DTU

MSc Thesis - Laouen Fernet - s192612@dtu.dk

Deciding Fragments of $(\alpha, \beta, \gamma, \delta)$-Privacy

Table of Contents

Introduction

Preliminaries

Modelling the intruder

Decision procedure

Conclusion

Privacy

Relevant in many fields, a security goal of its own:

- electronic voting, digital health information, mobile payments...
- distributed systems in general
- more than just secrecy

De facto standard = indistinguishability

- given two possible worlds, can they be distinguished?
- automated verification is difficult
- specification of goals is not intuitive
- there is no guarantee that every privacy aspect has been covered

Novel approach

(α, β)-privacy = logical approach with many advantages

- declarative and intuitive
- recast privacy as a reachability problem
- decidable fragments: possibility for automated verification

Table of Contents

Introduction

Preliminaries

Modelling the intruder

Decision procedure

Conclusion

Overview

Based on the paper (α, β)-*Privacy* (Mödersheim and Viganó, ACM Trans. Priv. Secur. 22, 2019)

Formalisation in Herbrand logic:

- Modelling of the intruder
- Declaration of (α, β)-privacy goals

Formal verification of privacy in communication protocols:

- Procedure for decidable fragments
- Witness of privacy violations, if any

Table of Contents

Introduction

Preliminaries
- Herbrand Logic
- Frames
- (α, β)-privacy

Modelling the intruder

Decision procedure

Conclusion

Grammar

$$
\begin{aligned}
\langle \textit{Term} \rangle \quad & ::= \langle \textit{Variable} \rangle \mid \langle \textit{Function} \rangle(\langle \textit{Term} \rangle, \ldots, \langle \textit{Term} \rangle) \\
\langle \textit{Formula} \rangle \quad & ::= \langle \textit{Term} \rangle = \langle \textit{Term} \rangle \\
& \mid \quad \langle \textit{Relation} \rangle(\langle \textit{Term} \rangle, \ldots, \langle \textit{Term} \rangle) \\
& \mid \quad \neg \langle \textit{Formula} \rangle \\
& \mid \quad \langle \textit{Formula} \rangle \wedge \langle \textit{Formula} \rangle \\
& \mid \quad \exists \langle \textit{Variable} \rangle . \langle \textit{Formula} \rangle
\end{aligned}
$$

Table of Contents

Introduction

Preliminaries

Herbrand Logic

Frames

(α, β)-privacy

Modelling the intruder

Decision procedure

Conclusion

Frame

Frames encode the knowledge of messages based on the protocol specification

$$F = \{\!| \mathsf{l}_1 \mapsto t_1, \ldots, \mathsf{l}_k \mapsto t_k |\!\}$$

l_i: distinguished constant (label)

t_i: term without any destructor or verifier

domain of F: $\{\mathsf{l}_1, \ldots, \mathsf{l}_k\}$ image of F: $\{t_1, \ldots, t_k\}$

Recipes and generable terms

Frames allow to reason about actions taken and not simply messages themselves

Set of recipes: least set that contains $\mathsf{l}_1, \ldots, \mathsf{l}_k$ and that is closed under cryptographic operators

$F\{\!|\, r \,|\!\}$: application of recipe r to frame F

t is generable: there is r such that $t = F\{\!|\, r \,|\!\}$

Static equivalence

$\mathcal{F}_1 \sim \mathcal{F}_2$:

$$\forall (r_1, r_2), \mathcal{F}_1\{\!| r_1 |\!\} \approx \mathcal{F}_1\{\!| r_2 |\!\} \iff \mathcal{F}_2\{\!| r_1 |\!\} \approx \mathcal{F}_2\{\!| r_2 |\!\}$$

Can be axiomatised in Herbrand logic

Example:

$$\mathcal{F}_1 = \{\!| \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, t_1), \mathsf{l}_2 \mapsto k, \mathsf{l}_3 \mapsto t_1 |\!\}$$
$$\mathcal{F}_2 = \{\!| \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, t_2), \mathsf{l}_2 \mapsto k, \mathsf{l}_3 \mapsto t_2 |\!\}$$

$\mathcal{F}_1 \sim \mathcal{F}_2$ because $t_1 \neq t_2$ but no way to distinguish the frames

