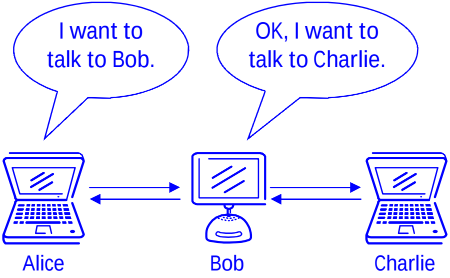
Selecting Theories and Nonce Generation for Recursive Protocols

Klaas Ole Kürtz, Ralf Küsters, Thomas Wilke

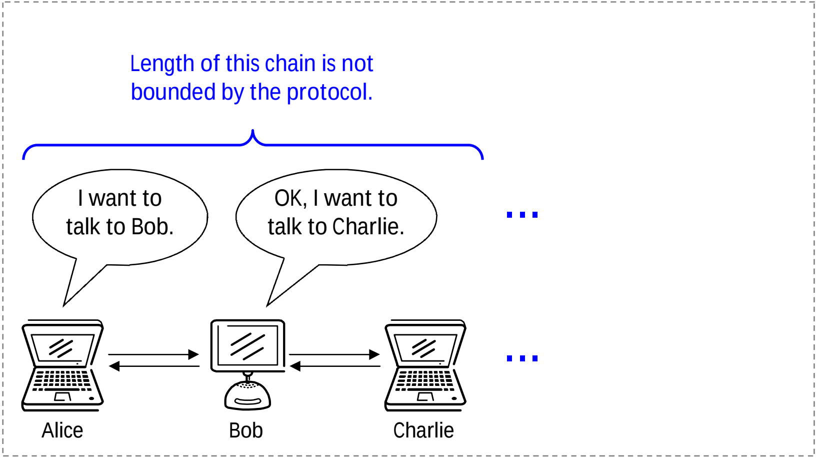
Klaas Ole Kürtz and Thomas Wilke
Christian-Albrechts-Universität
Kiel, Germany

Ralf Küsters
ETH Zürich
Zurich, Switzerland

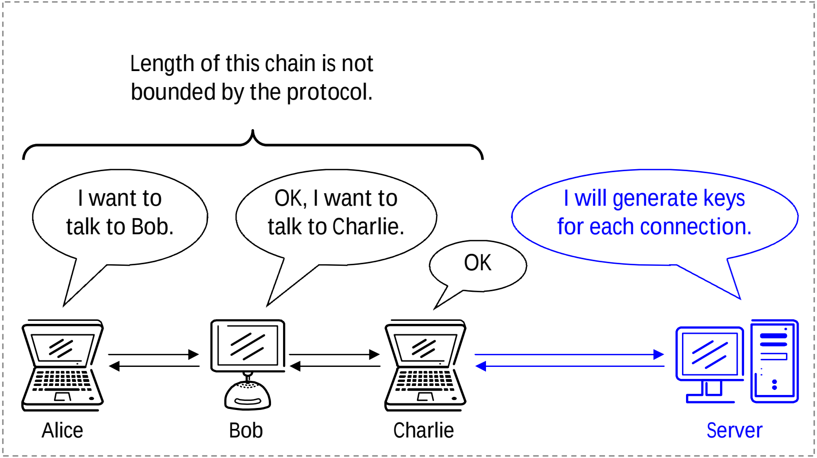
FMSE 2007, Fairfax Virginia (USA), November 2nd, 2007



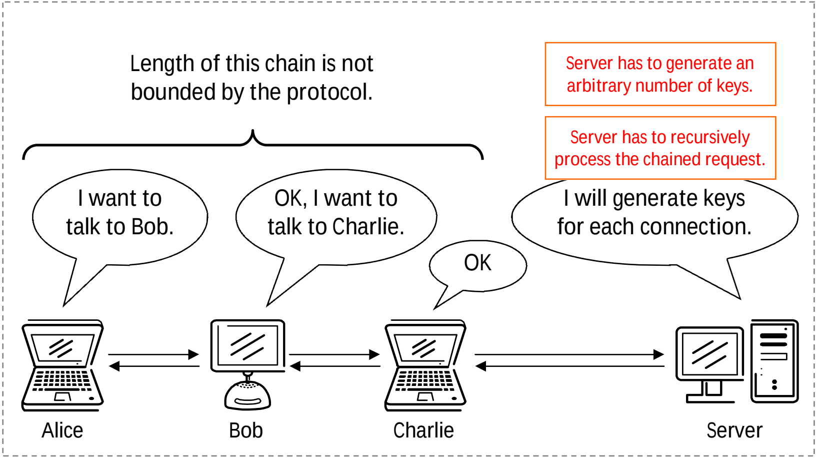
The Recursive Authentication Protocol (Bull, Otway, Paulson, 1997) allows a **chain of connections**.



The length of the chain, i. e., the number of principals, is **not bounded** by the protocol.



Each principal P shares a symmetric key K_P with a **server** that will **generate session keys** K_{AB} and K_{BC}.



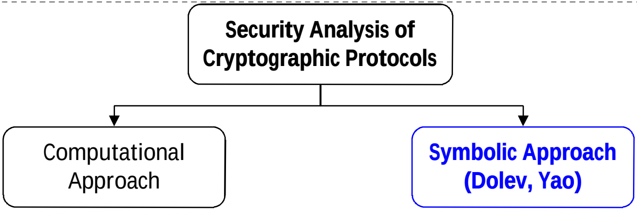
The depth of the request message and thus the number of keys that have to be generated by the server are also **not bounded** by the protocol.

Outline

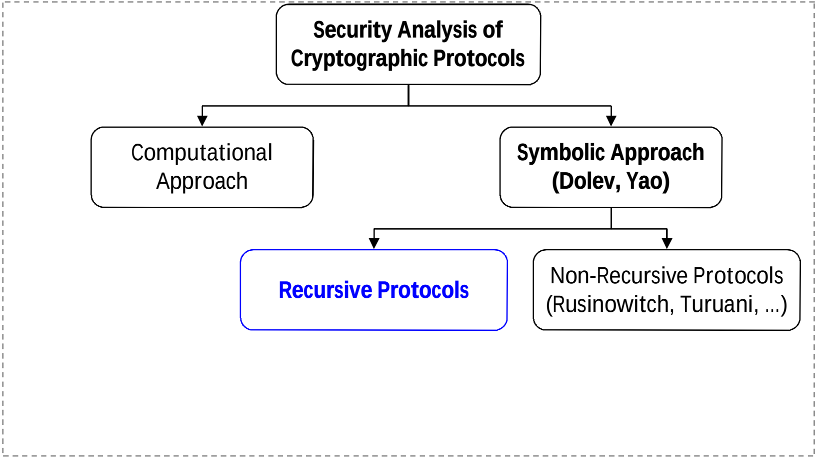
1. The Problem
2. Our Protocol Model
3. (Un)Decidability Results
4. The Technical Heart
5. Conclusion and Outlook

