Rewriting in Protocol Verification

Stéphanie Delaune

Univ Rennes, CNRS, IRISA, France

Monday, June 29th, 2020
Cryptographic protocols everywhere!

Cryptographic protocols

- small programs designed to secure communication (e.g. secrecy, authentication, anonymity, ...)
- use cryptographic primitives (e.g. encryption, signature, ....)

The network is unsecure!
Communications take place over a public network like the Internet.
Cryptographic protocols everywhere!

Cryptographic protocols

- small programs designed to secure communication (e.g. secrecy, authentication, anonymity, . . . )
- use cryptographic primitives (e.g. encryption, signature, . . . . . . )

It becomes more and more important to protect our privacy.

---

Union européenne République Française

e-Voting

NAVI GO

STIF TRANSPORTS PUBLIQUE EN ILE DE FRANCE

Windows Mobile
How cryptographic protocols can be attacked?

Cryptanalysis

- Differential attacks,
- Boomerang attacks,
- Cube attacks,
- ...

How cryptographic protocols can be attacked?

Logical attacks

- can be mounted even assuming perfect cryptography,
  - replay attack, man-in-the-middle attack, ...
- subtle and hard to detect by “eyeballing” the protocol

This is the so-called Dolev-Yao attacker!
How cryptographic protocols can be attacked?

Logical attacks

- can be mounted even assuming perfect cryptography,
  - replay attack, man-in-the-middle attack, . . .
- subtle and hard to detect by “eyeballing” the protocol

Example: An authentication flaw on the Needham Schroeder protocol

\[
\begin{align*}
A &\rightarrow B: \{A, N_A\}_{\text{pub}(B)} \\
B &\rightarrow A: \{N_A, N_B\}_{\text{pub}(A)} \\
A &\rightarrow B: \{N_B\}_{\text{pub}(B)}
\end{align*}
\]

NS protocol (1978)
How cryptographic protocols can be attacked?

Logical attacks

- can be mounted even assuming perfect cryptography,
  → replay attack, man-in-the-middle attack, . . .
- subtle and hard to detect by “eyeballing” the protocol

Example: An authentication flaw on the Needham Schroeder protocol

\[
\begin{align*}
A & \rightarrow B : \{A, N_A\}_{\text{pub}(B)} \\
B & \rightarrow A : \{N_A, N_B\}_{\text{pub}(A)} \\
A & \rightarrow B : \{N_B\}_{\text{pub}(B)}
\end{align*}
\]

NS protocol (1978)

\[
\begin{align*}
A & \rightarrow B : \{A, N_A\}_{\text{pub}(B)} \\
B & \rightarrow A : \{N_A, N_B, B\}_{\text{pub}(A)} \\
A & \rightarrow B : \{N_B\}_{\text{pub}(B)}
\end{align*}
\]

NS-Lowe protocol (1995)
How cryptographic protocols can be attacked?

Logical attacks

► can be mounted even assuming perfect cryptography,
  → replay attack, man-in-the-middle attack, . . .

► subtle and hard to detect by “eyeballing” the protocol

Example: FREAK attack by Barghavan et al. (2015)

A logical flaw that allows a man-in-the-middle attacker to downgrade connections from ’strong’ RSA to ’export grade’ RSA.
How cryptographic protocols can be attacked?

Logical attacks

- can be mounted even assuming perfect cryptography,
  - replay attack, man-in-the-middle attack, 
- subtle and hard to detect by “eyeballing” the protocol

Example: A traceability attack on the BAC protocol (2010)

Security

Defects in e-passports allow real-time tracking

This threat brought to you by RFID

The register - Jan. 2010
Basic Access Control (BAC) protocol

Passport
$(K_E, K_M)$

Reader
$(K_E, K_M)$

Passport
$(N_P, K_P)$

Reader
$(N_R, K_R)$

$K_{seed} = K_P \oplus K_R$

$K_{seed} = K_P \oplus K_R$
Unlinkability/Untraceability

Informally, an observer/attacker can not observe the difference between the two following situations:

1. a situation where the same passport may be used twice (or even more);
2. a situation where each passport is used at most once.
Unlinkability/Untraceability

Informally, an observer/attacker can not observe the difference between the two following situations:

1. a situation where the same passport may be used twice (or even more);
2. a situation where each passport is used at most once.

