theory knowledge
- please choose "fitting group," otherwise discussed questions might not match your needs

Overview

Part I: Crypto Protocols

- Foundations
- Cryptography
- An Example and an Attack
- More Examples

Formal Protocol Model

- Message Construction: Dolev-Yao Closure
- Algorithm: Computing the Dolev-Yao Closure
- Message Parsing and Delivery: Receive/Send Actions, Substitutions, Matching
- Protocol Specifications: Instances and Protocols
- Sessions and Scheduling: Execution Orders

Definition: Receive/Send Actions

formalize protocol instruction

parse incoming message, send reply

receive/send actions

receive/send action: pair $(r, s) \in \mathcal{T} \times \mathcal{T}$, write $r \to s$.

example from Needham-Schroeder

Bob's rule: $\mathsf{enc}^{\mathsf{a}}_{k_B}(A, x) \to \mathsf{enc}^{\mathsf{a}}_{k_A}(x, N_B)$

- k_A, k_B, A: known, assume knowledge of $\hat{k}_B$
- N_B: new nonce (generated by Bob)
- x: references Alice's nonce, repeated in Bob's response

variable handling

- x: stores (supposedly) nonce from Alice
- nonce (value of x) potentially used again later in protocol, must be stored

Needham Schroeder (informal)

$$
\begin{aligned}
A &\to B \quad \mathsf{enc}^{\mathsf{a}}_{k_B}(A, N_A) \\
B &\to A \quad \mathsf{enc}^{\mathsf{a}}_{k_A}(N_A, N_B) \\
A &\to B \quad \mathsf{enc}^{\mathsf{a}}_{k_B}(N_B)
\end{aligned}
$$

recall

Bob's (Alice's) protocol description must not contain N_A (N_B, N_C, …).

Definition: Substitutions

definition: substitutions

- substitution: function $\sigma\colon \mathcal{V} \to \mathcal{T}$ with $\sigma(x) \neq x$ for a finite number of x
- σ **ground substitution**, if $\sigma(x)$ message for all x with $\sigma(x) \neq x$.

intuition

- finite local memory of participants
- $\sigma(x) = x$: “uninitialized” variable

extension to terms

for $t \in \mathcal{T}$, σ substitution, $\sigma(t)$ defined inductively:

- $\sigma(x)$ defined for $x \in \mathcal{V}$
- $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$

examples

$\rightsquigarrow$ lecture notes

Receiving and Parsing a Message

central step: react to incoming message

- state: $\sigma(x) \neq x$ for for some x, next r/s action is $r \to s$
- incoming message: m
- reaction: updated substitution σ', reply message

Alice

- substitution: $\sigma(x) = N_B^1$, $\sigma(y) = y$
- next step: $(x, y, N_A^1) \to \mathsf{enc}^{\mathsf{a}}_{k_B}(y, N_A^2)$
- incoming message: $(N_B^1, (\mathtt{ok}, N_B^2), N_A^1)$

action:

- set $\sigma'(y) = (\mathtt{ok}, N_B^2)$
- send $\mathsf{enc}^{\mathsf{a}}_{k_B}((\mathtt{ok}, N_B^2), N_A^2)$

Bob

- substitution: $\sigma(z) = N_A^1 \neq N_A^2$
- next step: $\mathsf{enc}^{\mathsf{a}}_{k_B}((\mathtt{ok}, N_B^2), z) \to \mathsf{enc}^{\mathsf{a}}_{k_A}(N_B^3)$
- incoming message: $\mathsf{enc}^{\mathsf{a}}_{k_B}((\mathtt{ok}, N_B^2), N_A^2)$

action:

incoming message cannot be parsed with receive/send rule, no action taken

Matching: Applying Receive/Send Actions

situation in protocol run

- memory: substitution σ
- next action: $r \to s$
- incoming message: m

parsing m with $r \to s$

- update substitution to σ'
- outgoing term: $\sigma'(s)$

definition: matching

a term r matches with message m and substitution σ via substitution σ', if

- $\sigma'(r) = m$, and — σ' consistent with incoming message
- $\sigma'(x) = \sigma(x)$ for all x with $\sigma(x) \neq x$. — σ' consistent with state