Table of Contents

Introduction

Preliminaries

Herbrand Logic

Frames

(α, β)-privacy

Modelling the intruder

Decision procedure

Conclusion

Idea

Formula α: high-level information which is voluntarily disclosed, based on Σ_0

Σ_0 contains only non-technical information

Formula β: includes the technical information, e.g. cryptographic messages exchanged during the execution of the protocol

$\Sigma_0 \subsetneq \Sigma$

Violation of privacy: logically deriving information from β that does not follow from α alone

Message-analysis problem

θ: a model of α (interpretation of symbols making the formula true)

$struct = \{\!|\, \mathsf{l}_1 \mapsto t_1, \ldots, \mathsf{l}_k \mapsto t_k \,|\!\}$: a frame for some $t_1, \ldots, t_k \in \mathcal{T}_\Sigma(fv(\alpha))$ (structural knowledge)

$concr = \theta(struct)$: one execution of the protocol (concrete knowledge)

$\beta \equiv MsgAna(\alpha, struct, \theta)$: knowledge of α and $concr \sim struct$

Idea of the procedure

- Study a message-analysis problem
- Generate a formula ϕ based on static equivalence of frames
- ϕ encodes relations between variables (e.g. $x = z \wedge y \neq \mathsf{h}(x) \ldots$)
- Check if ϕ derives from α

Table of Contents

Introduction

Preliminaries

Modelling the intruder

Decision procedure

Conclusion

Table of Contents

Introduction

Preliminaries

Modelling the intruder
- Intruder theory
- Frames with shorthands

Decision procedure

Conclusion

Intruder theory

- $\Sigma_{pub} \subseteq \Sigma_f$: public functions, i.e. the intruder can apply them
- $\mathcal{V}_{pub} \subseteq \mathcal{V}$: variables with public range, i.e. the intruder knows all possible values of the variables instantiation
- $\Sigma_{op} \subseteq \Sigma_f$: cryptographic operators (constructors, destructors, verifiers)
- Set of algebraic equations: characterises the operators

Convergent intruder theory I

Requirements for the algebraic equations:

- $\mathsf{destr}(k_1, \ldots, k_m, \mathsf{constr}(t_1, \ldots, t_n)) \approx t_i$
- $\mathsf{verif}(k_1, \ldots, k_m, \mathsf{constr}(t_1, \ldots, t_n)) \approx \mathsf{yes}$
- Can have 0 keys ($m = 0$)
- Every destructor has a corresponding verifier
- No ambiguous equations: either different constructor or same arguments

Convergent intruder theory II

$\approx$: least relation from the algebraic equations

Convergent rewriting system: $LHS \to RHS$

Analysing one term: required keys and derivable subterms

$$\begin{aligned} ana(\mathsf{constr}(t_1,\ldots,t_n)) = (&\{k_1,\ldots,k_m\}, \\ &\{(\mathsf{destr}, t_i) \mid \\ &\quad \mathsf{destr}(k_1,\ldots,k_m,\mathsf{constr}(t_1,\ldots,t_n)) \approx t_i\}) \end{aligned}$$

Example cryptographic operators

Constructors	Destructors	Verifiers	Properties
pub, priv			
crypt	dcrypt	vcrypt	$\text{dcrypt}(\text{priv}(s), \text{crypt}(\text{pub}(s), r, t)) \approx t$ $\text{vcrypt}(\text{priv}(s), \text{crypt}(\text{pub}(s), r, t)) \approx \text{yes}$
sign	retrieve	vsign	$\text{retrieve}(\text{sign}(\text{priv}(s), t)) \approx t$ $\text{vsign}(\text{pub}(s), \text{sign}(\text{priv}(s), t)) \approx \text{yes}$
scrypt	dscrypt	vscrypt	$\text{dscrypt}(k, \text{scrypt}(k, t)) \approx t$ $\text{vscrypt}(k, \text{scrypt}(k, t)) \approx \text{yes}$
pair	proj_i	vpair	$\text{proj}_i(\text{pair}(t_1, t_2)) \approx t_i$ $\text{vpair}(\text{pair}(t_1, t_2)) \approx \text{yes}$
h			

