Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 168 (2007) 91–107
www.elsevier.com/locate/entcs

Cryptographic Pattern Matching

Christoffer Rosenkilde Nielsen[1] Flemming Nielson[2]
Hanne Riis Nielson[3]

Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, bldg. 321, DK-2800 Kongens Lyngby, Denmark

Abstract

We construct a language extension for process calculi for modelling the exchange of cryptographically composed data. More specifically, we devise a succinct syntax for terms and patterns that captures the intention behind perfect cryptography. The proposed language extension is independent of the choice of process calculus and is applicable to any calculus that supports exchange of data. Initially we restrict the model to symmetric cryptography, but we also show how it can be extended with support for asymmetric encryption and digital signatures.

Keywords: process calculi, pattern matching, cryptography, formal methods, protocols

1 Introduction

The modelling of security protocols often relies on process calculi. As a common method for ensuring security in a system is the application of cryptography, several process calculi have emerged which incorporates cryptography in the design, e.g. LySA [3,4] and the Spi-calculus [1]. However, the modelling of cryptography and the underlying communication model are orthogonal factors in the resulting system, and this motivates an independent development of these components.

In this paper we shall design a language extension for process calculi that allows for modelling the use of cryptographic operations for securing the exchange of data. Specifically, we develop a term language with pattern matching that captures the intention behind perfect cryptography in an intuitive and succinct manner. The design is independent of the underlying communication model, and the language extension can be applied to any process calculus that supports exchange of data.

[1] Email: crn@imm.dtu.dk
[2] Email: nielson@imm.dtu.dk
[3] Email: riis@imm.dtu.dk

doi:10.1016/j.entcs.2006.08.024

We shall develop the language extension in two phases. In Section 2 we present the formal foundation of our model for extending a calculus with support for cryptography. Initially we restrict our attention to symmetric encryption, as this suffices for introducing the general concepts and eases readability. However, the model is easily extendable with other cryptographic primitives, and in Section 3 we shall see how to deal with asymmetric cryptography. It turns out that asymmetric cryptography introduces new requirements to the model that the language extension must cater for. Essentially, asymmetric cryptography allows for a new scenario, namely that a principal can decrypt a message without being able to recreate the message by encryption.

To illustrate the use of the pattern matching primitive we show in Section 4 how to formulate different security protocols and in Section 5 we shall recapitulate and reflect upon our contribution. Finally, in Appendix A we show the validity of our language extension with respect to the principles of perfect cryptography.

2 Design

2.1 Syntax

The basic building blocks used for modelling cryptography are terms and patterns. The syntax for these are listed in Table 1 where $\mathcal{N}$ and $\mathcal{V}$ denote the disjoint sets of names and variables, respectively.

t	::=	n	Name ($n \in \mathcal{N}$)
	|	x	Variable ($x \in \mathcal{V}$)
	|	$\mathsf{T}(t_1, \cdots, t_k)$	Tuple
	|	$\mathsf{E}_{t_0}(t)$	Shared Key Encryption
p	::=	$t \triangleleft [x_1, \cdots, x_k]$	Pattern

Table 1
Syntax for terms and patterns.

The syntax for terms bears similarities to LySA. Names are used for describing all basic elements such as text, nonces, symmetric keys or time stamps. Variables can be mapped during execution, and may hold both names and composite terms. The tuple construct $\mathsf{T}(t_1, \cdots, t_k)$ is used for concatenation of terms and finally the construct $\mathsf{E}_{t_0}(t)$ denotes symmetric encryption of a term t using the key t_0. In the following we shall often refer to a closed term, that is a term with no variables, as a *value* ($v \in \mathsf{Val}$).

The other basic building block is patterns. As mentioned already, patterns are used to match on terms. Hence the syntax for patterns is identical to the syntax for terms, except that a pattern includes a list of the variables that should be mapped within the term.

