Transforming Password Protocols to Compose

Steve Kremer
(joint work with Céline Chevalier, Stéphanie Delaune and Mark Ryan)

INRIA Nancy

SRM seminar

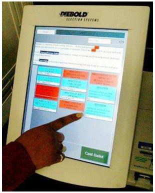
Part I

Introduction: security protocols and formal verification

Cryptographic protocols everywhere!

Cryptographic protocol:

a distributed program which uses cryptographic primitives (e.g. encryption, digital signatures, ...) to ensure a security property (e.g. confidentiality, authentication, anonymity, ...)



Cryptographic protocols everywhere!

Cryptographic protocol:

a distributed program which uses cryptographic primitives (e.g. encryption, digital signatures, . . .) to ensure a security property (e.g. confidentiality, authentication, anonymity, . . .)

FEVAD — 2010 key numbers

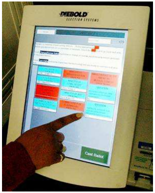
fédération du e-commerce et de la vente à distance

- 78% of French people use remote selling
- 82% of remote selling over the Internet
- online transactions: 25 billion of euros

Cryptographic protocols everywhere!

Cryptographic protocol:

a distributed program which uses cryptographic primitives (e.g. encryption, digital signatures, . . .) to ensure a security property (e.g. confidentiality, authentication, anonymity, . . .)



Legally binding Internet elections in Europe in 2011

- parliamentary elections in Switzerland (several cantons)
- parliamentary election in Estonia (all eligible voters)
- municipal and county elections in Norway (selected municipalities, selected voter groups)

Formal protocol analysis and composition

Nowadays tools exist that succeed in automatically analysing complex protocols, e.g. AVISPA and ProVerif

Formal protocol analysis and composition

Nowadays tools exist that succeed in automatically analysing complex protocols, e.g. AVISPA and ProVerif

But: protocols are analysed in isolation

Other protocols may be executed in parallel

Need for compositional security guarantees

Cryptographic process calculi and composition

Cryptographic pi calculi, e.g., the applied pi calculus or the spi calculus are well-suited for reasoning about composition

if P_1 is secure and P_2 is secure then $P_1 \mid P_2$ is secure

Cryptographic process calculi and composition

Cryptographic pi calculi, e.g., the applied pi calculus or the spi calculus are well-suited for reasoning about composition

$$\text{if } P_1 \approx S_1 \text{ and } P_2 \approx S_2 \text{ then } P_1 \mid P_2 \approx S_1 \mid S_2$$

Cryptographic process calculi and composition

Cryptographic pi calculi, e.g., the applied pi calculus or the spi calculus are well-suited for reasoning about composition

$$\text{if } P_1 \approx S_1 \text{ and } P_2 \approx S_2 \text{ then } P_1 \mid P_2 \approx S_1 \mid S_2$$

There are two main reasons for this

Cryptographic process calculi and composition

Cryptographic pi calculi, e.g., the applied pi calculus or the spi calculus are well-suited for reasoning about composition

$$\text{if } P_1 \approx S_1 \text{ and } P_2 \approx S_2 \text{ then } P_1 \mid P_2 \approx S_1 \mid S_2$$

There are two main reasons for this

1. processes are shown secure in the presence of an arbitrary environment

Cryptographic process calculi and composition

Cryptographic pi calculi, e.g., the applied pi calculus or the spi calculus are well-suited for reasoning about composition

$$\text{if } P_1 \approx S_1 \text{ and } P_2 \approx S_2 \text{ then } P_1 \mid P_2 \approx S_1 \mid S_2$$

There are two main reasons for this

1. processes are shown secure in the presence of an arbitrary environment
2. processes do not share any secrets (this is due to the scope operator)

Cryptographic process calculi and composition

Cryptographic pi calculi, e.g., the applied pi calculus or the spi calculus are well-suited for reasoning about composition

$$\text{if } P_1 \approx S_1 \text{ and } P_2 \approx S_2 \text{ then } P_1 \mid P_2 \approx S_1 \mid S_2$$

There are two main reasons for this

1. processes are shown secure in the presence of an arbitrary environment
2. processes do not share any secrets (this is due to the scope operator)