More formally,

$$\text{!new } ke \text{.new } km \cdot (\text{!P}_{BAC} \mid \text{!R}_{BAC}) \approx \text{!new } ke \text{.new } km \cdot (\text{P}_{BAC} \mid \text{R}_{BAC})$$

↑

many sessions for each passport

only one session for each passport

(we still have to formalize the notion of equivalence)
Some other equivalence-based security properties

Vote privacy
the fact that a particular voter voted in a particular way is not revealed to anyone

Strong secrecy
the fact that an adversary cannot see any difference when the value of the secret changes
→ stronger than the notion of secrecy as non-deducibility.

Guessing attack
the fact that an adversary can not learn the value of passwords even if he knows that they have been choosen in a particular dictionary.
How rewriting and unification theory can help us in protocol verification?
Messages as terms - Back to the BAC protocol

Nonces $n_r$, $n_p$, and keys $k_r$, $k_p$, $k_e$, $k_m$ are modelled using names.

Cryptographic primitives are modelled using function symbols:

- encryption/decryption: $\text{senc}/2$, $\text{sdec}/2$
- concatenation/projections: $\langle \cdot, \cdot \rangle/2$, $\text{proj}_1/1$, $\text{proj}_2/1$
- mac construction: $\text{mac}/2$

\[
\text{sdec}((\text{senc}(x, y), y)) = x \quad \text{proj}_1(\langle x, y \rangle) = x \quad \text{proj}_2(\langle x, y \rangle) = y
\]
Messages as terms - Back to the BAC protocol

Nonces $n_r$, $n_p$, and keys $k_r$, $k_p$, $k_e$, $k_m$ are modelled using names.

Cryptographic primitives are modelled using function symbols:

- encryption/decryption: $\text{senc}/2$, $\text{sdec}/2$
- concatenation/projections: $\langle\cdot,\cdot\rangle/2$, $\text{proj}_1/1$, $\text{proj}_2/1$
- mac construction: $\text{mac}/2$

\[
\text{sddec}(\text{senc}(x,y),y) = x \quad \text{proj}_1(\langle x, y \rangle) = x \quad \text{proj}_2(\langle x, y \rangle) = y
\]

Exclusive-or operator: $\oplus$ of arity 2 and 0 (neutral element)

\[
\begin{align*}
x \oplus (y \oplus z) &= (x \oplus y) \oplus z \\
x \oplus x &= 0 \\
x \oplus y &= y \oplus x \\
x \oplus 0 &= x
\end{align*}
\]
Messages as terms - Back to the BAC protocol

Nonces $n_r$, $n_p$, and keys $k_r$, $k_p$, $k_e$, $k_m$ are modelled using names.

Cryptographic primitives are modelled using function symbols:

- encryption/decryption: $\text{senc}/2$, $\text{sdec}/2$
- concatenation/projections: $\langle , \rangle/2$, $\text{proj}_1/1$, $\text{proj}_2/1$
- mac construction: $\text{mac}/2$

\[
\text{sdec}(\text{senc}(x, y), y) = x \quad \text{proj}_1(\langle x, y \rangle) = x \quad \text{proj}_2(\langle x, y \rangle) = y
\]

Equational theories are useful to model algebraic properties of cryptographic primitives.
Computations as recipes

\textbf{frame} = \text{knowledge of the attacker} = \text{sequence of messages}

\[ \phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \} \]
Computations as recipes

frame = knowledge of the attacker = sequence of messages

\[ \phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \} \]

Example: \( \text{adec}(\text{aenc}(x, \text{pk}(y)), y) \rightarrow x \)

\[ \{ w_1 \triangleright \text{pk}(ska); \ w_2 \triangleright \text{pk}(skb); \ w_3 \triangleright skc; \ w_4 \triangleright \text{aenc}(\langle a, n_a \rangle, \text{pk}(skc)) \}. \]

initial knowledge

1st message of NS
Computations as recipes

frame = knowledge of the attacker = sequence of messages

\[ \phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \} \]

Example: \( \text{adec}(\text{aenc}(x, \text{pk}(y)), y) \rightarrow x \)

\[ \{ w_1 \triangleright \text{pk}(ska); \ w_2 \triangleright \text{pk}(skb); \ w_3 \triangleright skc; \ w_4 \triangleright \text{aenc}((a, n_a), \text{pk}(skc)) \}. \]