Example 2.1 Assume the existence of a matching operator $\triangleright$ for matching a value v against a pattern p, written $v \triangleright p$. Such a construct should be used upon input or decryption in a process calculus, and the general idea is then that a process $v \triangleright p$ in P first verifies that v matches p; if this is the case the continuation process P

is updated with any newly mapped variables, and executed, otherwise the execution will be blocked.

The matching $\mathsf{T}(n,m) \rhd \mathsf{T}(y,m) \lhd [y]$ in P should succeed because the tuple $\mathsf{T}(n,m)$ matches $\mathsf{T}(y,m)$. This results in the mapping of the variable y to the name n in the continuation process P. The matching $\mathsf{T}(n,m) \rhd \mathsf{T}(y,y) \lhd [y]$ in P on the other hand would not succeed as both n and m cannot match the pattern y, and thus further execution is garbled.

2.2 Semantics

Having presented the syntax of the model we shall now define a semantics. We rely on reduction semantics, and assume that a similar semantics exists for the calculus to be extended.

As reduction semantics is only concerned with closed processes, it is clear that the semantics of terms are merely values. The semantics of pattern matching on the other hand, must record all new mappings of variables. Formally, the semantics needs to produce an environment

$$\theta : \mathcal{V} \to \mathsf{Val}$$

that maps the variables in the matching to their respective value. We shall write $[\,]$ for the empty mapping and $\theta[x \mapsto v]$ for the mapping that is like θ except it maps x to v. Furthermore we define an equality operator for mappings $\doteq$ as

$$\theta_1 \doteq \theta_2 \text{ iff } \forall x \in (dom(\theta_1) \cap dom(\theta_2)) \quad : \quad \theta_1(x) = \theta_2(x)$$

The equality operator $\doteq$ ensures that if a variable is mapped in both environments then these mappings are equal, i.e. the variable is mapped to the same value in both environments. Thus we can unambiguously unify two environments, $\theta_1 \cup \theta_2$, whenever $\theta_1 \doteq \theta_2$.

The judgement for the semantics of pattern matching takes the form $\vdash v \rhd p : \theta$ and states that the value v correctly matches the pattern p, giving rise to the variable mappings θ. This is captured by the definition in Table 2. The definition simply requires an auxiliary judgement for decomposition $\mathcal{X} \vdash v \rhd t : \theta$ to be satisfied, which ensures that the value v matches the term t giving rise to the variable mappings in θ, assuming the variables to be mapped are all included the set $\mathcal{X}$. The semantics assumes that all sub-patterns are matched in parallel and ensures that possible multiple mappings of variables are equal using the $\doteq$ operator. Whenever a value v is matched against a variable x we use the rule (Bind), where the resulting environment is the mapping of x to v. As the semantics is defined as a reduction semantics, we know that all variables that have been mapped prior to the pattern matching are already replaced by their corresponding values. Thus the rule (Bind) only requires the variable x to belong to the list of variables to be defined in the matching, $\mathcal{X}$. Symmetric decryption (SDec) simply requires both the key and the content to match the pattern, and tuples (Tup) require all sub-patterns to match.

The resulting environment θ of a pattern matching is supposed to be used to update a possible continuation process with the new mappings of variables that the matching resulted in.

$$\text{(Match)}\quad \frac{\mathcal{X} \vdash v \rhd t : \theta}{\vdash v \rhd t \lhd \mathcal{X} : \theta}$$

$$\text{(Name)}\quad \mathcal{X} \vdash n \rhd n : [\,] \qquad \text{(Bind)}\quad \frac{x \in \mathcal{X}}{\mathcal{X} \vdash v \rhd x : [x \mapsto v]}$$

$$\text{(SDec)}\quad \frac{\mathcal{X} \vdash v_0 \rhd t_0 : \theta_0 \quad \mathcal{X} \vdash v \rhd t : \theta \quad \theta_0 \doteq \theta}{\mathcal{X} \vdash \mathsf{E}_{v_0}(v) \rhd \mathsf{E}_{t_0}(t) : \theta_0 \cup \theta}$$