Security Analysis of Cryptographic Protocols

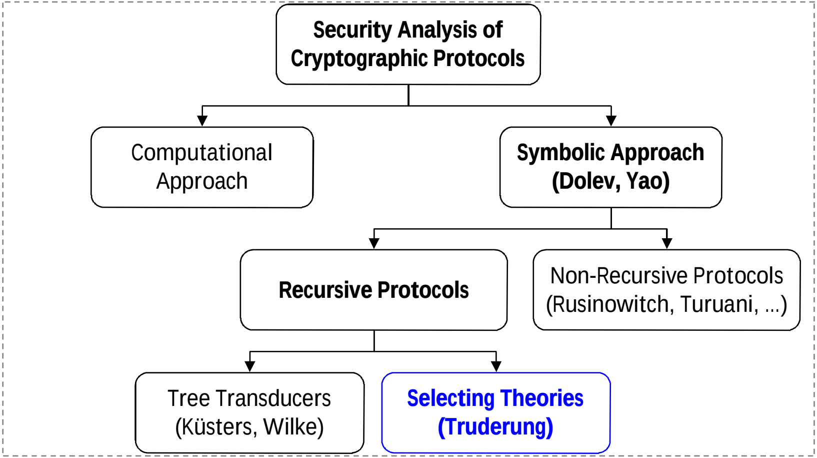
The security of protocols has been studied for a long time in a variety of different ways.



In the Dolev-Yao model, messages are terms over a formal **term algebra**, the **intruder** controls the network and can manipulate messages, but is not able to break encryption or hashing algorithms.

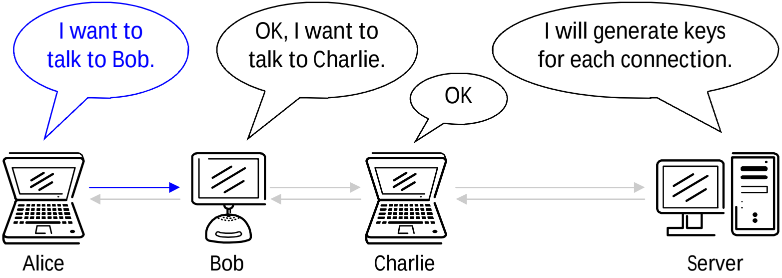


Most results cover non-recursive protocols (and frankly, most protocols are non-recursive). We focus on **recursive protocols**, e. g., the Recursive Authentication Protocol.



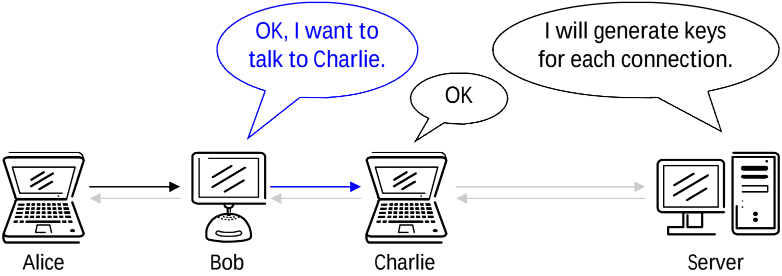
We extend Truderung's model of **Selecting Theories** which allows automatically **deciding security**.

1. $A \rightarrow B$: $m_1 = \text{hash}_{K_A}(A, B, N_A, \square)$



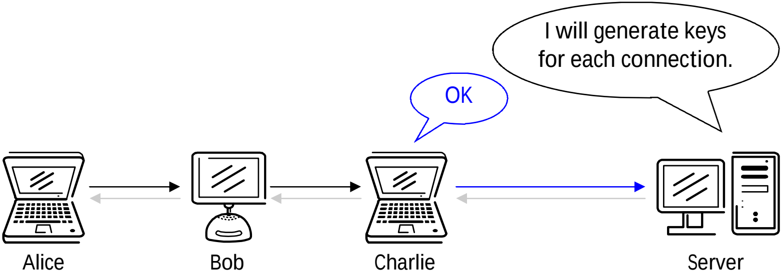
Alice sends Bob the initial request for the Recursive Authentication Protocol.

1. $A \rightarrow B$: $m_1 = \text{hash}_{K_A}(A, B, N_A, \square)$
2. $B \rightarrow C$: $m_2 = \text{hash}_{K_B}(B, C, N_B, m_1)$



Bob includes Alice's message in his own request.

1. $A \to B$: $m_1 = \text{hash}_{K_A}(A, B, N_A, \square)$
2. $B \to C$: $m_2 = \text{hash}_{K_B}(B, C, N_B, m_1)$
3. $C \to S$: $m_3 = \text{hash}_{K_C}(C, S, N_C, m_2)$



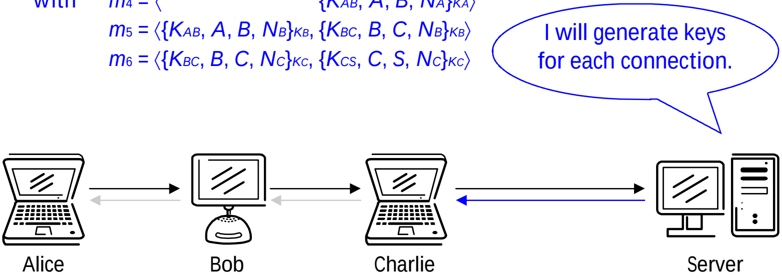
Charlie sends the nested requests to the server.

1. $A \to B$: $m_1 = \mathrm{hash}_{K_A}(A, B, N_A, \square)$
2. $B \to C$: $m_2 = \mathrm{hash}_{K_B}(B, C, N_B, m_1)$
3. $C \to S$: $m_3 = \mathrm{hash}_{K_C}(C, S, N_C, m_2)$
4. $S \to C$: $\langle m_4, m_5, m_6 \rangle$

with $m_4 = \langle \{K_{AB}, A, B, N_A\}_{K_A} \rangle$

$m_5 = \langle \{K_{AB}, A, B, N_B\}_{K_B}, \{K_{BC}, B, C, N_B\}_{K_B} \rangle$

$m_6 = \langle \{K_{BC}, B, C, N_C\}_{K_C}, \{K_{CS}, C, S, N_C\}_{K_C} \rangle$



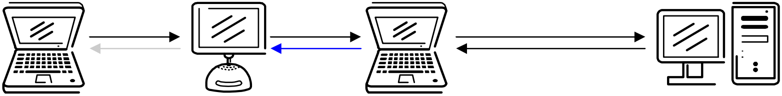
The server generates the session keys K_{AB} and K_{BC} (as well as K_{CS}) and sends the three certificates to Charlie.