Table: Example set Σ_{op}

Table of Contents

Introduction

Preliminaries

Modelling the intruder

- Intruder theory
- Frames with shorthands

Decision procedure

Conclusion

Frame with shorthands I

Frames with shorthands extend the previous definition of frames

$$F' = \{\!| \, \mathsf{l}_1 \mapsto t_1, \ldots, \mathsf{l}_k \mapsto t_k, \mathsf{m}_1 \mapsto s_1, \ldots, \mathsf{m}_n \mapsto s_n \, |\!\}$$

- $F = \{\!| \, \mathsf{l}_1 \mapsto t_1, \ldots, \mathsf{l}_k \mapsto t_k \, |\!\}$: frame
- m_j: recipes over the l_i
- s_j: terms that do not contain any l_i
- $F\{\!| \, \mathsf{m}_j \, |\!\} \approx s_j$
- $\mathsf{m}_1 \mapsto s_1, \ldots, \mathsf{m}_n \mapsto s_n$: shorthands

Frame with shorthands II

Example:

$$\mathcal{F} = \{\!| \, \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, t), \mathsf{l}_2 \mapsto k \, |\!\}$$
$$\mathcal{F}' = \{\!| \, \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, t), \mathsf{l}_2 \mapsto k, \mathsf{dscrypt}(\mathsf{l}_2, \mathsf{l}_1) \mapsto t \, |\!\}$$

$$\mathcal{F}\{\!| \, \mathsf{dscrypt}(\mathsf{l}_2, \mathsf{l}_1) \, |\!\} = \mathsf{dscrypt}(k, \mathsf{scrypt}(k, t)) \approx t = \mathcal{F}'\{\!| \, \mathsf{dscrypt}(\mathsf{l}_2, \mathsf{l}_1) \, |\!\}$$

Table of Contents

Introduction

Preliminaries

Modelling the intruder

Decision procedure

Conclusion

Illustration

Example:

$$\theta = \{x \mapsto 0, y \mapsto 1, z \mapsto 0\}$$
$$struct = \{\!| \, \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, x), \mathsf{l}_2 \mapsto \mathsf{scrypt}(k, y), \mathsf{l}_3 \mapsto \mathsf{scrypt}(k, z) \, |\!\}$$
$$concr = \{\!| \, \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, 0), \mathsf{l}_2 \mapsto \mathsf{scrypt}(k, 1), \mathsf{l}_3 \mapsto \mathsf{scrypt}(k, 0) \, |\!\}$$

$\alpha \equiv x, y, z \in \{0, 1\} \wedge x + y + z = 1 \qquad \beta \equiv MsgAna(\alpha, struct, \theta)$

Intruder deduction:
$concr\{\!| \, \mathsf{l}_1 \, |\!\} = concr\{\!| \, \mathsf{l}_3 \, |\!\}$: two messages are equal at the concrete level
$struct\{\!| \, \mathsf{l}_1 \, |\!\} = struct\{\!| \, \mathsf{l}_3 \, |\!\}$: they must also be equal at the structural level
$x = z$: violation of privacy!

Table of Contents

Introduction

Preliminaries

Modelling the intruder

Decision procedure
- Composition
- Analysis
- Relations between variables

Conclusion

Composition in a ground frame I

Input: $concr, t$
Output: $compose(concr, t)$ = set of recipes over labels of $concr$ to compose t

Two methods to compose the ground term:

- use a label directly if it maps to the term
- if the top-level is a public function, try to compose all arguments

Composition in a ground frame II

Example:

$$concr = \{\!|\ \mathsf{l}_1 \mapsto a, \mathsf{l}_2 \mapsto \mathsf{h}(a), \mathsf{l}_3 \mapsto \mathsf{scrypt}(a, c)\ |\!\}$$

$compose(concr, \mathsf{h}(a)) = \{\mathsf{h}(\mathsf{l}_1), \mathsf{l}_2\}$
The intruder knows two ways to compose $\mathsf{h}(a)$

$compose(concr, c) = \{\}$
The intruder cannot compose c, the encrypted term needs to be decrypted first

Composition in a structural frame I

Input: $\theta, struct, t$
Output: $composeUnder(\theta, struct, t)$ = set of pairs (recipe, substitution) to compose t

Three methods to compose the term:

- try to use labels with a corresponding substitution
- if the term is a variable with public range, its concrete value $\theta(t)$ (a constant) is a recipe under the substitution $\{t \mapsto \theta(t)\}$
- if the top-level is a public function, try to compose all arguments (and combine the substitutions)