Some recipes:

- from his private key \( skc \), the attacker is able to get his public key with \( R = \text{pk}(w_3) \);
- \( R = \text{aenc}(\text{adec}(w_4, w_3), w_2) \) – this is the first step of the man-in-the-middle attack on NS protocol
Computations as recipes

frame = knowledge of the attacker = sequence of messages

\[ \phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \} \]

Example: \( \text{adec}(\text{aenc}(x, \text{pk}(y)), y) \rightarrow x \)

\[ \{ w_1 \triangleright \text{pk}(\text{ska}); \; w_2 \triangleright \text{pk}(\text{skb}); \; w_3 \triangleright \text{skc}; \; w_4 \triangleright \text{aenc}(\langle a, n_a \rangle, \text{pk}(\text{skc})) \} \]

initial knowledge

1st message of NS

Some recipes:

- from his private key \( \text{skc} \), the attacker is able to get his public key with \( R = \text{pk}(w_3) \);
- \( R = \text{aenc}(\text{adec}(w_4, w_3), w_2) \) – this is the first step of the man-in-the-middle attack on NS protocol

Rewriting is useful to express computations performed by the attacker.
Static equivalence

Warm-up

→ this is the so-called passive attacker
The static equivalence problem ($\phi \sim \psi$)

- **Input:** two substitutions (called frames) $\phi$ and $\psi$
  
  \[ \phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \} \quad \psi = \{ w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell \} \]

- **Output:** Can the attacker distinguish the two frames, i.e. does there exist a test $R_1 \equiv R_2$ such that:
  
  \[ R_1\phi =_E R_2\phi \text{ but } R_1\psi \neq _E R_2\psi \] (or the converse).
The static equivalence problem ($\phi \sim \psi$)

- **Input:** two substitutions (called frames) $\phi$ and $\psi$
  
  \[ \phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \}, \quad \psi = \{ w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell \} \]

- **Output:** Can the attacker distinguish the two frames, i.e. does there exist a test $R_1 = R_2$ such that:
  \[ R_1 \phi =_E R_2 \phi \text{ but } R_1 \psi \not= _E R_2 \psi \] (or the converse).

**Example 1:** $\text{adec}(\text{aenc}(x, \text{pk}(y)), y) = x$

- $\phi = \{ w_1 \triangleright \text{pk}(\text{sks}); \ w_2 \triangleright \text{aenc}(\text{yes}, \text{pk}(\text{sks})) \}$; and
- $\psi = \{ w_1 \triangleright \text{pk}(\text{sks}); \ w_2 \triangleright \text{aenc}(\text{no}, \text{pk}(\text{sks})) \}$. 
The static equivalence problem \((\phi \sim \psi)\)

- **Input:** two substitutions (called frames) \(\phi\) and \(\psi\)
  \[
  \phi = \{w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell\} \quad \psi = \{w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell\}
  \]

- **Output:** Can the attacker distinguish the two frames, i.e. does there exist a test \(R_1 \equiv R_2\) such that:
  \[
  R_1\phi =_E R_2\phi \text{ but } R_1\psi \neq_E R_2\psi \text{ (or the converse)}.
  \]

**Example 1:** \(\text{adec}(\text{aenc}(x, \text{pk}(y)), y) = x\)

- \(\phi = \{w_1 \triangleright \text{pk}(\text{sks}); \ w_2 \triangleright \text{aenc}(\text{yes}, \text{pk}(\text{sks}))\}\); and
- \(\psi = \{w_1 \triangleright \text{pk}(\text{sks}); \ w_2 \triangleright \text{aenc}(\text{no}, \text{pk}(\text{sks}))\}\).

\[\rightarrow\] They are not in static equivalence: \(\text{aenc}(\text{yes}, w_1) \neq w_2\).
The static equivalence problem ($\phi \sim \psi$)

- **Input:** two substitutions (called frames) $\phi$ and $\psi$
  \[\phi = \{w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell\} \quad \psi = \{w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell\}\]

- **Output:** Can the attacker distinguish the two frames, i.e. does there exist a test $R_1 \neq R_2$ such that:
  \[R_1 \phi =_E R_2 \phi \text{ but } R_1 \psi \neq_E R_2 \psi\] (or the converse).

