Logic for Privacy in Security Protocols

Sebastian Mödersheim
Danmarks Tekniske Universitet

Based on joint work with
Laouen Fernet, Sébastien Gondron, Thomas Gross, Luca Viganò

Workshop on Formal Methods in Security
Reykjavik May 23, 2023

Why Privacy?

Vote in public?

- Advantage: Verifiability
- Serious disadvantage:
  You may not be free to vote what you want.
  - ★ Your boss, spouse, friends, potential future employer can see what you vote.
  - ★ Somebody may bribe or threaten you for voting.

General need for privacy:

- If your actions are observable it can mean subtle restrictions on your freedom.

Alpha-Beta Privacy

Alpha-Beta Privacy

- Novel approach based on Herbrand logic
- Declarative privacy goal specification
  - ★ Specify what private information you deliberately release
  - ★ Allows for incremental approach: discovering the strongest privacy property.
- Reachability problem
  - ★ There is just one reality in each state
- Easier to reason about
  - ★ manually: often easy proof arguments
  - ★ automatically: symbolic/rewriting approaches
  - ★ noname Tool: new automated analysis for bounded sessions
- Deeper understanding: relating to existing approaches

Idea

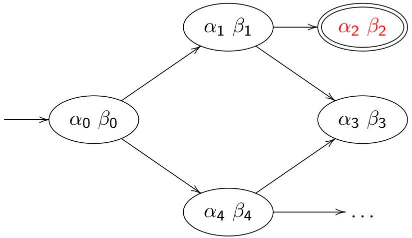
Inspiration/Idea

In zero-knowledge proofs we can usually specify a statement that is being proved.

- Definitely, that statement is revealed to the verifier
  - e.g. "Alice is over 18"
- The verifier (or others) should not learn anything else
  - e.g. "Alice is over 65"
- Everybody can draw conclusions from everything they learned
  - e.g. "Alice is over 15"

Can we do something logical in general for privacy?

State Space



Every state includes two formulae:

- α_i: the information that has been deliberately released so far
  - ★ e.g. the end result of an election
- β_i: the observations that the intruder has made so far.
  - ★ e.g. cryptographic messages exchanged

Attack states:

- when β_i allows the intruder to derive more than α_i.

Alphabet Σ contains:

- cryptographic functions and predicates to represent intruder knowledge
- distinguished subset $\Sigma_0 \subseteq \Sigma$ the high-level information
  - e.g. voters, candidates, natural numbers

In every state:

- α over alphabet Σ_0
- β over alphabet Σ
- $fv(\alpha) \subseteq fv(\beta)$

Alphabet Σ contains:

- cryptographic functions and predicates to represent intruder knowledge
- distinguished subset $\Sigma_0 \subseteq \Sigma$ the high-level information
  - ★ e.g. voters, candidates, natural numbers

In every state:

- α over alphabet Σ_0
- β over alphabet Σ
- $\mathit{fv}(\alpha) \subseteq \mathit{fv}(\beta)$

Definition (α-β privacy)

Privacy in a state (α, β) holds iff
for every Σ_0-model $\mathcal{I} \models \alpha$ exists a Σ-model $\mathcal{I}' \models \beta$ such that
$\mathcal{I}$ and $\mathcal{I}'$ agree on the interpretation of the symbols in Σ_0 and $\mathit{fv}(\alpha)$.

Thus from β the intruder does not learn anything (except "technical" stuff) that is not implied by α already.

Example

Three RFID tags have interacted with the airport passport reader:

$$\alpha \equiv x_1, x_2, x_3 \in \texttt{Agent}$$

The intruder has observed some messages that allow to deduce

$$\beta \models x_1 \dot{\neq} x_3$$

This violates α-β privacy
because for some models of α there is no corresponding model of β.