Composition in a structural frame II

Example:

$$\theta = \{x \mapsto a, y \mapsto b\}$$
$$struct = \{\!| \ \mathsf{l}_1 \mapsto x, \mathsf{l}_2 \mapsto \mathsf{h}(y) \ |\!\}$$

$composeUnder(\theta, struct, \mathsf{h}(y)) = \{(\mathsf{l}_1, \{x \mapsto \mathsf{h}(y)\}), (\mathsf{l}_2, \varepsilon), (\mathsf{h}(\mathsf{l}_1), \{x \mapsto y\})\}$
The intruder knows three ways to compose $\mathsf{h}(y)$ (substitution = constraints for the recipe to work)

Recipes may generate different terms in $concr = \theta(struct)$!

Table of Contents

Introduction

Preliminaries

Modelling the intruder

Decision procedure
- Composition
- Analysis
- Relations between variables

Conclusion

Objective

- We know how to find recipes with *composition only*
- We want to *all* generable terms using only composition

$\longrightarrow$ Thus we need to perform *analysis steps*: decrypt messages, open signed messages, deserialise etc.

Analysis of a structural frame I

Input: $\theta, struct$
Output: $analyse(\theta, struct)$ = analysed frame + set of substitutions inconsistent with $concr \sim struct$

Analysis of a structural frame II

Analysis of a mapping l $\mapsto t \in struct$: $ana(t) = (K, FT)$ gives required keys and derivable terms

- If the analysis fails in $concr$, i.e. one key cannot be composed, no term can be derived. But if all keys can be composed in $struct$, the substitutions allowing this are inconsistent with $concr \sim struct$.
- If the analysis succeeds in $concr$, i.e. all keys can be composed, then it succeeds in $struct$ and derivable terms are added. The destructor attached to a term is used to create a shorthand.
- Repeat until no more new derivable terms can be added (frame saturation).

Analysis of a structural frame III

Example 1:

$$\theta = \{x \mapsto s, y \mapsto r, z \mapsto t, u \mapsto s\}$$

$$struct = \{\!| \, \mathsf{l}_1 \mapsto \mathsf{crypt}(\mathsf{pub}(x), y, z), \mathsf{l}_2 \mapsto \mathsf{pair}(\mathsf{priv}(u), \mathsf{pub}(u)) \, |\!\}$$

$$\begin{aligned} analyse(\theta, struct) = (\{\!| \, & \mathsf{l}_1 \mapsto \mathsf{crypt}(\mathsf{pub}(x), y, z), \\ & \mathsf{l}_2 \mapsto \mathsf{pair}(\mathsf{priv}(u), \mathsf{pub}(u)), \\ & \mathsf{proj}_1(\mathsf{l}_2) \mapsto \mathsf{priv}(u), \\ & \mathsf{proj}_2(\mathsf{l}_2) \mapsto \mathsf{pub}(u), \\ & \mathsf{dcrypt}(\mathsf{proj}_1(\mathsf{l}_2), \mathsf{l}_1) \mapsto z \, |\!\}, \{\}) \end{aligned}$$

Analysis of a structural frame IV

Example 2:

$$\theta = \{x \mapsto \mathsf{secret}, y \mapsto k'\}$$
$$struct = \{\!| \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, x), \mathsf{l}_2 \mapsto y |\!\}$$
$$analyse(\theta, struct) = (struct, \{\{y \mapsto k\}\})$$

The intruder cannot decrypt the message because analyis fails in $concr$. But they learn that y is not the key.

Table of Contents

Introduction

Preliminaries

Modelling the intruder

Decision procedure
- Composition
- Analysis
- Relations between variables

Conclusion

Relations between variables

1. Try to compose terms in $concr$ in different ways by calling $compose$:
   - Pairs of recipes for the same term must generate a unique term in $struct$ because $concr \sim struct$. $\longrightarrow$ find equalities ($x = t \wedge y = t' \ldots$).
2. Try to compose terms in $struct$ in different ways by calling $composeUnder$:
   - Check pairs (label, recipe) for the same term.
   - If they generate a unique term in $concr$ as well, nothing to deduce (it comes from $concr \sim struct$ and has been found previously).
   - If they generate different terms in $concr$, the substitution attached to the recipe is inconsistent with $concr \sim struct$. $\longrightarrow$ find inequalities ($x \neq \mathsf{h}(z) \vee y \neq 0 \ldots$)
3. ϕ = conjunction of equalities and inequalities = relations between variables