1. $A \to B$: $m_1 = \text{hash}_{K_A}(A, B, N_A, \square)$
2. $B \to C$: $m_2 = \text{hash}_{K_B}(B, C, N_B, m_1)$
3. $C \to S$: $m_3 = \text{hash}_{K_C}(C, S, N_C, m_2)$
4. $S \to C$: $\langle m_4, m_5, m_6 \rangle$

   with $m_4 = \langle \{K_{AB}, A, B, N_A\}_{K_A} \rangle$

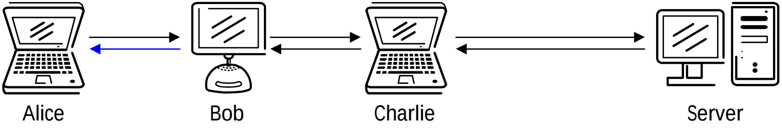
   $m_5 = \langle \{K_{AB}, A, B, N_B\}_{K_B}, \{K_{BC}, B, C, N_B\}_{K_B} \rangle$

   $m_6 = \langle \{K_{BC}, B, C, N_C\}_{K_C}, \{K_{CS}, C, S, N_C\}_{K_C} \rangle$
5. $C \to B$: $\langle m_4, m_5 \rangle$



Charlie forwards Bob's and Alice's certificates to Bob.

1. $A \to B$: $m_1 = \text{hash}_{K_A}(A, B, N_A, \square)$
2. $B \to C$: $m_2 = \text{hash}_{K_B}(B, C, N_B, m_1)$
3. $C \to S$: $m_3 = \text{hash}_{K_C}(C, S, N_C, m_2)$
4. $S \to C$: $\langle m_4, m_5, m_6 \rangle$
   with $m_4 = \langle \{K_{AB}, A, B, N_A\}_{K_A} \rangle$
   $m_5 = \langle \{K_{AB}, A, B, N_B\}_{K_B}, \{K_{BC}, B, C, N_B\}_{K_B} \rangle$
   $m_6 = \langle \{K_{BC}, B, C, N_C\}_{K_C}, \{K_{CS}, C, S, N_C\}_{K_C} \rangle$
5. $C \to B$: $\langle m_4, m_5 \rangle$
6. $B \to A$: $\langle m_4 \rangle$



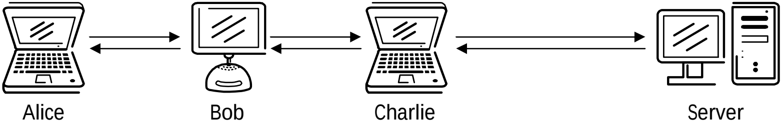
Bob forwards Alices's certificate to her.

1. $A \rightarrow B$: $m_1 = \text{hash}_{K_A}(A, B, N_A, \square)$
2. $B \rightarrow C$: $m_2 = \text{hash}_{K_B}(B, C, N_B, m_1)$
3. $C \rightarrow S$: $m_3 = \text{hash}_{K_C}(C, S, N_C, m_2)$
4. $S \rightarrow C$: $\langle m_4, m_5, m_6 \rangle$

   with $m_4 = \langle \{\mathbf{K_{AB}}, A, B, N_A\}_{K_A} \rangle$

   $m_5 = \langle \{\mathbf{K_{AB}}, A, B, N_B\}_{K_B}, \{\mathbf{K_{BC}}, B, C, N_B\}_{K_B} \rangle$

   $m_6 = \langle \{\mathbf{K_{BC}}, B, C, N_C\}_{K_C}, \{\mathbf{K_{CS}}, C, S, N_C\}_{K_C} \rangle$
5. $C \rightarrow B$: $\langle m_4, m_5 \rangle$
6. $B \rightarrow A$: $\langle m_4 \rangle$



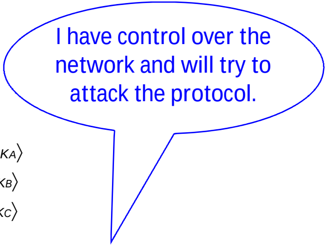
The server has to generate session keys, but the protocol defines no restriction on the number of nested requests, i. e., the server may have to generate an arbitrary number of keys.

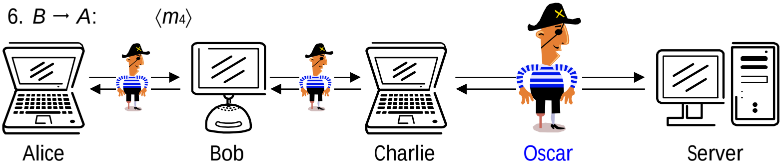
1. $A \rightarrow B$: $m_1 = \text{hash}_{K_A}(A, B, N_A, \square)$
2. $B \rightarrow C$: $m_2 = \text{hash}_{K_B}(B, C, N_B, m_1)$
3. $C \rightarrow S$: $m_3 = \text{hash}_{K_C}(C, S, N_C, m_2)$
4. $S \rightarrow C$: $\langle m_4, m_5, m_6 \rangle$

   with $m_4 = \langle \quad \{K_{AB}, A, B, N_A\}_{K_A} \rangle$

   $m_5 = \langle \{K_{AB}, A, B, N_B\}_{K_B}, \{K_{BC}, B, C, N_B\}_{K_B} \rangle$

   $m_6 = \langle \{K_{BC}, B, C, N_C\}_{K_C}, \{K_{CS}, C, S, N_C\}_{K_C} \rangle$
5. $C \rightarrow B$: $\langle m_4, m_5 \rangle$
6. $B \rightarrow A$: $\langle m_4 \rangle$





A Dolev-Yao style intruder can control all the messages in the network and may try to exploit a flaw in the protocol design.

The Protocol Model: Basic Model (Tomasz Truderung)

A principal consists of a sequence of **receive-send actions** and some rules for **recursive computation**.

The Protocol Model: Basic Model (Tomasz Truderung)

A principal consists of a sequence of **receive-send actions** and some rules for **recursive computation**.

Receive-Send Actions

modeled by
rewrite rules

$$t \to r(s)$$

with terms t and s and a predicate symbol r

The Protocol Model: Basic Model (Tomasz Truderung)

A principal consists of a sequence of **receive-send actions** and some rules for **recursive computation**.