**Example 2:** (randomized encryption)

- $\phi = \{w_1 \triangleright \text{pk}(\text{sks}); w_2 \triangleright \text{aenc}(\langle \text{yes}, r \rangle, \text{pk}(\text{sks}))\}$; and
- $\psi = \{w_1 \triangleright \text{pk}(\text{sks}); w_2 \triangleright \text{aenc}(\langle \text{no}, r \rangle, \text{pk}(\text{sks}))\}$. 
The static equivalence problem \( (\phi \sim \psi) \)

- **Input:** two substitutions (called frames) \( \phi \) and \( \psi \)
  \[
  \phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \} \quad \psi = \{ w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell \}
  \]

- **Output:** Can the attacker distinguish the two frames, i.e. does there exist a test \( R_1 \overset{?}{=} R_2 \) such that:
  \[
  R_1 \phi =_E R_2 \phi \text{ but } R_1 \psi \neq_\neq E R_2 \psi \text{ (or the converse)}.
  \]

**Example 2:** (randomized encryption)

- \( \phi = \{ w_1 \triangleright \text{pk}(\text{sks}); \ w_2 \triangleright \text{aenc}(\langle \text{yes}, r \rangle, \text{pk}(\text{sks})) \} \); and

- \( \psi = \{ w_1 \triangleright \text{pk}(\text{sks}); \ w_2 \triangleright \text{aenc}(\langle \text{no}, r \rangle, \text{pk}(\text{sks})) \} \).

\[\rightarrow\] They are in static equivalence.
## Static equivalence – some existing results

<table>
<thead>
<tr>
<th>Theory E</th>
<th>Deduction</th>
<th>Static Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>subterm convergent</td>
<td>PTIME</td>
<td></td>
</tr>
<tr>
<td>blind signature,</td>
<td>decidable</td>
<td>[Abadi &amp; Cortier, 06]</td>
</tr>
<tr>
<td>homo. encryption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACU</td>
<td>NP-complete</td>
<td>PTIME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Cortier &amp; D., 10]</td>
</tr>
<tr>
<td>ACUN/AG</td>
<td>PTIME</td>
<td>PTIME</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Chevalier et al, 03]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Cortier &amp; D., 10]</td>
</tr>
<tr>
<td>ACUNh/AGh</td>
<td>PTIME</td>
<td>decidable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[D., 06]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Cortier &amp; D., 10]</td>
</tr>
</tbody>
</table>
The case of monoidal theories, e.g. XOR, AG, ...

Getting some inspiration from existing results and proofs in unification theory, e.g. [Nutt, 90], and [Baader & Schulz, 96], we associate a semi-ring $S_E$ to a monoidal theory $E$:

$\mathbb{Z}/2\mathbb{Z}$ for ACUN, $\mathbb{Z}$ for AG, $\mathbb{Z}/2\mathbb{Z}[X]$ for ACUNh, ...
The case of monoidal theories, e.g. XOR, AG, ...

Getting some inspiration from existing results and proofs in unification theory, e.g. [Nutt, 90], and [Baader & Schulz, 96], we associate a semi-ring $S_E$ to a monoidal theory $E$:

$\mathbb{Z}/2\mathbb{Z}$ for ACUN, $\mathbb{Z}$ for AG, $\mathbb{Z}/2\mathbb{Z}[X]$ for ACUNh, ...

Theorem [Cortier & D., 10]

Static equivalence in $E$ is reducible in PTIME to

Input  Two matrices $A_1$ and $A_2$ over $S_E$

Output  Does the following equality holds?

$$\{(X, Y) \in S_E^\ell \times S_E^\ell \mid X \cdot A_1 = Y \cdot A_1\}$$

$$=\$$

$$\{(X, Y) \in S_E^\ell \times S_E^\ell \mid X \cdot A_2 = Y \cdot A_2\}$$
Reduction on an example – $E = AG$ and $S_E = \mathbb{Z}$

$\rightarrow$ $a$ and $b$ two constants (e.g. nonces)

$\triangleright \phi_1 = \{ w_1 \triangleright a + b + b; \ w_2 \triangleright b; \ w_3 \triangleright a + a + a \}$
Reduction on an example – \( E = AG \) and \( S_E = \mathbb{Z} \)

\[ \rightarrow \ a \text{ and } b \text{ two constants (e.g. nonces)} \]

\[ \phi_1 = \{w_1 \triangleright a + b + b; \ w_2 \triangleright b; \ w_3 \triangleright a + a + a\} \]