Relations between variables

Recall the illustration example:

$$\theta = \{x \mapsto 0, y \mapsto 1, z \mapsto 0\}$$
$$struct = \{\!| \, \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, x), \mathsf{l}_2 \mapsto \mathsf{scrypt}(k, y), \mathsf{l}_3 \mapsto \mathsf{scrypt}(k, z) \, |\!\}$$
$$concr = \{\!| \, \mathsf{l}_1 \mapsto \mathsf{scrypt}(k, 0), \mathsf{l}_2 \mapsto \mathsf{scrypt}(k, 1), \mathsf{l}_3 \mapsto \mathsf{scrypt}(k, 0) \, |\!\}$$

$$\alpha \equiv x, y, z \in \{0, 1\} \wedge x + y + z = 1 \qquad \beta \equiv MsgAna(\alpha, struct, \theta)$$

With our decision procedure:

- We analyse $struct$
- We generate $\phi \equiv x = z \wedge x \neq y$
- $\alpha \not\models \phi$: violation of privacy!

Relations between variables

ϕ is enough to decide privacy: we only need to check whether $\alpha \models \phi$

If $\alpha \models \phi$: the protocol is proven to respect privacy

If $\alpha \not\models \phi$: the protocol is not secure and we have a witness

Table of Contents

Introduction

Preliminaries

Modelling the intruder

Decision procedure

Conclusion

Recap

Current state:

- Standard approach to privacy has limitations
- (α, β)-privacy overcomes them
- We have a decision procedure for message-analysis problems (protocols without and with branching), already implemented in Haskell

Objectives for the future:

- Support more general algebraic theories (e.g. commutativity of exponentiation)
- Investigate fragments outside of message-analysis problems
- Apply to real protocols
- Develop further implementation

Alphabet

$\Sigma = \Sigma_f \uplus \Sigma_i \uplus \Sigma_r$

- Σ_f: free function symbols
- Σ_i: interpreted function symbols
- Σ_r: relation symbols

f^n: n-ary function c^0: constant

$\mathcal{V}$: variable symbols $\mathcal{T}_\Sigma(\mathcal{V})$: terms built from Σ and $\mathcal{V}$

Σ_{op}: cryptographic operators (constructors, destructors, verifiers)

Herbrand universe

$\approx$: congruence relation modelling algebraic properties

Example 1: for $\mathsf{f}^2 \in \Sigma_f, \forall x, y.\mathsf{f}(x, y) \approx \mathsf{f}(y, x)$

$\forall t \in \mathcal{T}_{\Sigma_f}, [\![t]\!]_{\approx} = \{t' \in \mathcal{T}_{\Sigma_f} \mid t \approx t'\}$: equivalence class

$U = \{[\![t]\!]_{\approx} \mid t \in \mathcal{T}_{\Sigma_f}\}$: Herbrand universe

Example 2: for $\Sigma_f = \{\mathsf{x}^0, \mathsf{s}^1\}$ and $\approx$ syntactic equality,
$U = \{\mathsf{x}, \mathsf{s}(\mathsf{x}), \mathsf{s}(\mathsf{s}(\mathsf{x})), \dots\}$

Interpretation

Σ_f-algebra $\mathcal{A} = \mathcal{T}_{\Sigma_f} / \approx$

To $\mathsf{f}^n \in \Sigma_f$, we associate $\mathsf{f}^{\mathcal{A}} : U^n \to U$ such that
$\mathsf{f}^{\mathcal{A}}([\![t_1]\!]_\approx, \ldots, [\![t_n]\!]_\approx) = [\![\mathsf{f}(t_1, \ldots, t_n)]\!]_\approx$

Interpretation $\mathcal{I}$ in Herbrand logic:

- for $\mathsf{f}^n \in \Sigma_f$ and $t_1, \ldots, t_n \in U$, $\mathcal{I}(\mathsf{f}(t_1, \ldots, t_n)) = \mathsf{f}^{\mathcal{A}}(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n))$
- for $f^n \in \Sigma_i$ and $t_1, \ldots, t_n \in U$, $\mathcal{I}(f[t_1, \ldots, t_n]) = \mathcal{I}(f)(\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n))$
- for $r^n \in \Sigma_r$, $\mathcal{I}(r) \subseteq U^n$
- for $x \in \mathcal{V}$, $\mathcal{I}(x) \in U$