Receive-Send Actions

modeled by
rewrite rules

$$t \rightarrow r(s)$$

with terms t and s and a predicate symbol r

Recursive Computations

modeled by a "**selecting theory**" containing clauses of the form

push clauses	$r(t) \rightarrow r'(x)$
send clauses	$r(t) \rightarrow \mathtt{I}(s)$

with terms t and s and a variable x

The Protocol Model: Basic Model (Tomasz Truderung)

A principal consists of a sequence of **receive-send actions** and some rules for **recursive computation**.

Receive-Send Actions

modeled by **rewrite rules**

$$t \to r(s)$$

with terms t and s and a predicate symbol r

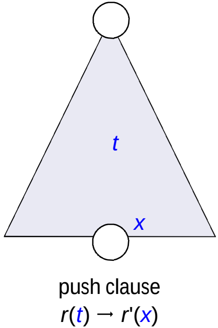
Recursive Computations

modeled by a "**selecting theory**" containing clauses of the form

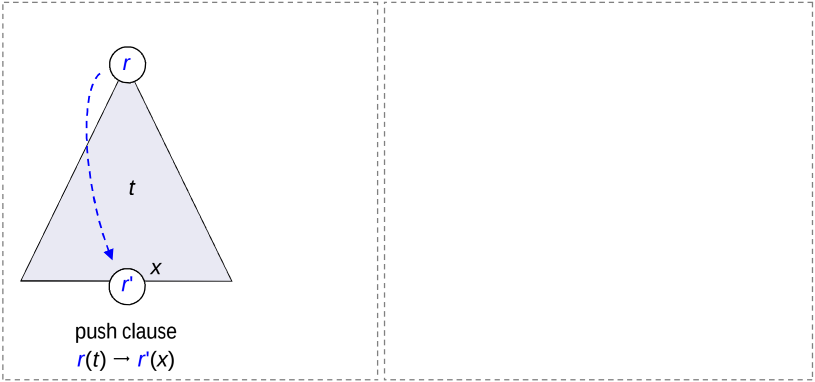
push clauses	$r(t) \to r'(x)$
send clauses	$r(t) \to \mathtt{I}(s)$

with terms t and s and a variable x

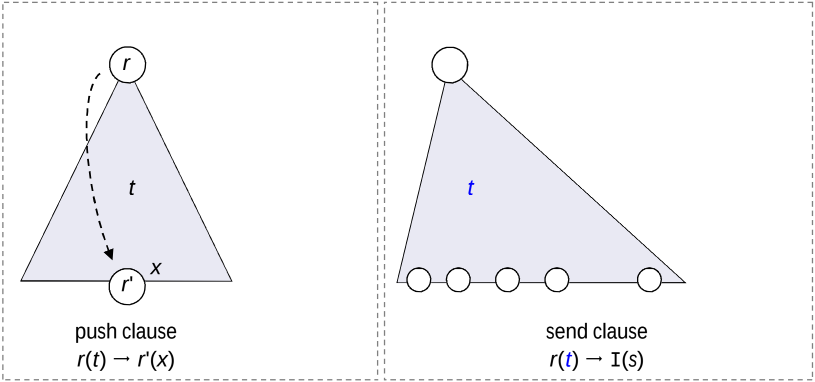
Push clauses model recursive computations; **Send clauses** send terms to the network, adding them to the intruder's knowledge.



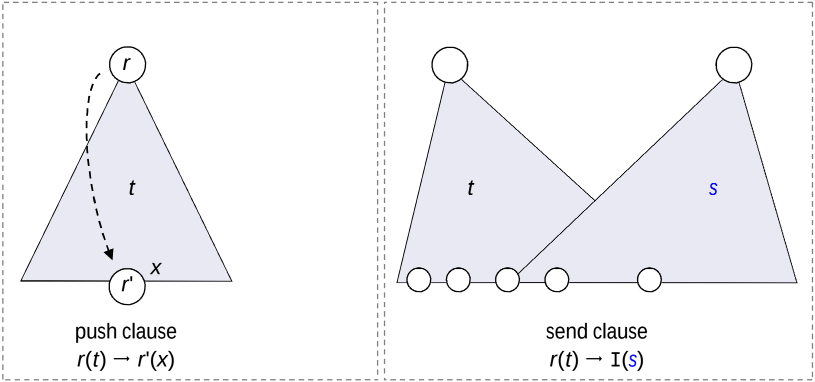
For a push clause $r(t) \to r'(x)$, consider a term t and a variable x occurring in t.



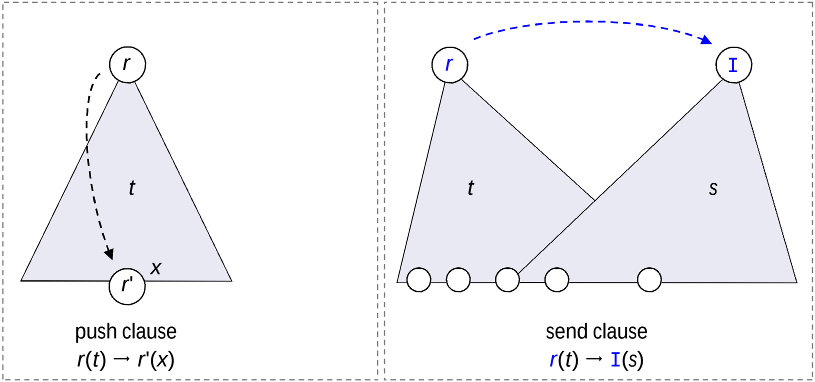
Let t be annotated with the predicate symbol r, our push clause will then annotate x with the predicate symbol r'.



For a send clause $r(t) \rightarrow \mathtt{I}(s)$, take a term t.



Then s can be any term with $\mathrm{Var}(s) \subseteq \mathrm{Var}(t)$.



Let t be annotated with the predicate symbol r, our clause will then annotate s with the predicate symbol $\mathtt{I}$, sending the term s to the network, i. e., adding the term to the intruder's knowledge.

Our Model: Extension for Nonce Generation

To model nonces and key generation, . . .

Our Model: Extension for Nonce Generation

To model nonces and key generation, ...

1. ... we extend the finite signature by an infinite set of constants called **anonymous constants**, and

Our Model: Extension for Nonce Generation

To model nonces and key generation, ...

1. ... we extend the finite signature by an infinite set of constants called **anonymous constants**, and
2. ... we extend the clauses by **register sequences** κ, i. e., a memory for a fixed number of anonymous constants.