Intruder Performs A Symbolic Execution

Example Transaction

$$\star\ x \in \texttt{Agent}.\ \star\ \ y \in \{\mathsf{yes}, \mathsf{no}\}.$$

$$
\begin{array}{l}
\mathsf{rcv}(M).\ \mathsf{try}\ N \doteq \mathsf{dcrypt}(\mathsf{inv}(\mathsf{pk}(\mathsf{s})), M) \\
\qquad \mathsf{in\ if}\ y \doteq \mathsf{yes\ then}\ \nu r.\mathsf{snd}(\mathsf{crypt}(\mathsf{pk}(x), \mathsf{pair}(\mathsf{yes}, N), r)) \\
\qquad\qquad\qquad \mathsf{else}\ \nu r.\mathsf{snd}(\mathsf{crypt}(\mathsf{pk}(x), \mathsf{no}, r)) \\
\qquad \mathsf{catch}\ 0
\end{array}
$$

α	$x \in \texttt{Agent}, y \in \{\mathsf{yes}, \mathsf{no}\}$
β	*true*
γ	$x \doteq a, y \doteq \mathsf{yes}$

The intruder knows that x and y are picked from the respective domains.
γ: what really happened—not seen by intruder.

Intruder Performs A Symbolic Execution

Example Transaction

$$\mathsf{rcv}(M).\ \mathsf{try}\ N \doteq \mathsf{dcrypt}(\mathsf{inv}(\mathsf{pk}(\mathsf{s})), M)$$
$$\mathsf{in\ if}\ y \doteq \mathsf{yes\ then}\ \nu r.\mathsf{snd}(\mathsf{crypt}(\mathsf{pk}(x), \mathsf{pair}(\mathsf{yes}, N), r))$$
$$\mathsf{else}\ \nu r.\mathsf{snd}(\mathsf{crypt}(\mathsf{pk}(x), \mathsf{no}, r))$$
$$\mathsf{catch}\ 0$$

α	$x \in \mathtt{Agent}, y \in \{\mathsf{yes}, \mathsf{no}\}$
β	*true*
γ	$x \doteq a, y \doteq \mathsf{yes}$

Intruder can pick any recipe r for M:

- intruder knowledge, closed under public functions
- there infinitely many
- say $r = \mathsf{crypt}(\mathsf{pk}(s), a)$

Intruder Performs A Symbolic Execution

Example Transaction

$$\text{try } N \doteq \text{dcrypt}(\text{inv}(\text{pk}(\text{s})), \text{crypt}(\text{pk}(s), a))$$
$$\text{in if } y \doteq \text{yes then } \nu r.\text{snd}(\text{crypt}(\text{pk}(x), \text{pair}(\text{yes}, N), r))$$
$$\text{else } \nu r.\text{snd}(\text{crypt}(\text{pk}(x), \text{no}, r))$$
$$\text{catch } 0$$

α	$x \in \texttt{Agent}, y \in \{\text{yes}, \text{no}\}$
β	*true*
γ	$x \doteq a, y \doteq \text{yes}$

Algebra: $\text{dcrypt}(\text{inv}(x), \text{crypt}(x, y, z)) =_E y$
Thus: decryption works $N \doteq a$—and the intruder knows it.

Intruder Performs A Symbolic Execution

Example Transaction

$$\text{if } y \doteq \text{yes then } \nu r.\mathsf{snd}(\mathsf{crypt}(\mathsf{pk}(x), \mathsf{pair}(\mathsf{yes}, a), r))$$
$$\text{else } \nu r.\mathsf{snd}(\mathsf{crypt}(\mathsf{pk}(x), \mathsf{no}, r))$$

α	$x \in \mathtt{Agent}, y \in \{\text{yes}, \text{no}\}$			
β		$struct_1$ $\phi_1 \equiv y \doteq \text{yes}$	$struct_2$ $\phi_2 \equiv y \doteq \text{no}$	$concr$
	l_1	$\mathsf{crypt}(\mathsf{pk}(x), \mathsf{pair}(\mathsf{yes}, a), r)$	$\mathsf{crypt}(\mathsf{pk}(x), \mathsf{no}, r)$	$\mathsf{crypt}(\mathsf{pk}(a), \mathsf{pair}(\mathsf{yes}, a), r)$
	$\wedge \bigvee_{i=1}^{2} \phi_i \wedge struct_i \sim concr$			
γ	$x \doteq a, y \doteq \text{yes}$			

The intruder does not know whether the condition is true:

- structural knowledge $struct_1$ or $struct_2$ – the structure the message could have
- *concr* – the concrete message observed.
- one of the ϕ_i is the case and *concr* is statically equivalent to $struct_i$.