Models

Interpretation $\mathcal{I}$ models formula ϕ is written $\mathcal{I} \models \phi$

$$
\begin{array}{lll}
\mathcal{I} \models s = t & \text{iff} & \mathcal{I}(s) = \mathcal{I}(t) \\
\mathcal{I} \models r(t_1, \ldots, t_n) & \text{iff} & (\mathcal{I}(t_1), \ldots, \mathcal{I}(t_n)) \in \mathcal{I}(r) \\
\mathcal{I} \models \neg\phi & \text{iff} & \text{not } \mathcal{I} \models \phi \\
\mathcal{I} \models \phi \wedge \psi & \text{iff} & \mathcal{I} \models \phi \text{ and } \mathcal{I} \models \psi \\
\mathcal{I} \models \exists x.\phi & \text{iff} & \text{there is } c \in U \text{ such that } \mathcal{I}\{x \mapsto c\} \models \phi
\end{array}
$$

$Sat(\phi)$: ϕ has a model

Interesting consequence

$\alpha \in \mathcal{L}_{\Sigma_0}(\mathcal{V})$: payload formula

$\beta \in \mathcal{L}_{\Sigma}(\mathcal{V})$: technical information formula

$\beta \models \alpha$ and $fv(\alpha) = fv(\beta)$ and both α and β are consistent

$\alpha' \in \mathcal{L}_{\Sigma_0}(fv(\alpha))$ is an interesting consequence of β (with respect to α) if $\beta \models \alpha'$ but $\alpha \not\models \alpha'$

We say that β respects the privacy of α if it has no interesting consequences, and that β violates the privacy of α otherwise

Axioms I

$$
\begin{aligned}
\phi_{frame}(F) \quad \equiv \quad & (\forall x. gen_F(x) \iff (x \in \{\mathsf{l}_1, \dots, \mathsf{l}_k\} \vee \\
& \bigvee_{f^n \in \Sigma_{pub}} \exists x_1 \dots x_n. \\
& \quad x = f(x_1, \dots, x_n) \wedge gen_F(x_1) \wedge \dots \wedge gen(x_n))) \\
& \wedge \\
& (kn_F[\mathsf{l}_1] = t_1 \wedge \dots \wedge kn_F[\mathsf{l}_k] = t_k) \\
& \wedge \\
& \bigwedge_{f^n \in \Sigma_{pub}} (\forall x_1 \dots x_n. gen_F(x_1) \wedge \dots \wedge gen(x_n) \implies \\
& \quad kn_F[f(x_1, \dots, x_n)] = f(kn_F[x_1], \dots, kn_F[x_n]))
\end{aligned}
$$

Axioms II

$$\begin{aligned}\phi_{F_1 \sim F_2} \quad \equiv \quad & (\forall x. gen_{F_1}(x) \iff gen_{F_2}(x)) \\ & \wedge \\ & (\forall x, y. gen_{F_1}(x) \wedge gen_{F_1}(y) \implies \\ & \quad (kn_{F_1}[x] = kn_{F_1}[y] \iff kn_{F_2}[x] = kn_{F_2}[y]))\end{aligned}$$

Static equivalence

$$F_1 \sim F_2 \text{ iff } Sat(\phi_{frame}(F_1) \wedge \phi_{frame}(F_2) \wedge \phi_{F_1 \sim F_2})$$

Message-analysis problem

Theorem

Let α be combinatoric, $\Theta = \{\theta_1, \ldots, \theta_n\}$ be the models of α, and $\beta \equiv MsgAna(\alpha, F, \theta_1)$ for some $\theta_1 \in \Theta$.

Then, we have that (α, β)-privacy holds iff $\theta_1(F) \sim \ldots \sim \theta_n(F)$.

We can look at static equivalence of frames to decide privacy for such problems

Unification

Unification is a standard problem. We can call an algorithm unify returning a most general unifier for a set of equalities.