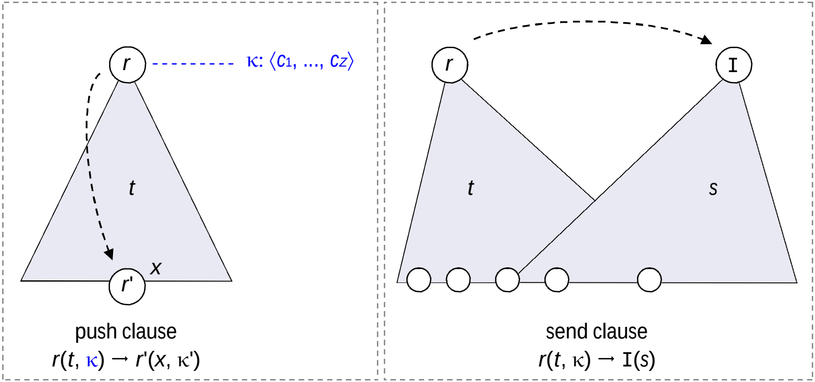
Our Model: Extension for Nonce Generation

To model nonces and key generation, ...

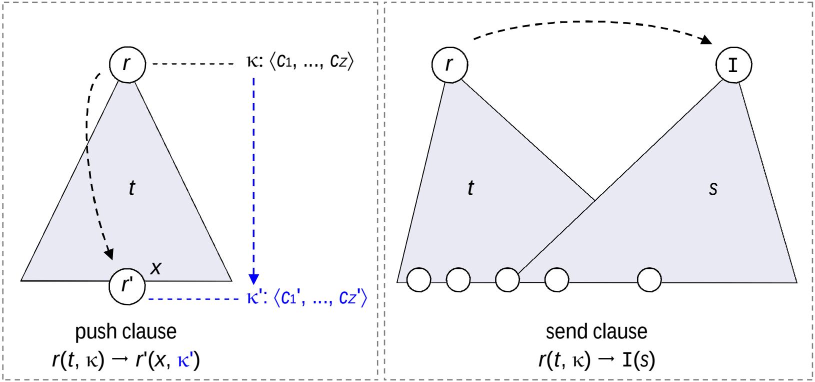
1. ... we extend the finite signature by an infinite set of constants called **anonymous constants**, and
2. ... we extend the clauses by **register sequences** κ, i. e., a memory for a fixed number of anonymous constants.

The extended clauses are basically of the form

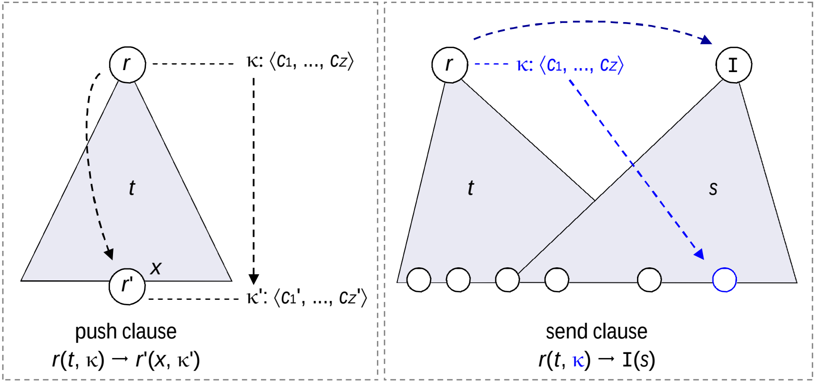
push clauses	$r(t, \kappa) \to r'(x, \kappa'),$
send clauses	$r(t, \kappa) \to \mathtt{I}(s).$



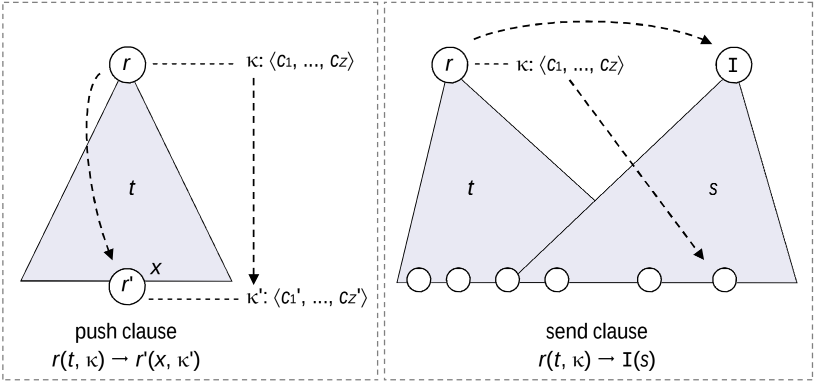
The push clause is now extended and contains κ and κ'. At t, the predicate symbol r has a register sequence κ.



The register sequence κ is transformed to κ' according to the clause.



The send clause is now extended and contains κ. At t, the predicate symbol r has a register sequence κ. The term s can also contain variables from κ.

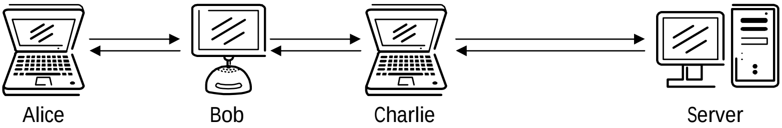


- **Push clauses** model recursive computations.
- **Send clauses** send terms to the network, adding them to the intruder's knowledge.

Modeling the Example Protocol

1. $A \rightarrow B$: $m_1 = \text{hash}_{K_A}(A, B, N_A, \square)$
2. $B \rightarrow C$: $m_2 = \text{hash}_{K_B}(B, C, N_B, m_1)$
3. $C \rightarrow S$: $m_3 = \text{hash}_{K_C}(C, S, N_C, m_2)$
4. $S \rightarrow C$: $\langle m_4, m_5, m_6 \rangle$

   with
   $m_4 = \langle \quad \{\mathbf{K_{AB}}, A, B, N_A\}_{K_A} \rangle$
   $m_5 = \langle \{\mathbf{K_{AB}}, A, B, N_B\}_{K_B}, \{\mathbf{K_{BC}}, B, C, N_B\}_{K_B} \rangle$
   $m_6 = \langle \{\mathbf{K_{BC}}, B, C, N_C\}_{K_C}, \{\mathbf{K_{CS}}, C, S, N_C\}_{K_C} \rangle$
5. $C \rightarrow B$: $\langle m_4, m_5 \rangle$
6. $B \rightarrow A$: $\langle m_4 \rangle$



Modeling the Example Protocol

Upon receiving the term t, the server ...

Modeling the Example Protocol

Upon receiving the term t, the server …

1. …applies a rewrite rule $x \to r(x, \kappa)$ where κ is a register sequence with fresh anonymous constants,

Modeling the Example Protocol

Upon receiving the term t, the server ...