Static Equivalence of Frames

$F_1 \sim F_2$ iff for all recipes r_1, r_2:

$$F_1(r_1) \doteq F_1(r_2) \text{ iff } F_2(r_1) \doteq F_2(r_2).$$

Example: encryption without randomization:

	$struct_1$ $\phi_1 \equiv y \doteq \text{yes}$	$struct_2$ $\phi_2 \equiv y \doteq \text{no}$	$concr$
l_1	$\text{crypt}(\text{pk}(x), \text{pair}(\text{yes}, a))$	$\text{crypt}(\text{pk}(x), \text{no})$	$\text{crypt}(\text{pk}(a), \text{pair}(\text{yes}, a))$

Static Equivalence of Frames

$F_1 \sim F_2$ iff for all recipes r_1, r_2:

$$F_1(r_1) \doteq F_1(r_2) \text{ iff } F_2(r_1) \doteq F_2(r_2).$$

Example: encryption without randomization:

	$struct_1$ $\phi_1 \equiv y \doteq \text{yes}$	$struct_2$ $\phi_2 \equiv y \doteq \text{no}$	$concr$
l_1	$\text{crypt}(\text{pk}(x), \text{pair}(\text{yes}, a))$	$\text{crypt}(\text{pk}(x), \text{no})$	$\text{crypt}(\text{pk}(a), \text{pair}(\text{yes}, a))$

- $r_1 = l_1$ and $r_2 = \text{crypt}(\text{pk}(a), no)$ is
  - ★ unequal in *concr*
  - ★ but equal in $struct_2$ if $x \doteq a$.

Static Equivalence of Frames

$F_1 \sim F_2$ iff for all recipes r_1, r_2:

$$F_1(r_1) \doteq F_1(r_2) \text{ iff } F_2(r_1) \doteq F_2(r_2).$$

Example: encryption without randomization:

	$struct_1$ $\phi_1 \equiv y \doteq \text{yes}$	$struct_2$ $\phi_2 \equiv y \doteq \text{no}$	$concr$
l_1	$\text{crypt}(\text{pk}(x), \text{pair}(\text{yes}, a))$	$\text{crypt}(\text{pk}(x), \text{no})$	$\text{crypt}(\text{pk}(a), \text{pair}(\text{yes}, a))$

- $r_1 = l_1$ and $r_2 = \text{crypt}(\text{pk}(a), no)$ is
  - ★ unequal in *concr*
  - ★ but equal in $struct_2$ if $x \doteq a$.
- Thus, $\beta \models \neg(x \doteq a \wedge y = \text{no})$ which does not follow from α.

Static Equivalence of Frames

$F_1 \sim F_2$ iff for all recipes r_1, r_2:

$$F_1(r_1) \doteq F_1(r_2) \text{ iff } F_2(r_1) \doteq F_2(r_2).$$

Example: encryption without randomization:

	struct$_1$ $\phi_1 \equiv y \doteq \mathsf{yes}$	*struct*$_2$ $\phi_2 \equiv y \doteq \mathsf{no}$	*concr*
l_1	$\mathsf{crypt}(\mathsf{pk}(x), \mathsf{pair}(\mathsf{yes}, a))$	$\mathsf{crypt}(\mathsf{pk}(x), \mathsf{no})$	$\mathsf{crypt}(\mathsf{pk}(a), \mathsf{pair}(\mathsf{yes}, a))$

- $r_1 = l_1$ and $r_2 = \mathsf{crypt}(\mathsf{pk}(a), no)$ is
  - unequal in *concr*
  - but equal in *struct*$_2$ if $x \doteq a$.
- Thus, $\beta \models \neg(x \doteq a \wedge y = \mathsf{no})$ which does not follow from α.
- The same experiment works for any $x \in \mathtt{Agent}$. Thus even $\beta \models y = \mathsf{yes}$.