Question: Is \( a \) deducible from \( \phi_1 \), i.e. does there exists \( X \) such that \( X \cdot A_1 = V \) ?

\[
A_1 = \begin{pmatrix}
1 & 2 \\
0 & 1 \\
3 & 0
\end{pmatrix} \quad \text{and} \quad V = \begin{pmatrix} 1 & 0 \end{pmatrix}
\]
Reduction on an example – $E = AG$ and $S_E = \mathbb{Z}$

$\rightarrow a$ and $b$ two constants (e.g. nonces)

$\triangleright \phi_1 = \{w_1 \triangleright a + b + b; \ w_2 \triangleright b; \ w_3 \triangleright a + a + a\}$

Question: Is $a$ deducible from $\phi_1$, i.e does there exists $X$ such that $X \cdot A_1 = V$ ?

$$A_1 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \\ 3 & 0 \end{pmatrix} \quad \text{and} \quad V = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Answer: Yes! $X = (1 \quad -2 \quad 0)$.

$\rightarrow$ this corresponds to recipe $R = w_1 + -(w_2) + -(w_2)$.

Indeed, we have that:

$$R\phi_1 \downarrow = a + b + b + -(b) + -(b)$$

$$= a$$
Reduction on an example – $E = AG$ and $S_E = \mathbb{Z}$

$\rightarrow a$ and $b$ two constants (e.g. nonces)

$\phi_1 = \{w_1 \triangleright a + b + b; \ w_2 \triangleright b; \ w_3 \triangleright a + a + a\}$

$\phi_2 = \{w_1 \triangleright a + b + b; \ w_2 \triangleright b + b; \ w_3 \triangleright 3a - 6b\}$

Question: Is $\phi_1 \sim \phi_2$, i.e. do $\{(X, Y) \mid X \cdot A_1 = Y \cdot A_1\}$ and $\{(X, Y) \mid X \cdot A_2 = Y \cdot A_2\}$ have the same set of solutions?

$A_1 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \\ 3 & 0 \end{pmatrix}$ and $A_2 = \begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 3 & -6 \end{pmatrix}$
Reduction on an example – $E = AG$ and $S_E = \mathbb{Z}$

$\rightarrow$ $a$ and $b$ two constants (e.g. nonces)

$\triangleright \phi_1 = \{w_1 \triangleright a + b + b; \ w_2 \triangleright b; \ w_3 \triangleright a + a + a\}$

$\triangleright \phi_2 = \{w_1 \triangleright a + b + b; \ w_2 \triangleright b + b; \ w_3 \triangleright 3a - 6b\}$

Question: Is $\phi_1 \sim \phi_2$, i.e. do $\{(X, Y) \mid X \cdot A_1 = Y \cdot A_1\}$ and $\{(X, Y) \mid X \cdot A_2 = Y \cdot A_2\}$ have the same set of solutions?

$$A_1 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \\ 3 & 0 \end{pmatrix} \quad \text{and} \quad A_2 = \begin{pmatrix} 1 & 2 \\ 0 & 2 \\ 3 & -6 \end{pmatrix}$$

Answer: Yes! $\phi_1 \sim_E \phi_2$
Reduction on an example – $E = AG$ and $S_E = \mathbb{Z}$

$\rightarrow a$ and $b$ two constants (e.g. nonces)

$\phi_1 = \{w_1 \triangleright a + b + b; w_2 \triangleright b; w_3 \triangleright a + a + a\}$

$\phi_2 = \{w_1 \triangleright a + b + b; w_2 \triangleright b + b; w_3 \triangleright 3a - 6b\}$

Question: Is $\phi_1 \sim \phi_2$, i.e. do $\{(X, Y) \mid X \cdot A_1 = Y \cdot A_1\}$ and $\{(X, Y) \mid X \cdot A_2 = Y \cdot A_2\}$ have the same set of solutions?

Answer: Yes! $\phi_1 \sim_E \phi_2$

In both cases (deduction and equivalence), we only have to consider sets of linear equations with coefficient in $\mathbb{Z}$. 
Combination result

If deduction and static equivalence are decidable for two disjoint theories $E_1$ and $E_2$ then they are also decidable for $E_1 \cup E_2$.