1. ... applies a rewrite rule $x \rightarrow r(x, \kappa)$ where κ is a register sequence with fresh anonymous constants,
2. ... processes the term $r(t, \kappa)$ using the following selecting theory:

$$r(\mathtt{hash}_{K_n}(P_n, P_{m_2}, x_1, x_2), \langle y_1, y_2 \rangle) \rightarrow r(x_2, \langle y_2, y^\star \rangle),$$

Modeling the Example Protocol

Upon receiving the term t, the server ...

1. ...applies a rewrite rule $x \to r(x, \kappa)$ where κ is a register sequence with fresh anonymous constants,
2. ...processes the term $r(t, \kappa)$ using the following selecting theory:

$$r(\mathtt{hash}_{K_n}(P_n, P_{m_2}, x_1, x_2), \langle y_1, y_2 \rangle) \to r(x_2, \langle y_2, y^\star \rangle),$$

Modeling the Example Protocol

Upon receiving the term t, the server ...

1. ... applies a rewrite rule $x \to r(x, \kappa)$ where κ is a register sequence with fresh anonymous constants,
2. ... processes the term $r(t, \kappa)$ using the following selecting theory:

$$r(\mathtt{hash}_{K_n}(P_n, P_{m_2}, x_1, x_2), \langle y_1, y_2 \rangle) \to r(x_2, \langle y_2, y^\star \rangle),$$

$$r(\mathtt{hash}_{K_n}(P_n, P_{m_2}, x_1, \mathtt{hash}_{K_{m_1}}(P_{m_1}, P_n, x_2, x_3)), \langle y_1, y_2 \rangle) \\ \to \mathtt{I}(\{y_2, P_{m_1}, P_n, x_1\}_{K_n}), \mathtt{I}(\{y_1, P_n, P_{m_2}, x_2\}_{K_n}),$$

Modeling the Example Protocol

Upon receiving the term t, the server ...

1. ...applies a rewrite rule $x \to r(x, \kappa)$ where κ is a register sequence with fresh anonymous constants,
2. ...processes the term $r(t, \kappa)$ using the following selecting theory:

$$r(\mathtt{hash}_{K_n}(P_n, P_{m_2}, x_1, x_2), \langle y_1, y_2 \rangle) \to r(x_2, \langle y_2, y^\star \rangle),$$

$$r(\mathtt{hash}_{K_n}(P_n, P_{m_2}, x_1, \mathtt{hash}_{K_{m_1}}(P_{m_1}, P_n, x_2, x_3)), \langle y_1, y_2 \rangle) \\ \to \mathtt{I}(\{y_2, P_{m_1}, P_n, x_1\}_{K_n}), \mathtt{I}(\{y_1, P_n, P_{m_2}, x_2\}_{K_n}),$$

$$r(\mathtt{hash}_{K_n}(P_n, P_m, x_1, \Box), \langle y_1, y_2 \rangle) \to \mathtt{I}(\{y_1, P_n, P_m, x_1\}_{K_n}).$$

where n, m_1, and m_2 range over the set of principals.

Main Results

The Secrecy Problem

Is there a run of a given protocol such that the intruder is able to access the **secret**, i. e., the special constant “$”?

Main Results

The Secrecy Problem

Is there a run of a given protocol such that the intruder is able to access the **secret**, i. e., the special constant “$”?

Decidability Result

The secrecy problem for protocols using anonymous constants is **decidable** in nondeterministic double exponential time.

Main Results

The Secrecy Problem

Is there a run of a given protocol such that the intruder is able to access the **secret**, i. e., the special constant “$”?

Decidability Result

The secrecy problem for protocols using anonymous constants is **decidable** in nondeterministic double exponential time.

Undecidability Result

The secrecy problem is **undecidable** for protocols without anonymous constants, but with **non-flat terms** on the left-hand side of push clauses.

The DAG of an Attack (ADAG)

The technical heart of the paper is ...

1. ... the notion of a DAG of an Attack (**ADAG**), a graph structure that **encodes an attack** on a recursive protocol, i. e., is an encoding of one concrete run of the protocol, and

The DAG of an Attack (ADAG)

The technical heart of the paper is . . .

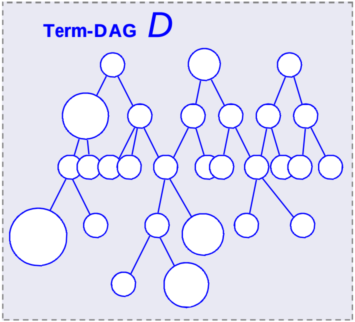
1. . . . the notion of a DAG of an Attack (**ADAG**), a graph structure that **encodes an attack** on a recursive protocol, i. e., is an encoding of one concrete run of the protocol, and
2. . . . the method to scale down the ADAGs to a **limited size**, allowing us to nondeterministically decide the security problem.

The DAG of an Attack (ADAG)

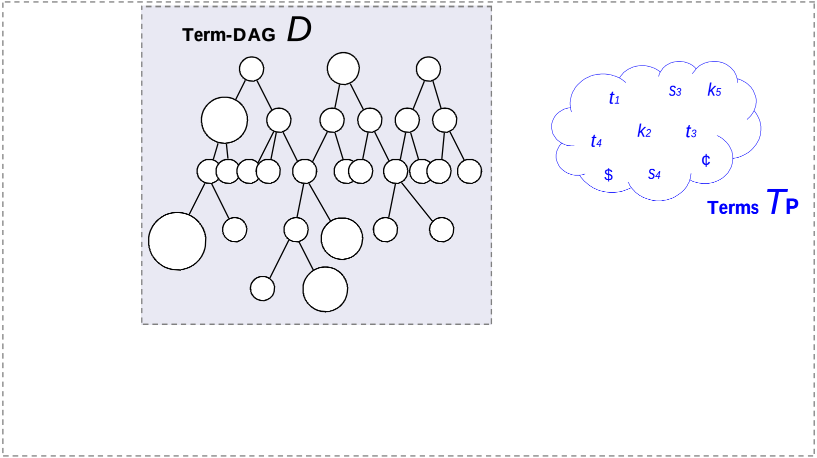
The technical heart of the paper is ...

1. ...the notion of a DAG of an Attack (**ADAG**), a graph structure that **encodes an attack** on a recursive protocol, i. e., is an encoding of one concrete run of the protocol, and
2. ...the method to scale down the ADAGs to a **limited size**, allowing us to nondeterministically decide the security problem.