Another Round

Example Transaction

$\star\ x' \in \mathtt{Agent}.\ \star\ y' \in \{\text{yes}, \text{no}\}.$

$\text{rcv}(M').\ \text{try}\ N' \doteq \text{dcrypt}(\text{inv}(\text{pk}(\text{s})), M')$

$\quad \text{in if}\ y' \doteq \text{yes then}\ \nu r'.\text{snd}(\text{crypt}(\text{pk}(x'), \text{pair}(\text{yes}, N'), r'))$

$\quad\quad \text{else}\ \nu r'.\text{snd}(\text{crypt}(\text{pk}(x'), \text{no}, r'))$

$\quad \text{catch}\ 0$

α	$x \in \mathtt{Agent}, y \in \{\text{yes}, \text{no}\}, x' \in \mathtt{Agent}, y' \in \{yes, no\}$			
β		$struct_1$ $\phi_1 \equiv y \doteq \text{yes}$	$struct_2$ $\phi_2 \equiv y \doteq \text{no}$	$concr$
	l_1	$\text{crypt}(\text{pk}(x), \text{pair}(\text{yes}, a), r)$	$\text{crypt}(\text{pk}(x), \text{no}, r)$	$\text{crypt}(\text{pk}(a), \text{pair}(\text{yes}, a), r)$
	$\wedge \bigvee_{i=1}^{2} \phi_i \wedge struct_i \sim concr$			
γ	$x \doteq a, y \doteq \text{yes}, x' \doteq b, y' \doteq \text{no}$			

Another Round

Example Transaction

$$
\begin{aligned}
\mathsf{rcv}(M').\ & \mathsf{try}\ N' \doteq \mathsf{dcrypt}(\mathsf{inv}(\mathsf{pk}(s)), M') \\
& \mathsf{in\ if}\ y' \doteq \mathsf{yes\ then}\ \nu r'.\mathsf{snd}(\mathsf{crypt}(\mathsf{pk}(x'), \mathsf{pair}(\mathsf{yes}, N'), r')) \\
& \qquad\qquad \mathsf{else}\ \nu r'.\mathsf{snd}(\mathsf{crypt}(\mathsf{pk}(x'), \mathsf{no}, r')) \\
& \mathsf{catch}\ 0
\end{aligned}
$$

α	$x \in \mathtt{Agent}, y \in \{\mathsf{yes}, \mathsf{no}\}, x' \in \mathtt{Agent}, y' \in \{yes, no\}$			
β		$struct_1$ $\phi_1 \equiv y \doteq \mathsf{yes}$	$struct_2$ $\phi_2 \equiv y \doteq \mathsf{no}$	$concr$
	l_1	$\mathsf{crypt}(\mathsf{pk}(x), \mathsf{pair}(\mathsf{yes}, a), r)$	$\mathsf{crypt}(\mathsf{pk}(x), \mathsf{no}, r)$	$\mathsf{crypt}(\mathsf{pk}(a), \mathsf{pair}(\mathsf{yes}, a), r)$
	$\wedge \bigvee_{i=1}^{2} \phi_i \wedge struct_i \sim concr$			
γ	$x \doteq a, y \doteq \mathsf{yes}, x' \doteq b, y' \doteq \mathsf{no}$			

Let's use l_1 as input message!