[Cortier & D., 10]

Example: this allows one to combine encryption and exclusive-or.
Combination result

If deduction and static equivalence are decidable for two disjoint theories $E_1$ and $E_2$ then they are also decidable for $E_1 \cup E_2$.

[Corrier & D., 10]

Example: this allows one to combine encryption and exclusive-or

Proof (sketch): given $\phi$ and $\psi$ built on $E_1 \cup E_2$

1. saturate both frames simultaneously with deducible subterms (notion of alien subterms); $\rightarrow$ this leads to $\phi_+$ and $\psi_+$

2. abstract subterms coming from $E_i$ ($i = 1, 2$) $\rightarrow$ this leads to $\phi^+_i$ and $\psi^+_i$

3. check whether: $\overline{\phi^+_i} \approx_{E_2-i} \overline{\psi^+_i}$ (with $i = 1, 2$)

$\rightarrow$ inspiration from [Schmidt-Schauß, 89; Baader & Schluž, 96]
Caution!

One should never underestimate the attacker!

The attacker can listen to the communication but also:

- **intercept** the messages that are sent by the participants,
- **build new messages** according to his deduction capabilities, and
- **send** messages on the communication network.

→ this is the co-called *active attacker*
from frames to constraint systems

A constraint system $\mathcal{C}$ is a triple $(\phi; \mathcal{D}; \mathcal{E})$ where:

1. $\phi = \{ w_1 \triangleright v_1; \ldots; w_\ell \triangleright v_\ell \}$ is an open frame;
2. $\mathcal{D}$ is a set of deducibility constraints: $X \triangleright x$ with $ar(X) < \ell$
3. $\mathcal{E}$ is a unification problem (modulo $E$)
   $+$ some conditions (e.g. monotonicity, origination).
A constraint system $\mathcal{C}$ is a triple $(\phi; \mathcal{D}; \mathcal{E})$ where:

1. $\phi = \{w_1 \triangleright v_1; \ldots; w_\ell \triangleright v_\ell\}$ is an open frame;
2. $\mathcal{D}$ is a set of deducibility constraints: $X \triangleright x$ with $ar(X) < \ell$
3. $\mathcal{E}$ is a unification problem (modulo $\mathcal{E}$) + some conditions (e.g. monotonicity, origination).

Example: a simple challenge-response protocol

\[ A \rightarrow B : \ \{\text{req, } n\}_k \]
\[ B \rightarrow A : \ \{\text{rep, hash}(n)\}_k \]
from frames to constraint systems

A constraint system $C$ is a triple $(\phi; \mathcal{D}; \mathcal{E})$ where:

1. $\phi = \{w_1 \triangleright v_1; \ldots; w_\ell \triangleright v_\ell\}$ is an open frame;
2. $\mathcal{D}$ is a set of deducibility constraints: $X \triangleright x$ with $ar(X) < \ell$
3. $\mathcal{E}$ is a unification problem (modulo $E$)
   + some conditions (e.g. monotonicity, origination).

Example: a simple challenge-response protocol

$$
A \rightarrow B : \{\text{req, n}\}_k \\
B \rightarrow A : \{\text{rep, hash(n)}\}_k
$$

1. $\{w_1 \triangleright \{\langle \text{req, n}\rangle\}_k; w_2 \triangleright \{\langle \text{rep, hash(proj}_2(\text{sdec(x, k}))\rangle\}_k\}$
2. $X \triangleright x$ with $ar(X) = 1$
3. $\text{proj}_1(\text{sdec(x, k)}) = \text{req}$
from frames to constraint systems

A constraint system $C$ is a triple $(\phi; D; E)$ where:

1. $\phi = \{ w_1 \triangleright v_1; \ldots; w_\ell \triangleright v_\ell \}$ is an open frame;
2. $D$ is a set of deducibility constraints: $X \triangleright x$ with $ar(X) < \ell$
3. $E$ is a unification problem (modulo $E$)
   + some conditions (e.g. monotonicity, origination).