One would like to show that

$$\text{if } \nu s.P_1 \text{ is secure and } \nu s.P_2 \text{ is secure then } \nu s.(P_1 \mid P_2) \text{ is secure}$$

which does not hold in general

Note that $\nu s.(P_1 \mid P_2)$ differs from $\nu s.P_1 \mid \nu s.P_2$

Guessing attacks

Solution: do not share secrets between protocols

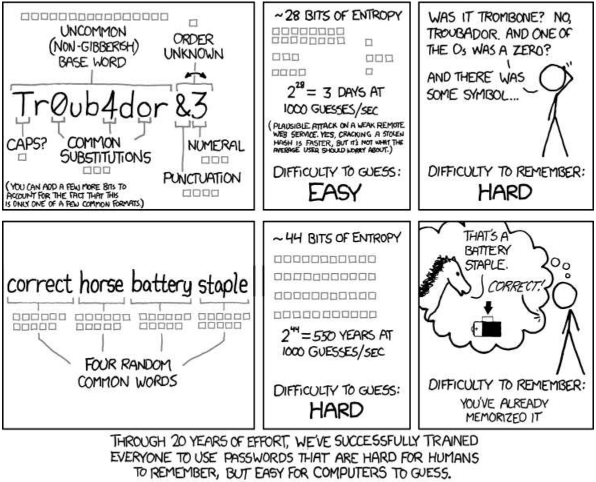
Guessing attacks

Solution: do not share secrets between protocols, but this is not always possible

Guessing attacks

Solution: do not share secrets between protocols, but this is not always possible

Passwords: it is not realistic that users never re-use the same password



Guessing attacks

Solution: do not share secrets between protocols, but this is not always possible

Passwords: it is not realistic that users never re-use the same password

In this talk we investigate the question:

if $\nu p.P_1$ and $\nu p.P_2$ are resistant against guessing attacks on p
is $\nu p.(P_1 \mid P_2)$ also resistant against guessing attacks on p?

Guessing attacks

Solution: do not share secrets between protocols, but this is not always possible

Passwords: it is not realistic that users never re-use the same password

In this talk we investigate the question:

if $\nu p.P_1$ and $\nu p.P_2$ are resistant against guessing attacks on p
is $\nu p.(P_1 \mid P_2)$ also resistant against guessing attacks on p?

An offline guessing or dictionnary attacks consists of two phases

1. the attacker interacts with (one or several sessions of) a protocol
2. the attacker tries offline each of the possible passwords (out of a dictionnary) on the data collected during the first phase

> This talk is based on results from [DKR, CSF'08] and [CDK, FSTTCS'11]

Part II

Modeling protocols and guessing attacks

Terms and equational theories

We consider a simple process language inspired by the applied pi calculus to describe protocols

Messages are modeled using terms

- Abstract algebra given by a signature, *i.e.* a set of function symbols with arities
- Equivalence relation ($=_E$) on terms induced by an equational theory

Example (equational theory)

Consider the signature $\Sigma_{\text{enc}} = \{\text{sdec}, \text{senc}, \text{adec}, \text{aenc}, \text{pk}, \langle\,\rangle, \text{proj}_1, \text{proj}_2\}$

$$\begin{array}{rclrcl} \text{sdec}(\text{senc}(x,y),y) & = & x & \text{proj}_i(\langle x_1, x_2\rangle) & = & x_i \quad (i \in \{1,2\}) \\ \text{senc}(\text{sdec}(x,y),y) & = & x & \text{adec}(\text{aenc}(x,\text{pk}(y)),y) & = & x \end{array}$$

Frames and static equivalence

Terms are regrouped into frames: a set of secrets + a substitution

$$\nu\tilde{n}.\{^{M_1}/_{x_1}, \ldots, ^{M_n}/_{x_n}\}$$

Definition (Static equivalence)

ϕ_1 and ϕ_2 are statically equivalent, $\phi_1 \approx_{\mathsf{E}} \phi_2$, when:

- $dom(\phi_1) = dom(\phi_2)$, and
- for all terms M, N, $(M =_{\mathsf{E}} N)\phi_1$ iff $(M =_{\mathsf{E}} N)\phi_2$

where $(M =_{\mathsf{E}} N)\phi$, if $\phi =_\alpha \nu\tilde{n}.\sigma$, $M\sigma =_{\mathsf{E}} N\sigma$, and $\tilde{n} \cap (fn(M, N)) = \emptyset$.