Another Round

Example Transaction

$$\text{try } N' \doteq \text{dcrypt}(\text{inv}(\text{pk}(s)), \text{crypt}(\text{pk}(x), \text{pair}(\text{yes}, a), r))$$
$$\text{in if } y' \doteq \text{yes then } \nu r'.\text{snd}(\text{crypt}(\text{pk}(x'), \text{pair}(\text{yes}, N'), r'))$$
$$\text{else } \nu r'.\text{snd}(\text{crypt}(\text{pk}(x'), \text{no}, r'))$$
$$\text{catch } 0$$

α	$x \in \texttt{Agent}, y \in \{\text{yes}, \text{no}\}, x' \in \texttt{Agent}, y' \in \{yes, no\}$			
β		$struct_1$ $\phi_1 \equiv y \doteq \text{yes}$	$struct_2$ $\phi_2 \equiv y \doteq \text{no}$	$concr$
	l_1	$\text{crypt}(\text{pk}(x), \text{pair}(\text{yes}, a), r)$	$\text{crypt}(\text{pk}(x), \text{no}, r)$	$\text{crypt}(\text{pk}(a), \text{pair}(\text{yes}, a), r)$
	$\wedge \bigvee_{i=0}^{2} \phi_i \wedge struct_i \sim concr$			
γ	$x \doteq a, y \doteq \text{yes}, x' \doteq b, y' \doteq \text{no}$			

Now the intruder cannot tell whether the decryption works—it depends on whether $x \doteq s$.

Another Round

Evaluating the conditions gives now 6 cases:

$$
\begin{array}{lll|ll}
x \doteq s & y' \doteq \mathsf{yes} & y \doteq \mathsf{yes} & \mathsf{snd}(\ldots yes) & struct_1 \\
x \doteq s & y' \doteq \mathsf{yes} & y \doteq \mathsf{no} & \mathsf{snd}(\ldots yes) & struct_2 \\
x \doteq s & y' \doteq \mathsf{no} & y \doteq \mathsf{yes} & \mathsf{snd}(\ldots no) & struct_1 \\
x \doteq s & y' \doteq \mathsf{no} & y \doteq \mathsf{no} & \mathsf{snd}(\ldots no) & struct_2 \\
x \not\doteq s & & y \doteq \mathsf{yes} & 0 & struct_1 \\
x \not\doteq s & & y \doteq \mathsf{no} & 0 & struct_2
\end{array}
$$

Another Round

Since the intruder can observe that no message is sent, only two cases remain:

$$\begin{array}{ccc|cc} x \dot{\neq} s & & y \doteq \mathsf{yes} & 0 & struct_1 \\ x \dot{\neq} s & & y \doteq \mathsf{no} & 0 & struct_2 \end{array}$$

Thus the intruder can derive: $\beta \models x \dot{\neq} s$.

Strongest Privacy Goal

In general, when detecting such a violation of (α, β)-privacy, one has two options:

- Strengthen the protocol, e.g., send a decoy message instead of 0.
- Declassification of some information, e.g., release to α that $x \dot{\neq} s$.

Incremental exploration of the strongest privacy goal that a protocol can achieve

- Start with no α-releases (just domain constraints).
- Whenever a violation is found, make a minimal release that fixes that violation.
- Repeat until no more violations are found.

Examples:

- Abadi-Fournet protocol from *Private Authentication*, TCS 2004.
- ICAO BAC – e.g. French vs. British implementation

Noname Tool

A decision procedure for (α, β)-privacy for a bounded number of transitions.

- Symbolic representation for the non-deterministic choices
- Symbolic representation for intruder-chosen recipes
- Handling of constructor/destructor theories
- Number of Case Studies (Unlinkability, Privacy)

Ask for more on Noname and attacks :-)

Alpha-Beta Privacy

Alpha-Beta Privacy

- Novel approach based on Herbrand logic
- Declarative privacy goal specification
  - ★ Specify what private information you deliberately release
  - ★ Allows for incremental approach: discovering the strongest privacy property.
- Reachability problem
  - ★ There is just one reality in each state
- Easier to reason about
  - ★ manually: often easy proof arguments
  - ★ automatically: symbolic/rewriting approaches
  - ★ noname Tool: new automated analysis for bounded sessions
- Deeper understanding: relating to existing approaches