Example: a simple challenge-response protocol

\[
\begin{align*}
A \rightarrow B : & \quad \{\text{req, n}\}_k \\
B \rightarrow A : & \quad \{\text{rep, hash(n)}\}_k \\
\end{align*}
\]

1. $\{ w_1 \triangleright \{\langle \text{req, n}\rangle\}_k; \ w_2 \triangleright \{\langle \text{rep, hash(proj}_2(\text{sdec(x, k))})\rangle\}_k \}$
2. $X \triangleright x$ with $ar(X) = 1$
3. $\text{proj}_1(\text{sdec(x, k)}) = \text{req}$

\[\rightarrow \text{Solution: } X \mapsto w_1 \text{ (and } x \mapsto \{\langle \text{req, n}\rangle\}_k).\]
Existing tools based on constraint solving

→ for a bounded number of sessions only

- **CL-AtSe** [Turuani, RTA’06], **OFMC** [Basin et al, 05] for reachability properties, e.g. secrecy, authentication: the security problem boils down to decide whether a constraint system admits a solution.
Existing tools based on constraint solving

→ for a bounded number of sessions only

- **CL-AtSe** [Turuani, RTA’06], **OFMC** [Basin et al, 05] for reachability properties, e.g. secrecy, authentication: the security problem boils down to decide whether a constraint system admits a solution.

- **DeepSec** [Cheval et al., 18] for equivalence-based properties, e.g. strong secrecy vote-privacy, unlinkability: the security problem boils down to decide whether two (sets) of constraint systems have the same set of solutions.

→ DeepSec also deals with
Going back to monoidal theories

Remember that we associate a semi-ring $S_E$ to a monoidal theory $E$:

- $\mathbb{Z}/2\mathbb{Z}$ for ACUN,
- $\mathbb{Z}$ for AG,
- $\mathbb{Z}/2\mathbb{Z}[X]$ for ACUNh, ...

→ the previous encoding leads to **quadratic** equations.
Going back to monoidal theories

Remember that we associate a semi-ring $S_E$ to a monoidal theory $E$:

- $\mathbb{Z}/2\mathbb{Z}$ for ACUN,
- $\mathbb{Z}$ for AG,
- $\mathbb{Z}/2\mathbb{Z}[X]$ for ACUNh, . . .

→ the previous encoding leads to quadratic equations.

However, it can be shown that they have a specific structure, and this has been exploited to derive the following results:

<table>
<thead>
<tr>
<th>Theory E</th>
<th>$S_E$</th>
<th>Satisfiability</th>
<th>Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACUN</td>
<td>$\mathbb{Z}/2\mathbb{Z}$</td>
<td>PTIME</td>
<td>PTIME</td>
</tr>
<tr>
<td>AG</td>
<td>$\mathbb{Z}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACUNh</td>
<td>$\mathbb{Z}/2\mathbb{Z}[h]$</td>
<td>PTIME</td>
<td>PTIME</td>
</tr>
<tr>
<td>AGh</td>
<td>$\mathbb{Z}[h]$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Chevalier et al., 10] [Delaune et al., 12]
A common difficulty in the active setting

Getting rid of rewriting steps that may occur inside $\mathcal{C} = (\phi; \mathcal{D}; \mathcal{E})$. 
A common difficulty in the active setting

Getting rid of rewriting steps that may occur inside $C = (\phi; D; E)$.

Protocol: $A \rightarrow B : \{n\}_k$

$B \rightarrow A : \{\text{hash}(n)\}_k$

Constraint system $C = (\phi; D; E)$:

1. $w_1 \triangleright \{n\}_k$; $w_2 \triangleright \{\text{hash(sdec}(x, k))\}_k$
2. $X ? x$ with $ar(X) = 1$
3. $E = \emptyset$. 
A common difficulty in the active setting

Getting rid of rewriting steps that may occur inside $C = (\phi; D; E)$.

Protocol:

Constraint system $C = (\phi; D; E)$:

1. $w_1 \triangleright \{n\}_k$; $w_2 \triangleright \{\text{hash(sdec}(x, k))\}_k$

2. $X \triangleright x$ with $ar(X) = 1$

3. $E = \emptyset$.

Two kinds of solutions for $C$:

- $C_1$: either $x$ is replaced by an encryption with $k$, i.e. $\{y\}_k$, and we can compute in advance that $w_2 \triangleright \{\text{hash}(y)\}_k$;

- $C_2$: or $x$ is replaced by something else, and in this case $w_2 \triangleright \{\text{hash(sdec}(x, k))\}_k$.

No further rewriting step is then authorised in $C_1$ and $C_2$.