Matching: Example

motivation

- matching: checks whether incoming term fits expectations
- expectations depend on
  - next rule in the protocol: receive/send rule from protocol
  - terms seen previously in protocol run: current substitution σ

example situation

- next receive/send rule:
  $$\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(x^1_A, N^1_A\right), \mathrm{sig}_{k_B}\left(x^2_A, y\right)\right) \rightarrow \mathrm{sig}_{k_A}\left(y, x^1_A, x^2_A, N^1_A, N^2_A\right)$$
- substitution:
  - $\sigma(x^1_A) = N^1_B$
  - $\sigma(x^2_A) = N^2_B$
  - $\sigma(y) = y$

reactions to incoming terms

matches? resulting substitution/reply?

- $\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(N^1_B\right), \mathrm{sig}_{k_B}\left(N^2_B, N_C\right)\right)$
- $\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(N^2_B, N^1_A\right), \mathrm{sig}_{k_B}\left(N^1_B, N_C\right)\right)$
- $\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(N^1_B, N^1_A\right), \mathrm{sig}_{k_B}\left(N^2_B, N_C\right)\right)$
- $\left(\mathrm{enc}^{\mathrm{a}}_{k_A}\left(N^1_B, N^1_A\right), \mathrm{sig}_{k_B}\left(N^2_B, N^2_B\right)\right)$

Overview

Part I: Crypto Protocols

- Foundations
- Cryptography
- An Example and an Attack
- More Examples

Formal Protocol Model

- Message Construction: Dolev-Yao Closure
- Algorithm: Computing the Dolev-Yao Closure
- Message Parsing and Delivery: Receive/Send Actions, Substitutions, Matching
- Protocol Specifications: Instances and Protocols
- Sessions and Scheduling: Execution Orders

Formal Protocol Definition

definition: protocol instance $\mathcal{I}$
sequence of actions

- $r_0 \to s_0$,
- $r_1 \to s_1$,
- $\ldots$,
- $r_{n-1} \to s_{n-1}$

with $\mathcal{V}(s_i) \subseteq \cup_{j \leq i} \mathcal{V}(r_i)$ for all i.
($\mathcal{V}(t)$: variables in term t)

example role

1. $\epsilon \to \mathsf{enc}^{\mathsf{a}}_{k_B}(A, N_A)$
2. $\mathsf{enc}^{\mathsf{a}}_{k_A}(N_A, x) \to \mathsf{enc}^{\mathsf{a}}_{k_B}(x)$

consequences for modeling

what kind of protocols can (can't) we express?

definition: protocol

protocol consists of

- instances $\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}$, and
- a finite set $\mathbf{I}$ of messages (the initial adversary knowledge).

example

formal representation of the Needham-Schroeder protocol

protocol

$$
\begin{array}{ll}
A \to B & \mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_a) \\
B \to A & \mathrm{enc}^{\mathrm{a}}_{k_A}(N_a, N_b) \\
A \to B & \mathrm{enc}^{\mathrm{a}}_{k_B}(N_b)
\end{array}
$$

formalization

Alice

$$
\begin{array}{lcl}
\epsilon & \to & \mathrm{enc}^{\mathrm{a}}_{k_B}(A, N_A) \\
\mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, y) & \to & \mathrm{enc}^{\mathrm{a}}_{k_B}(y)
\end{array}
$$

Bob

$$
\mathrm{enc}^{\mathrm{a}}_{k_B}(A, x) \to \mathrm{enc}^{\mathrm{a}}_{k_A}(x, N_B)
$$

Overview

Part I: Crypto Protocols

- Foundations
- Cryptography
- An Example and an Attack
- More Examples

Formal Protocol Model

- Message Construction: Dolev-Yao Closure
- Algorithm: Computing the Dolev-Yao Closure
- Message Parsing and Delivery: Receive/Send Actions, Substitutions, Matching
- Protocol Specifications: Instances and Protocols
- Sessions and Scheduling: Execution Orders

Modeling Protocols: Attack?