Example

$$\phi = \nu k.\{^{\mathsf{senc}(s_0,k)}/_{x_1}, ^{k}/_{x_2}\} \not\approx \nu k.\{^{\mathsf{senc}(s_1,k)}/_{x_1}, ^{k}/_{x_2}\} = \phi'$$

because of the test $(\mathsf{sdec}(x_1, x_2), s_0)$. However,

$$\nu k.\{^{\mathsf{senc}(s_0,k)}/_{x_1}\} \approx \nu k.\{^{\mathsf{senc}(s_1,k)}/_{x_1}\}$$

An example protocol

Consider the SPEKE protocol

$$\begin{array}{ll} A \rightarrow B: & \exp(w, ra) \\ B \rightarrow A: & \exp(w, rb) \\ A \rightarrow B: & \mathrm{senc}(ca, \exp(\exp(w, rb), ra)) \\ B \rightarrow A: & \mathrm{senc}(\langle ca, cb \rangle, \exp(\exp(w, ra), rb)) \\ A \rightarrow B: & \mathrm{senc}(cb, \exp(\exp(w, rb), ra)) \end{array}$$

where exp models modular exponenatiation; shared key is $\exp(\exp(w, ra), rb) =_E \exp(\exp(w, rb), ra)$.

An example protocol

Consider the SPEKE protocol

$$\begin{array}{ll} \mathsf{A} \rightarrow \mathsf{B}: & \mathsf{exp}(w, ra) \\ \mathsf{B} \rightarrow \mathsf{A}: & \mathsf{exp}(w, rb) \\ \mathsf{A} \rightarrow \mathsf{B}: & \mathsf{senc}(ca, \mathsf{exp}(\mathsf{exp}(w, rb), ra)) \\ \mathsf{B} \rightarrow \mathsf{A}: & \mathsf{senc}(\langle ca, cb \rangle, \mathsf{exp}(\mathsf{exp}(w, ra), rb)) \\ \mathsf{A} \rightarrow \mathsf{B}: & \mathsf{senc}(cb, \mathsf{exp}(\mathsf{exp}(w, rb), ra)) \end{array}$$

where exp models modular exponenatiation; shared key is $\mathsf{exp}(\mathsf{exp}(w, ra), rb) =_{\mathsf{E}} \mathsf{exp}(\mathsf{exp}(w, rb), ra)$.

Formalized in a simple process calculus: one session of the protocol is $\nu w.(A \mid B)$ where

$$\begin{array}{ll} A = & \nu ra, ca.\mathsf{out}(\mathsf{exp}(w, ra)).\mathsf{in}(x_1). \\ & \mathsf{out}(\mathsf{senc}(ca, ka)).\mathsf{in}(x_2). \\ & \mathsf{out}(\mathsf{senc}(\mathsf{proj}_2(\mathsf{sdec}(x_2, ka)), ka)) \end{array}$$

$$\begin{array}{ll} B = & \nu rb, cb.\mathsf{in}(y_1).\mathsf{out}(\mathsf{exp}(w, rb)). \\ & \mathsf{in}(y_2).\mathsf{out}(\mathsf{senc}(\langle \mathsf{sdec}(y_2, kb), cb \rangle, kb)). \\ & \mathsf{in}(y_3).\ \mathsf{if}\ \mathsf{sdec}(y_3, kb) = cb\ \mathsf{then}\ P\ \mathsf{else}\ 0. \end{array}$$

where $ka = \mathsf{exp}(x_1, ra)$, $kb = \mathsf{exp}(y_1, rb)$

Semantics (informally)

$\nu w.(A \mid B)$ where

$$A = \nu ra, ca.\mathsf{out}(\mathsf{exp}(w, ra)).\mathsf{in}(x_1).\\ \mathsf{out}(\mathsf{senc}(ca, ka)).\mathsf{in}(x_2).\\ \mathsf{out}(\mathsf{senc}(\mathsf{proj}_2(\mathsf{sdec}(x_2, ka)), ka))$$

$$B = \nu rb, cb.\mathsf{in}(y_1).\mathsf{out}(\mathsf{exp}(w, rb)).\\ \mathsf{in}(y_2).\mathsf{out}(\mathsf{senc}(\langle \mathsf{sdec}(y_2, kb), cb\rangle, kb)).\\ \mathsf{in}(y_3).\ \mathsf{if}\ \mathsf{sdec}(y_3, kb) = cb\ \mathsf{then}\ P\ \mathsf{else}\ 0.$$

Semantics (informally)