$\rightarrow$ variant($C$) = $\{C_1, C_2\}$
An equational theory $E$ (represented by a rewrite system $\downarrow$ - possibly modulo $E' = AC$) has the finite variant property if for any term $t$, we can compute a finite set of instances $t\sigma_1, \ldots, t\sigma_n$ such that:

$$\{t\sigma \downarrow | \sigma \in \Sigma\} = \bigcup_{i=1}^{n} \{t\sigma_i \downarrow \theta | \theta \in \Sigma\}$$

where $\Sigma$ is the set of normalized substitutions.

$$\longrightarrow \text{variant}(t) = \{t\sigma_1, \ldots, t\sigma_n\}.$$
Finite Variant Property (FVP)  

An equational theory \( E \) (represented by a rewrite system \( \downarrow \) - possibly modulo \( E' = AC \)) has the finite variant property if for any term \( t \), we can compute a finite set of instances \( t\sigma_1, \ldots, t\sigma_n \) such that:

\[
\{ t\sigma \downarrow \mid \sigma \in \Sigma \} = \bigcup_{i=1}^{n} \{ t\sigma_i \downarrow \theta \mid \theta \in \Sigma \}
\]

where \( \Sigma \) is the set of normalized substitutions.

\[\rightarrow \text{variant}(t) = \{ t\sigma_1, \ldots, t\sigma_n \}.\]

Examples:

- Symmetric, asymmetric encryptions, (blind) signatures, several equational theories used to model modular exponentiation (up to AC) have the FVP;
- homomorphic encryption \( \{ \langle x, y \rangle \}_z = \langle \{ x \}_z, \{ y \}_z \} \) does not satisfy the FVP.
An equational theory $E$ (represented by a rewrite system $\downarrow$ - possibly modulo $E' = AC$) has the finite variant property if for any term $t$, we can compute a finite set of instances $t_{\sigma_1}, \ldots, t_{\sigma_n}$ such that:

$$\{t_{\sigma} \downarrow \mid \sigma \in \Sigma\} = \bigcup_{i=1}^{n} \{t_{\sigma_i} \downarrow \theta \mid \theta \in \Sigma\}$$

where $\Sigma$ is the set of normalized substitutions.

$$\rightarrow \text{variant}(t) = \{t_{\sigma_1}, \ldots, t_{\sigma_n}\}.$$  

A link with rewriting theory!
Actually, the finite variant property is implied by the termination of basic narrowing when $E' = \emptyset$. A bit more tricky when $E' = AC$. 

Finite Variant Property (FVP) [Comon & D., 05]
Some existing tools exploiting the FVP

[Meier et al., 13]  [Escobar et al., 07]  [Cheval et al., 18]
Some existing tools exploiting the FVP

[Meier et al., 13] [Escobar et al., 07] [Cheval et al., 18]

From its input:

rule B: [!Key(k), In(x)] --> [Out(senc{h(sdec(x,k))}k)]

Tamarin computes:

```
rule (modulo AC) B:
  [ !Key( k ), In( x ) ] --> [ Out( senc(h(z), k ) ) ]
variants (modulo AC)
1. k = k.4
   x = x.4
   z = sdec(x.4, k.4)

2. k = x.4
   x = senc(x.5, x.4)
   z = x.5
```
Many UNIF topics are of interest for protocol verification:

- Equational unification and unification modulo theories
- Narrowing
- Higher-Order Unification
- Constraint Solving
- Disunification
- ...
Conclusion

Many UNIF topics are of interest for protocol verification:

- Equational unification and unification modulo theories
- Narrowing
- Higher-Order Unification
- Constraint Solving
- Disunification
- ...

**Challenging theory:** A useful equational theory on which existing tools behave badly is homomorphich encryption (e-voting protocols):

\[
\{x\}_{pk(s)} \star \{y\}_{pk(s)} = \{x + y\}_{pk(s)}
\]
Conclusion

Many UNIF topics are of interest for protocol verification:

- Equational unification and unification modulo theories
- Narrowing
- Higher-Order Unification
- Constraint Solving
- Disunification
- ...

Challenging theory: A useful equational theory on which existing tools behave badly is homomorphic encryption (e-voting protocols):

\[ \{x\}_{pk(s)} \ast \{y\}_{pk(s)} = \{x + y\}_{pk(s)} \]

Thanks you for listening