components: instances

- contain r/s actions
- example NSL: $\mathsf{enc}^{\mathsf{a}}_{k_B}(A, x) \rightarrow \mathsf{enc}^{\mathsf{a}}_{k_A}(x, N_B)$
- hard-coded assumption: Bob replies to Alice

fixed by protocol

- actual “protocol” (r/s actions)
- participants and “roles”
  - Alice: initiator
  - Bob: responder

issue

- protocol as formalized cannot be attacked!
- need different situation to show attack …

example

formal representation of the Needham-Schroeder protocol with attacker

messages by A, B

$A \to C(\mathcal{A})$ $\quad \mathrm{enc}^{\mathrm{a}}_{k_C}(A, N_A)$

$B \to A(\mathcal{A})$ $\quad \mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, N_B)$

$A \to C(\mathcal{A})$ $\quad \mathrm{enc}^{\mathrm{a}}_{k_C}(N_B)$

formalization

Alice

$$\begin{array}{lcl} \epsilon & \to & \mathrm{enc}^{\mathrm{a}}_{k_C}(A, N_A) \\ \mathrm{enc}^{\mathrm{a}}_{k_A}(N_A, y) & \to & \mathrm{enc}^{\mathrm{a}}_{k_C}(y) \end{array}$$

Bob

$$\mathrm{enc}^{\mathrm{a}}_{k_B}(A, x) \quad \to \quad \mathrm{enc}^{\mathrm{a}}_{k_A}(x, N_B)$$

Protocol Model

saw

formal protocol model, example for protocol run

crucial question

when is a protocol secure?

need: attacker model

- completely controls network
- can control participants (obtain their private keys)
- **also:** can start protocol sessions!

intuition

- Needham-Schroeder: attack when Alice “starts protocol with $\mathcal{A}$ (C)”
- to find attack: adversary must be able to “start protocol”
- also: attacker controls interleaving

our model: sessions (for now) part of the protocol

consequence?

Executing Instances I

situation: instances given

1. Alice as initiator with Charlie ($\mathcal{A}$)
2. Bob as responder with Alice (played by $\mathcal{A}$)

adversary:

- $\mathcal{A}$ controls C (knows C's private key)
- $\mathcal{A}$ impersonates A in session with Bob

attack works only with this order

allow $\mathcal{A}$ to control order

execution steps

1. $A \to C$
2. $A \to B$
3. $B \to A$
4. $C \to A$
5. $A \to C$
6. $A \to B$

Execution Order

intuition

$P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$ protocol

- each instance $\mathcal{I}_j$ has $|\mathcal{I}_j|$ steps
- instance $\mathcal{I}_j$ activated "at most $|\mathcal{I}_j|$ times"
- order **inside** $\mathcal{I}_j$: fixed by protocol

definition: execution order

$P = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n-1}\}$ protocol. An execution order for P is a sequence o over $\{0, \ldots, n-1\}$ such that each $j \in \{0, \ldots, n-1\}$ appears at most $|\mathcal{I}_j|$ times.

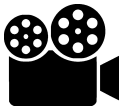
notation

- $o(t)$: t-th element in o
- $\#o(t)$: $|\{\ell \mid \ell < t \text{ and } o(\ell) = o(t)\}|$

position t

$\#o(t)$-th step of instance $\mathcal{I}_{o(t)}$

Next Session: Review Questions



Computing the Dolev-Yao Closure

`https://cloud.rz.uni-kiel.de/index.php/s/No4yD7SPJaYz4wn`

video content

- characterization of $\mathsf{DY}(S)$ with *derivation rules*
- properties of “minimal derivations”
- a fixpoint algorithm for “computing” $\mathsf{DY}(S)$

study

- watch video—feedback welcome!
- video slides contained in slide set (gray background), additional material in lecture notes
- next week: discussion of content (in small groups), bring questions!

Next Session

review questions

- we will start the session with discussing review questions
- 5-15 minutes, depending on
  - time (I will roughly follow last year's schedule)
  - participation

your preparation

- review lecture notes up to today
- try to answer review questions marked "during semester"

your participation

- to have a nice discussion: activate cameras!
- come with follow-up questions or ideas for answers!
- present in class orally or via screen-sharing

before we go

any questions?

Thanks!

"See you" next time!

References

Michaël Rusinowitch and Mathieu Turuani. “Protocol insecurity with a finite number of sessions, composed keys is NP-complete”. In: Theoretical Computer Science 1-3.299 (2003), pp. 451–475.