$\nu w.(A \mid B)$ where

$$A = \nu ra, ca.\mathrm{out}(\exp(w, ra)).\mathrm{in}(x_1).\ \mathrm{out}(\mathrm{senc}(ca, ka)).\mathrm{in}(x_2).\ \mathrm{out}(\mathrm{senc}(\mathrm{proj}_2(\mathrm{sdec}(x_2, ka)), ka))$$

$$B = \nu rb, cb.\mathrm{in}(y_1).\mathrm{out}(\exp(w, rb)).\ \mathrm{in}(y_2).\mathrm{out}(\mathrm{senc}(\langle \mathrm{sdec}(y_2, kb), cb \rangle, kb)).\ \mathrm{in}(y_3).\ \mathrm{if}\ \mathrm{sdec}(y_3, kb) = cb\ \mathrm{then}\ P\ \mathrm{else}\ 0.$$

- νra: generate fresh name

Semantics (informally)

$\nu w.(A \mid B)$ where

$$A = \nu ra, ca.\mathsf{out}(\exp(w, ra)).\mathsf{in}(x_1).\ \mathsf{out}(\mathsf{senc}(ca, ka)).\mathsf{in}(x_2).\ \mathsf{out}(\mathsf{senc}(\mathsf{proj}_2(\mathsf{sdec}(x_2, ka)), ka))$$

$$B = \nu rb, cb.\mathsf{in}(y_1).\mathsf{out}(\exp(w, rb)).\ \mathsf{in}(y_2).\mathsf{out}(\mathsf{senc}(\langle \mathsf{sdec}(y_2, kb), cb\rangle, kb)).\ \mathsf{in}(y_3).\ \text{if } \mathsf{sdec}(y_3, kb) = cb \text{ then } P \text{ else } 0.$$

- νra: generate fresh name
- $\mathsf{out}(\exp(w, ra))$: outputs term on the network; adds $\{^{\exp(w,ra)}/_{z_1}\}$ to the frame

Semantics (informally)

$\nu w.(A \mid B)$ where

$$A = \nu ra, ca.\mathsf{out}(\mathsf{exp}(w, ra)).\mathsf{in}(x_1).\ \mathsf{out}(\mathsf{senc}(ca, ka)).\mathsf{in}(x_2).\ \mathsf{out}(\mathsf{senc}(\mathsf{proj}_2(\mathsf{sdec}(x_2, ka)), ka))$$

$$B = \nu rb, cb.\mathsf{in}(y_1).\mathsf{out}(\mathsf{exp}(w, rb)).\ \mathsf{in}(y_2).\mathsf{out}(\mathsf{senc}(\langle \mathsf{sdec}(y_2, kb), cb\rangle, kb)).\ \mathsf{in}(y_3).\ \mathsf{if}\ \mathsf{sdec}(y_3, kb) = cb\ \mathsf{then}\ P\ \mathsf{else}\ 0.$$

- νra: generate fresh name
- $\mathsf{out}(\mathsf{exp}(w, ra))$: outputs term on the network; adds $\{^{\mathsf{exp}(w,ra)}/_{z_1}\}$ to the frame
- $\mathsf{in}(x_1)$: binds variable x_1 to a term that can be constructed by the attacker from the current frame

Password protocols and offline guessing attacks

Definition from [Baudet05] (inspired from [Corin et al.03])

Definition (Guessing attacks)

A frame $\nu w.\phi$ is resistant to guessing attacks against w iff

$$\nu w.(\phi \mid \{^{w}/_{x}\}) \approx \nu w.(\phi \mid \nu w'.\{^{w'}/_{x}\})$$

A process A is resistant to guessing attack against w if, for every process B such that $A \rightarrow^{*} B$, we have that $\phi(B)$ is resistant to guessing attacks against w.

Composing resistance against passive guessing attacks

Proposition

The three following statements are equivalent:

1. $\nu w.\phi \mid \{^{w}/_{x}\} \approx \nu w.\phi \mid \nu w'.\{^{w'}/_{x}\}$ [Baudet05]
2. $\phi \approx \nu w.\phi$ [Corin et al.03]
3. $\phi \approx \phi\{^{w'}/_{w}\}$

Composing resistance against passive guessing attacks

Proposition

The three following statements are equivalent:

1. $\nu w.\phi \mid \{^{w}/_{x}\} \approx \nu w.\phi \mid \nu w'.\{^{w'}/_{x}\}$ [Baudet05]
2. $\phi \approx \nu w.\phi$ [Corin et al.03]
3. $\phi \approx \phi\{^{w'}/_{w}\}$

It follows from the last point that passive guessing attacks do compose!