Example: $\mathit{unify}(\{(f(x, y), f(0, g(z)))\}) = \{x \mapsto 0, y \mapsto g(z)\}$

unify can be used to find relations between variables in the messages $(x = 0 \wedge y = g(z))$

Algorithms

Algorithm 1: Composition in a ground frame

$compose(concr, t) =$

let $R = \{\mathsf{l} \mid \mathsf{l} \mapsto t \in concr\}$ **in**

if $t = f(t_1, \ldots, t_n)$ *and* f *is public* **then**

$$R \cup \{f(r_1, \ldots, r_n) \mid r_1 \in compose(concr, t_1), \ldots, r_n \in compose(concr, t_n)\}$$

else

R

Algorithms

Let $concr$ be a ground frame and $t \in \mathcal{T}_{\Sigma}$.

- The call $compose(concr, t)$ terminates.
- $\forall r \in compose(concr, t), concr\{\!|\, r \,|\!\} = t$
- Let r be a recipe containing only constructors such that $concr\{\!|\, r \,|\!\} = t$. Then $r \in compose(concr, t)$.

Algorithms

Algorithm 2: Composition in a structural frame

$composeUnder(\theta, struct, t) =$

let $RU = \{(\mathsf{l}, \sigma) \mid \mathsf{l} \mapsto t' \in struct, \sigma = unify(t = t')\}$ **in**

if $t = x$ *and* x *has a public range* **then**

$RU \cup \{(\theta(x), \{x \mapsto \theta(x)\})\}$

else if $t = f(t_1, \ldots, t_n)$ *and* f *is public* **then**

$$RU \cup \{(f(r_1, \ldots, r_n), \sigma) \mid (r_1, \sigma_1) \in composeUnder(\theta, struct, t_1), \ldots, (r_n, \sigma_n) \in composeUnder(\theta, struct, t_n), \sigma = unify(\sigma_1, \ldots, \sigma_n)\}$$

else

RU

Algorithms

Let θ be a substitution, $struct$ be a frame and $t \in \mathcal{T}_{\Sigma}(\mathcal{V})$.

- The call $composeUnder(\theta, struct, t)$ terminates.
- $\forall (r, \sigma) \in composeUnder(\theta, struct, t), \sigma(struct\{\!| r |\!\}) = \sigma(t)$
- Let r be a recipe and τ be a substitution such that $\tau(struct\{\!| r |\!\}) = \tau(t)$ and r contains only constructors. Then $\exists \sigma, (r, \sigma) \in composeUnder(\theta, struct, t)$ and $\sigma \lesssim \tau$.

Example

$$ana(t) = \begin{cases} (\{\mathsf{priv}(s)\}, \{(\mathsf{dcrypt}, t')\}) & \text{if } t = \mathsf{crypt}(\mathsf{pub}(s), r, t') \\ (\{k\}, \{(\mathsf{dscrypt}, t')\}) & \text{if } t = \mathsf{scrypt}(k, t') \\ (\{\}, \{(\mathsf{retrieve}, t')\}) & \text{if } t = \mathsf{sign}(p', t') \\ (\{\}, \{(\mathsf{proj}_1, t_1), (\mathsf{proj}_2, t_2)\}) & \text{if } t = \mathsf{pair}(t_1, t_2) \\ (\{\}, \{\}) & \text{otherwise} \end{cases}$$

$$ana(\mathsf{scrypt}(k, \mathsf{pair}(t_1, t_2)) = (\{k\}, \{(\mathsf{dscrypt}, \mathsf{pair}(t_1, t_2))\}$$

Algorithms

Algorithm 3: Analysis of a structural frame (wrapper)

$analyse(\theta, struct) =$

$\quad analyseRec(\theta, struct, \{\!|\ |\!\}, \{\!|\ |\!\}, \{\})$

Algorithms

Let θ be a substitution and $struct$ be a frame.

- The call $analyse(\theta, struct)$ terminates.
- $\forall r, struct_{ana}\{\!| r |\!\} \approx struct\{\!| r |\!\}$, where $(struct_{ana}, E) = analyse(\theta, struct)$.
- For every recipe r, there exists a recipe r' containing only constructors such that $struct_{ana}\{\!| r' |\!\} \approx struct\{\!| r |\!\}$, where $(struct_{ana}, E) = analyse(\theta, struct)$.