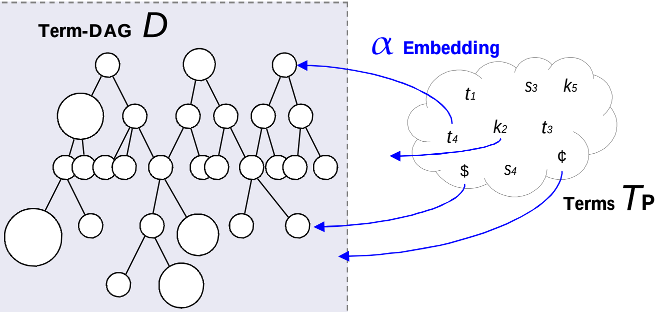
An ADAG is a complex combinatorial structure, its **definition is lengthy** and hideous.



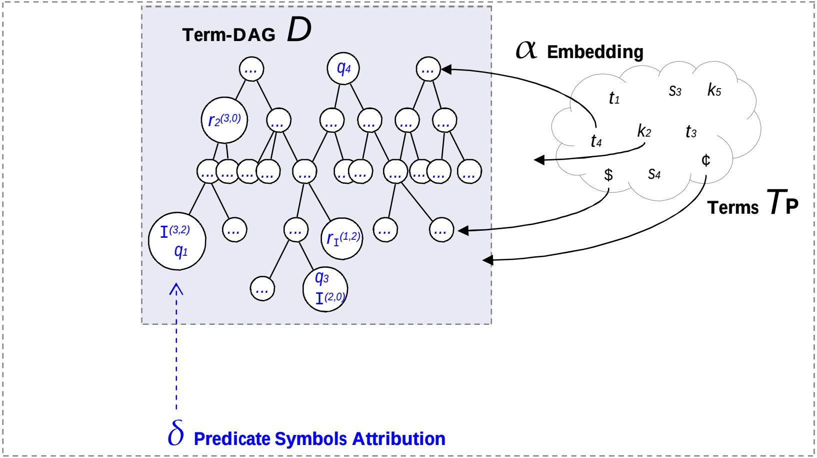
Start with a Term DAG D (actual function symbols and constants of the terms are omitted) containing all terms occurring in the run of a protocol.



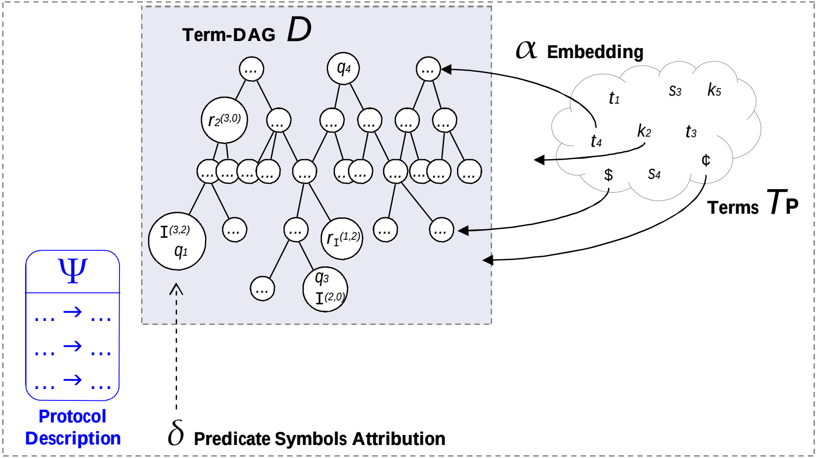
Take the set of terms T_P containing t, s of the receive-send steps, the keys k and the constants for the intruder's initial knowledge and the secret $\$$.



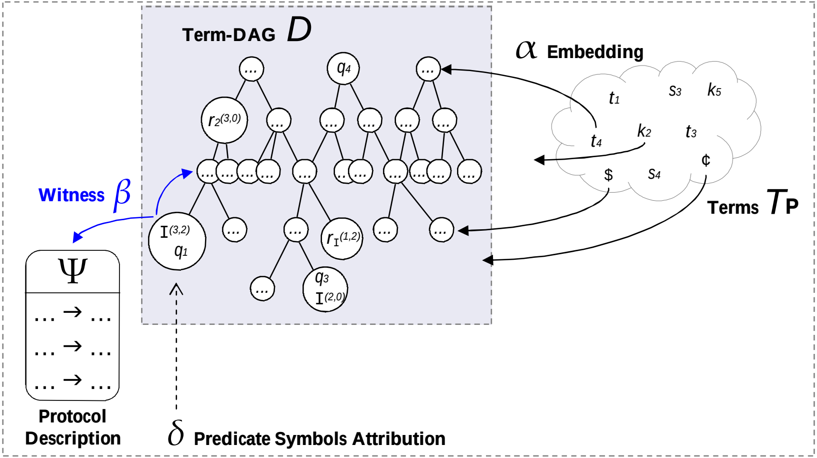
Embed the terms of T_P in the Term DAG, i. e., each term is represented by a fixed node and its descendants.



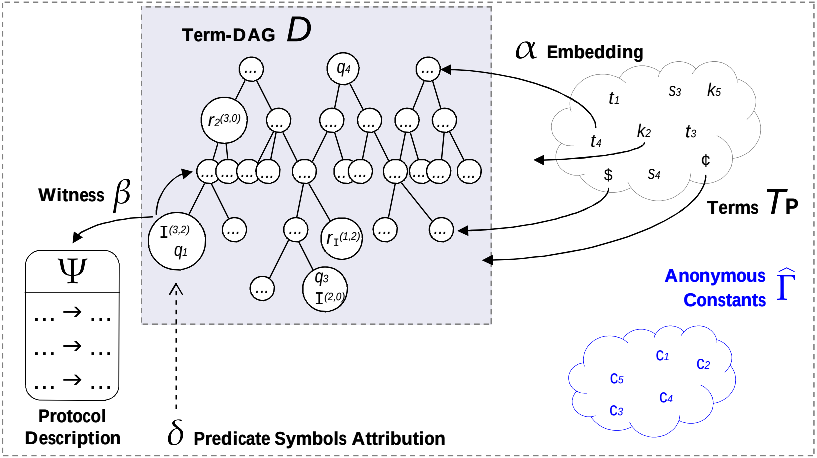
Label the nodes with the predicate symbols occurring in the run of the protocol, i. e., if the Horn fact $\mathtt{I}^{(3,2)}(k_5)$ occurs in the run, label the node corresponding to k_5 with $\mathtt{I}^{(3,2)}$.