Corollary

If $\nu w.\phi_1$ and $\nu w.\phi_2$ are resistant to guessing attacks against w then $\nu w.(\phi_1 \mid \phi_2)$ is also resistant to guessing attacks against w.

A consequence for password-only protocols:
if one session of the protocol is safe against a passive adversary then an unbounded number of sessions are safe against a passive adversary

Results for password protocols: active adversary

The "disjoint" case

Theorem (composition without sharing)

Let $A_1, \ldots, A_k$ be such that A_i is resistant to guessing attack against w_i.

$A_1 \mid \cdots \mid A_k$ *is resistant to guessing attack against* $w_1, \ldots, w_k$.

Results for password protocols: active adversary

The "disjoint" case

Theorem (composition without sharing)

Let $A_1, \dots, A_k$ be such that A_i is resistant to guessing attack against w_i.

$A_1 \mid \cdots \mid A_k$ is resistant to guessing attack against $w_1, \dots, w_k$.

Resistance against guessing attacks does not compose in general as soon as a password is reused!

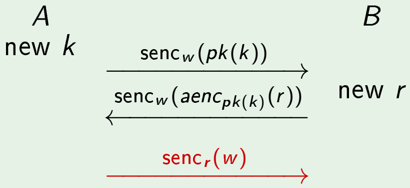
> *Let $\nu w.A_1, \dots, \nu w.A_k$ be such that A_i is resistant to guessing attack against w.*
>
> *$\nu w.(A_1 \mid \cdots \mid A_k)$ is resistant to guessing attack against w.*

does not hold in general

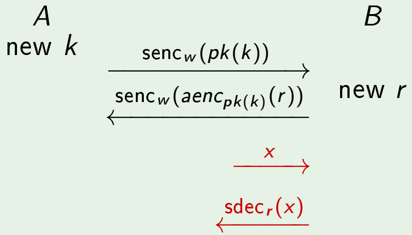
A "chosen protocol" attack

Contrary to passive case, resistance does not compose in general.

EKE variant 1



EKE variant 2



After the execution in which $x = \mathsf{senc}_r(w)$:

$$\phi = \nu w, k, r.(\quad \{^{\mathsf{senc}_w(pk(k))}/_{x_1}\}, \{^{\mathsf{senc}_w(aenc_{pk(k)}(r))}/_{x_2}\}, \{^{\mathsf{senc}_r(w)}/_{x_3}\}, \{^{w}/_{x_4}\})$$

Composition results for password protocols

The "joint state" case

Use unique protocol identifiers pid_i to tag protocols. h is a free symbol in E (modelling a hash function).

Theorem (inter-protocol composition)

Let $pid_1, \ldots, pid_k$ be distinct names, and $\nu w.A_1, \ldots, \nu w.A_k$ be such that $\nu w.A_i$ is resistant to guessing attack against w

$\nu w.(A_1\{^{\mathsf{h}(pid_1, w)}/_{w}\} \mid \cdots \mid A_k\{^{\mathsf{h}(pid_k, w)}/_{w}\})$ is resistant to guessing attack against w.

Composing different sessions of a same protocol

Use a dynamically created tag by preliminary nonce exchange
(same idea as in [Barak, Lindell, Rabin, 2004] and [Arapinis, Delaune, Kremer, 2008])

Definition (transformation adding dynamically created tags)

An ℓ-*party password protocol specification* Π is a process such that:

$$\Pi = \nu w.(\nu \tilde{m}_1.P_1 \mid \ldots \mid \nu \tilde{m}_\ell.P_\ell)$$

where each P_i is a closed plain processes. The processes $\nu \tilde{m}_i.P_i$ are called the roles of Π.

We define $\overline{\Pi} = \nu w.(\nu \tilde{m}_1, n_1.\overline{P_1} \mid \ldots \mid \nu \tilde{m}_\ell, n_\ell.\overline{P_\ell})$ as follows:

$$\overline{P_i} \;=\; \mathsf{in}(x_i^1).\ldots.\mathsf{in}(x_i^{i-1}).\mathsf{out}(n_i).\mathsf{in}(x_i^{i+1}).\mathsf{in}(x_i^\ell).P_i\{^{\mathsf{h}(tag_i,w)}/_w\}$$

where $tag_i = \langle x_i^1, \langle \ldots \langle x_i^{\ell-1}, x_i^\ell \rangle\rangle\rangle$ and $x_i^i = n_i$.