Algorithms

Algorithm 4: Analysis of a structural frame (recursive)

$analyseRec(\theta, N, H, D, E) =$

if $N = \{\!|\ |\!\}$ **then**

  $(H \cup D, E)$

else

  let $\{\!| \mathsf{l} \mapsto t |\!\} \cup LT = N$

    $(K, FT) = ana(t)$

    $\{k_1, \ldots, k_n\} = K$

    $FT_{new} = \{(f, t') \in FT \mid \forall r, r \mapsto t' \notin D\}$ **in**

  if $FT_{new} = \{\}$ **then**

    $analyseRec(\theta, FT, H, \{\!| \mathsf{l} \mapsto t |\!\} \cup D, E)$

  else

    let $struct = N \cup H \cup D$

      $concr = \theta(struct)$ **in**

    if $\{\} \in \{compose(concr, \theta(k)) \mid k \in K\}$ **then**

      let $E_{new} = \{\sigma \mid (r_1, \sigma_1) \in composeUnder(\theta, struct, k_1),$

        $\ldots,$

        $(r_n, \sigma_n) \in composeUnder(\theta, struct, k_n),$

        $\sigma = unify(\sigma_1, \ldots, \sigma_n)\}$ **in**

      $analyseRec(\theta, LT, \{\!| \mathsf{l} \mapsto t |\!\} \cup H, D, E \cup E_{new})$

    else

      let $LT_{new} = \{\!|\ f(r_1, \ldots, r_n, \mathsf{l}) \mapsto t' \mid$

        $(f, t') \in FT_{new},$

        $pick\ r_1 \in compose(concr, \theta(k_1)),$

        $\ldots$

        $pick\ r_n \in compose(concr, \theta(k_n))\ |\!\}$

        $\cup \{\!|\ r \mapsto k \mid k \in K,$

          $pick\ r \in compose(concr, \theta(k)),$

          $\forall t', r \mapsto t' \notin struct\ |\!\}$ **in**

      $analyseRec(\theta, LT_{new} \cup LT \cup H, \{\!|\ |\!\}, \{\!| \mathsf{l} \mapsto t |\!\} \cup D, E)$

Algorithms

Algorithm 5: Relations between variables

$$
\begin{aligned}
&\mathit{findRelations}(\theta, \mathit{struct}) = \\
&\quad \textbf{let } (\mathit{struct}_{ana}, E) = \mathit{analyse}(\theta, \mathit{struct}) \\
&\qquad \mathit{concr}_{ana} = \theta(\mathit{struct}_{ana}) \\
&\qquad \mathit{pairs} = \mathit{pairsEcs}(\{\mathit{compose}(\mathit{concr}_{ana}, t) \mid \exists \mathsf{l}, \mathsf{l} \mapsto t \in \mathit{concr}_{ana}\}) \\
&\qquad \mathit{eqs} = \{(\mathit{struct}_{ana}\{\!| r_1 |\!\}, \mathit{struct}_{ana}\{\!| r_2 |\!\}) \mid (r_1, r_2) \in \mathit{pairs}\} \\
&\qquad \mathit{ineqs} = E \cup \{\sigma' \mid \mathsf{l} \mapsto t \in \mathit{struct}_{ana}, \\
&\qquad\qquad\qquad (r, \sigma') \in \mathit{composeUnder}(\theta, \mathit{struct}_{ana}, t), \\
&\qquad\qquad\qquad \mathsf{l} \neq r, \\
&\qquad\qquad\qquad \mathit{concr}_{ana}\{\!| \mathsf{l} |\!\} \neq \mathit{concr}_{ana}\{\!| r |\!\}\} \\
&\qquad \sigma = \mathit{unify}(\mathit{eqs}) \textbf{ in} \\
&\quad \sigma \wedge \bigwedge_{\tau \in \mathit{ineqs}} \neg \tau
\end{aligned}
$$

Different problem

- 1 concrete frame $concr$;
- n structural frames $struct_1, \ldots, struct_n$;
- Only one correct $struct_i$: $concr = \theta(struct_1)$

Lifting results

Approach of the extended procedure:

1. Analyse each $struct_i$ separately.
2. Rule out possibilities where $struct_i \not\sim \theta(struct_1)$.
3. Generate ϕ_i (relations between variables) for the remaining possibilities.
4. Check if $\phi_1 \wedge \cdots \wedge \phi_n$ is a violation of privacy.