$$\text{(Tup)}\quad \frac{\mathcal{X} \vdash v_1 \rhd t_1 : \theta_1 \quad \cdots \quad \mathcal{X} \vdash v_k \rhd t_k : \theta_k}{\mathcal{X} \vdash \mathsf{T}(v_1, \cdots, v_k) \rhd \mathsf{T}(t_1, \cdots, t_k) : \theta_1 \cup \cdots \cup \theta_k} \quad \text{if } \forall i, j : \theta_i \doteq \theta_j$$

Table 2
Semantics of pattern matching; $\vdash v \rhd p : \theta$ and $\mathcal{X} \vdash v \rhd t : \theta$.

2.3 Well-formedness

The syntax and semantics of our language extension does not enforce the usual assumption of perfect cryptography. Matching on a pattern such as $\mathsf{E}_x(y) \lhd [x, y]$ obviously violates this assumption, as it allows for learning x based only on an encrypted message. This design choice has been made to make the model as flexible as possible, one may want to encode the possibility of a principal to guess certain keys or encrypted contents in order to analyse different attack scenarios. Perfect cryptography is only one such scenario, although the most commonly used one, and thus we shall design a well-formedness condition for enforcing this assumption, hereby allowing the protocol analyst to choose when this criteria should apply.

In order to design a well-formedness requirement we shall assume the existence of the sets of defined names $\mathcal{N}$ and defined variables $\mathcal{V}$. Each element in these sets are supposed to be defined prior to the pattern matching, and thus for a given term we shall simply require all names and variables used within the term to belong to the sets $\mathcal{N}$ and $\mathcal{V}$, respectively. For this we shall assume the existence of the function for finding free names fn and free variables fv in a term, defined in the usual manner. The resulting well-formedness condition for terms is given as judgement $(\mathcal{N}, \mathcal{V}) \vdash t$ in Table 3.

The requirement for patterns is more subtle as we must ensure that the patterns enforce the rules of perfect cryptography. We approach this challenge by first noticing that perfect cryptography is violated whenever we cannot sort the list $\mathcal{X}$ of variables to be learnt, such that the first variable can be learnt without knowledge of the remaining variables, learning the second variable requires at most knowledge of the first variable, etc. One example of this could be the pattern $\mathsf{T}(\mathsf{E}_y(x), \mathsf{E}_x(y)) \lhd [x, y]$, where neither x can be learnt without knowledge of y nor y without knowledge of x. This means that we can require the list of variables to be defined $\mathcal{X}$ to be ordered, without limiting the model.

Now, assuming an ordered list of variables, we must have a method of detecting whether the variables one by one can be learnt correctly from in the pattern. We achieve this goal by introducing an auxiliary judgement $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$ for validating that the variable x can be learnt from the term t strictly following the cryptographic rules and given the set of known names $\mathcal{N}$ and variables $\mathcal{V}$. The judgement is defined straightforwardly, the variable x is obviously learnt legally from the term x. If the term is composite, a recursive search is initiated; in a tuple at least one of the sub-

terms must provide knowledge of x, and a cryptographic construction should only allow x to be learnt from the contents, if the key consists of purely known names and variables.

$$(\mathsf{WT}) \quad \frac{\mathsf{fn}(t) \subseteq \mathcal{N} \quad \mathsf{fv}(t) \subseteq \mathcal{V}}{(\mathcal{N}, \mathcal{V}) \vdash t}$$

$$(\mathsf{WPDef}) \quad \frac{x \notin \mathcal{V} \quad (\mathcal{N}, \mathcal{V}) \vdash t \sim x \quad (\mathcal{N}, \mathcal{V} \cup \{x\}) \vdash t \lhd \mathcal{X}}{(\mathcal{N}, \mathcal{V}) \vdash t \lhd (x :: \mathcal{X})}$$