Composition result

Theorem (Inter-session composition)

Let $\Pi = \nu w.(\nu \tilde{m}_1, P_1 \mid \ldots \mid \nu \tilde{m}_\ell.P_\ell)$ *be a password protocol specification resistant to guessing attacks against* w*.*

Let Π' *be such that* $\overline{\Pi} = \nu w.\Pi'$*, and* $\Pi'_1, \ldots \Pi'_p$ *be* p *instances of* Π'*.*

Then we have that $\nu w.(\Pi'_1 \mid \ldots \mid \Pi'_p)$ *is resistant to guessing attacks against* w*.*

Allows to verify one session and conclude security for an unbounded number of sessions of the transformed protocol.

Composition result

Theorem (Inter-session composition)

Let $\Pi = \nu w.(\nu \tilde{m}_1, P_1 \mid \ldots \mid \nu \tilde{m}_\ell.P_\ell)$ be a password protocol specification resistant to guessing attacks against w.

Let Π' be such that $\overline{\Pi} = \nu w.\Pi'$, and $\Pi'_1, \ldots \Pi'_p$ be p instances of Π'.

Then we have that $\nu w.(\Pi'_1 \mid \ldots \mid \Pi'_p)$ is resistant to guessing attacks against w.

Allows to verify one session and conclude security for an unbounded number of sessions of the transformed protocol.

Putting the pieces together: inter-protocol + inter-session composition

- use tags $\mathsf{h}(\langle n_1, \ldots, n_\ell \rangle, \mathsf{h}(pid, w))$ (direct consequence of the theorems)
- more natural tag $\mathsf{h}(\langle pid, \langle n_1, \ldots, n_\ell \rangle \rangle, w)$ by small adaptation of the proofs

A very rough proof sketch

- Assume that tagged protocol admits guessing attack. Hence there exists an attack trace.

A very rough proof sketch

- Assume that tagged protocol admits guessing attack. Hence there exists an attack trace.
- Let $t_1, \ldots, t_k$ be the tags computed on the attack trace. Regroup roles into buckets which agree on the same tag t_i.

A very rough proof sketch

- Assume that tagged protocol admits guessing attack. Hence there exists an attack trace.
- Let $t_1, \ldots, t_k$ be the tags computed on the attack trace. Regroup roles into buckets which agree on the same tag t_i.
- Show that tag t_i can be replaced by simple tag $\mathsf{h}(sid_i, w_i)$ (sid_i distinct constants) to obtain a similar executable trace, which admits a guessing attack on some w_i. Note that sid_i is a "magically" shared tag.

A very rough proof sketch

- Assume that tagged protocol admits guessing attack. Hence there exists an attack trace.
- Let $t_1, \ldots, t_k$ be the tags computed on the attack trace. Regroup roles into buckets which agree on the same tag t_i.
- Show that tag t_i can be replaced by simple tag $\mathrm{h}(sid_i, w_i)$ (sid_i distinct constants) to obtain a similar executable trace, which admits a guessing attack on some w_i. Note that sid_i is a “magically” shared tag.
- From disjoint composition result conclude that there exists a guessing attack on one instance of the protocol.

A very rough proof sketch

- Assume that tagged protocol admits guessing attack. Hence there exists an attack trace.
- Let $t_1, \ldots, t_k$ be the tags computed on the attack trace. Regroup roles into buckets which agree on the same tag t_i.
- Show that tag t_i can be replaced by simple tag $\mathsf{h}(sid_i, w_i)$ (sid_i distinct constants) to obtain a similar executable trace, which admits a guessing attack on some w_i. Note that sid_i is a “magically” shared tag.
- From disjoint composition result conclude that there exists a guessing attack on one instance of the protocol.
- We showed that this way of tagging preserves resistance against guessing attacks. Hence, there exists a guessing attack on the untagged protocol.

Conclusion and future work

- Composition of password protocols: inter protocol and inter session composition
- Allows to safely limit verification to one session of a protocol
- Resistance against offline guessing attacks is not a protocol goal in its own
  $\rightarrow$ want to guarantee other properties, e.g. authentication, under composition
  $\rightarrow$ trace properties composition should directly follow from our proof (some tedious work to formalize the properties to be done)
- Composition of more general equivalence properties? (much more difficult)