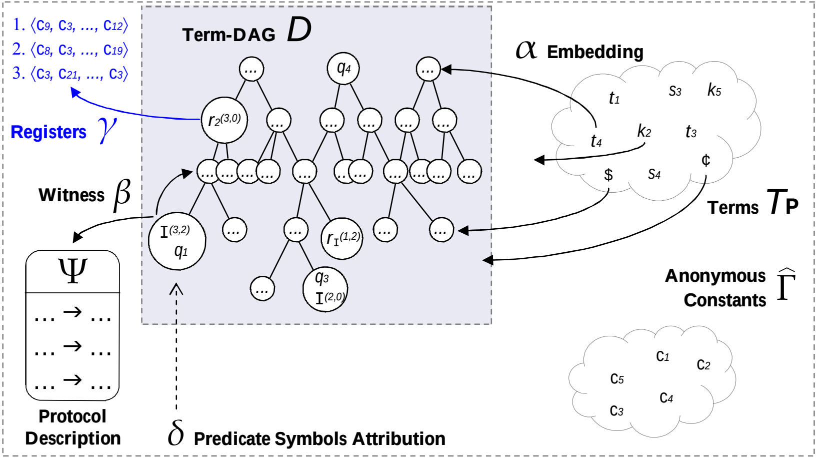
Take the protocol description, i. e., a special merge of the protocol's selecting theory and the intruder's theory.



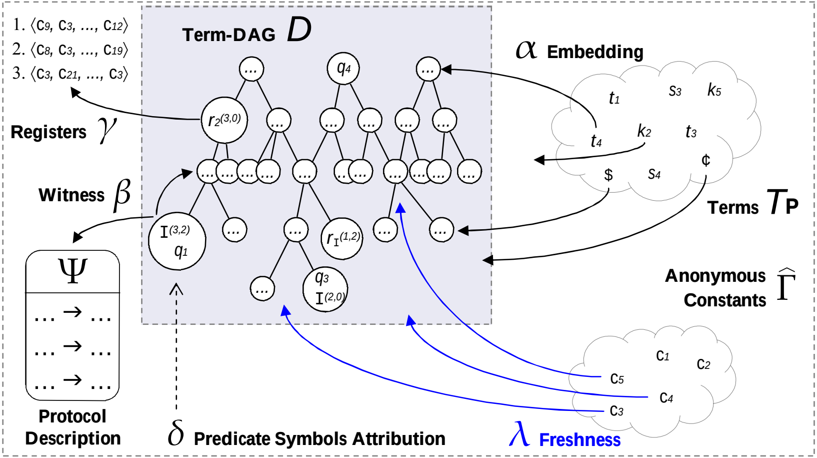
For each predicate symbol r, a function β witnesses the corresponding clause and the prerequisites for applying that clause, i. e., a node, a predicate symbol, and a register sequence?



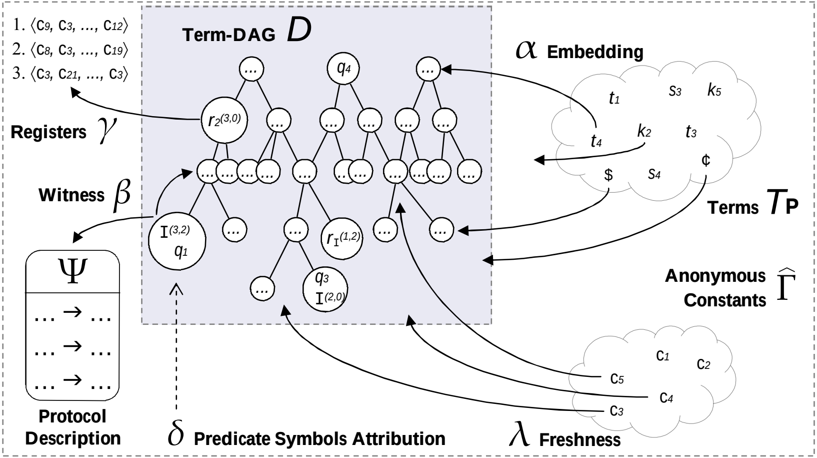
Take the set of anonymous constants. For each concrete run, we only need a finite subset $\hat{\Gamma}$ of Γ.



Each predicate symbol at each node can have multiple register sequences containing a fixed number of anonymous constants.



The freshness of anonymous constants is guaranteed by a function which maps each constant to the location where it is generated.



This is the basic structure of an ADAG $\mathcal{D} = (D, \Psi, \hat{\Gamma}, \alpha, \beta, \gamma, \delta, \lambda)$, which is accompanied by a set of conditions.

Conclusion

- We have extended Truderung's model to model **nonce and key generation**.

Conclusion

- We have extended Truderung's model to model **nonce and key generation**.
- This was done by adding a **infinite set of constants** to the finite signature and by extending the clauses of the selecting theory with a **memory** for a fixed number of constants.

Conclusion

- We have extended Truderung's model to model **nonce and key generation**.
- This was done by adding a **infinite set of constants** to the finite signature and by extending the clauses of the selecting theory with a **memory** for a fixed number of constants.
- In this setting secrecy is **decidable** as long as we do not allow non-linear terms in the push clauses.

Conclusion

- We have extended Truderung's model to model **nonce and key generation**.
- This was done by adding a **infinite set of constants** to the finite signature and by extending the clauses of the selecting theory with a **memory** for a fixed number of constants.
- In this setting secrecy is **decidable** as long as we do not allow non-linear terms in the push clauses.
- The exact modeling of **hash values** in our paper leads to a technical problem we just discovered, which we cannot resolve yet. This doesn't invalidate the example as we can express it in a slightly modified protocol model.

Related and Future Work

- The selecting theory model has, e. g., been **extended** by Küsters and Truderung to allow the modeling of the **XOR** operator.

Related and Future Work

- The selecting theory model has, e. g., been **extended** by Küsters and Truderung to allow the modeling of the **XOR** operator.
- **Future work** could include:
  - Extend the model with Diffie-Hellman Exponentiation.

Related and Future Work

- The selecting theory model has, e. g., been **extended** by Küsters and Truderung to allow the modeling of the **XOR** operator.
- **Future work** could include:
  - Extend the model with Diffie-Hellman Exponentiation.
  - Although the model itself and the usage of selecting theories seems to be elegant, the proof is heavy of technical details and can hopefully be improved.

Related and Future Work

- The selecting theory model has, e. g., been **extended** by Küsters and Truderung to allow the modeling of the **XOR** operator.
- **Future work** could include:
  - Extend the model with Diffie-Hellman Exponentiation.
  - Although the model itself and the usage of selecting theories seems to be elegant, the proof is heavy of technical details and can hopefully be improved.
  - As shown by the undecidability result, there is a trade-of between features of the model and decidability than can be explored further.

Thank you for your attention!
Questions?