$$(\mathsf{WPEmp}) \quad \frac{(\mathcal{N}, \mathcal{V}) \vdash t}{(\mathcal{N}, \mathcal{V}) \vdash t \lhd [\,]}$$

$$(\mathsf{WH1}) \quad (\mathcal{N}, \mathcal{V}) \vdash x \sim x \qquad (\mathsf{WH2}) \quad \frac{\mathsf{fv}(t_0) \subseteq \mathcal{V} \quad \mathsf{fn}(t_0) \subseteq \mathcal{N} \quad (\mathcal{N}, \mathcal{V}) \vdash t \sim x}{(\mathcal{N}, \mathcal{V}) \vdash \mathsf{E}_{t_0}(t) \sim x}$$

$$(\mathsf{WH3}) \quad \frac{(\mathcal{N}, \mathcal{V}) \vdash t_i \sim x}{(\mathcal{N}, \mathcal{V}) \vdash \mathsf{T}(t_1, \cdots, t_k) \sim x} \quad 1 \leq i \leq k$$

Table 3
Well-formedness of terms and patterns; $(\mathcal{N}, \mathcal{V}) \vdash t$, $(\mathcal{N}, \mathcal{V}) \vdash p$ and $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$.

The auxiliary judgement allows for defining a well-formedness condition for patterns. This condition relies on the set of defined names $\mathcal{N}$, the set of defined variables $\mathcal{V}$ and the ordered list of the variables that should be defined in the pattern $\mathcal{X}$. If $\mathcal{X}$ is not empty the rule (WPDef) applies, where the first element x of $\mathcal{X}$, i.e. the next variable to be defined, is checked to be a fresh variable $x \notin \mathcal{V}$ and then validated that it can be learnt correctly from the term by $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$. If this is fulfilled, the variable x can safely be added to the set of known variables and the next variable in the list can be checked. At some point the list $\mathcal{X}$ will be empty, in this case the rule (WPEmp) applies which merely establishes that t is a well-formed term given the sets of now known variables $\mathcal{V}$ and names $\mathcal{N}$.

In order to fully understand the need for and the workings of the well-formedness condition, consider the following example.

Example 2.2 Consider these entirely legal patterns:

(1) $\mathsf{E}_K(x) \lhd [x]$ where $K \in \mathcal{N}$
(2) $\mathsf{E}_x(y) \lhd [y]$ where $x \in \mathcal{V}$
(3) $\mathsf{T}(\mathsf{E}_y(x), \mathsf{E}_z(y), z) \lhd [z, y, x]$

In (1) the new variable x is learnt by decryption with K, an already known name. Similarly in (2) y is learnt using the prior knowledge of x to decrypt the encryption. In (3) z can be learnt from the third element of the tuple, then the second part of the tuple can be decrypted using z hereby learning y, and finally from the first part of the tuple x can be learnt using y.

All of the above patterns follow the cryptographic rules and are correctly accepted by our well-formedness condition. But now see how small alterations of the patterns above change them into patterns with unwanted properties that are correctly disqualified by the well-formedness condition.

(1′) $\mathsf{E}_x(K) \lhd [x]$ where $K \in \mathcal{N}$
(1″) $\mathsf{E}_K(K) \lhd [x]$ where $K \in \mathcal{N}$
(2′) $\mathsf{E}_x(y) \lhd [x, y]$
(3′) $\mathsf{T}(\mathsf{E}_y(x), \mathsf{E}_x(y)) \lhd [x, y]$
(3″) $\mathsf{T}(\mathsf{E}_y(x), \mathsf{E}_z(y), z) \lhd [x, y, z]$

The pattern $(1')$ is illegal as either $x \in \mathcal{V}$ and the definition is a renaming which is not allowed, or else x is not previously defined, meaning that x is learnt from a key, thus violating perfect cryptography and not satisfying $(\mathcal{V}) \vdash p \sim x$. In pattern $(1'')$ we simply can't learn x from the pattern and in $(2')$ x is again only learnt based on a key. In $(3')$ x cannot be learnt without prior knowledge of y (and y cannot be learnt without knowledge of x) and in $(3'')$ x cannot be learnt without prior knowledge of y (reverse definition order of (3)).

3 Extending the Model

We shall now show how the model presented in Section 2, hereafter referred to as the basic model, can be extended with different cryptographic operations. We do this by extending it with asymmetric cryptography, as this extension introduces some interesting new aspects that our model must cater for.

Extending the model with asymmetric cryptography opens for scenarios where a principal can decompose a term but cannot recreate this term afterwards. This is exemplified by digital signatures, where a principal that receives a signed value may be able to learn the signed content by decrypting it with the signer's public key, but cannot reproduce this signed value itself because it does not know the signer's private key. This introduces complications when modelling protocols where a principal receives a value, verifies that it is signed by the expected principal, and forwards it to a receiver. To cater for this, we shall extend our model with a capability for enforcing restrictions on the values a variable may be mapped to. With such an extension, the example mentioned here could be modelled by letting the principal bind the incoming value to a variable x, if and only if it is signed by the expected principal, and then forward the value that x is mapped to, to the receiver.

3.1 Syntax

As already described, we shall extend the syntax of our model to allow restrictions of the mappings of variables. We do this by introducing a new syntactic element s which we call a *sieve*. Sieves express the restrictions under which we allow a variable to be mapped, and we write $x \mapsto s$ for defining the variable x which is allowed to be mapped to any value v that matches the sieve s.

The resulting syntax for terms, sieves and patterns is given in Table 4. Here we annotate names with a tag $\tau \in \{\epsilon, +, -\}$ for modelling whether the name is a public key n^+ or a private key n^- as part of a key pair, or if it is just a regular name n^ϵ. Public key encryption is modelled using the construct $\mathsf{P}_{t_0}(t)$, and as usually we shall model digital signatures by encryption using a private key.

Sieves are supposed to match on terms and thus the syntax is merely terms extended with a wildcard $\star$ for a sieve that matches everything, i.e. variables defined with the sieve $\star$ correspond to the variables of the basic model. Notice that sieves allow for both recursive variable definitions and for defining variables based on the mappings to other variables. Recursive variable definitions such as in the pattern $x \triangleleft$

t	$::=$	n^τ	Name ($n^\tau \in \mathcal{N}, \tau \in \{\epsilon, +, -\}$)
	$\mid$	x	Variable ($x \in \mathcal{V}$)
	$\mid$	$\mathsf{T}(t_1, \cdots, t_k)$	Tuple
	$\mid$	$\mathsf{E}_{t_0}(t)$	Shared key encryption
	$\mid$	$\mathsf{P}_{t_0}(t)$	Public key encryption
s	$::=$	t	Restrictive sieve
	$\mid$	$\star$	Non − restrictive sieve
p	$::=$	$t \triangleleft [x_1 \mapsto s_1, \cdots, x_k \mapsto s_k]$	Pattern

Table 4
Revised syntax.

$[x \mapsto \mathsf{T}(x, x)]$ would in this pattern matching context never match any values, as we do not allow remapping of variables. Thus such sieves should be prohibited by a well-formedness condition. Defining variables based on other variables however, provides a useful tool when describing protocols, e.g. the pattern $x \triangleleft [x \mapsto \mathsf{P}_{n^-}(y), y \mapsto \star]$ allows for both binding a signed value and the value itself to some variables, in one pattern matching.

The defining list of variables $[x_1 \mapsto s_1, \cdots, x_k \mapsto s_k]$ in the pattern, can be seen as an environment Γ, which maps the variables to be defined to their respective sieve:

$$\Gamma : \mathcal{V} \to \{s \mid s \text{ defined in Table 4}\}$$

This allows for a simpler definition of the semantics.

3.2 Semantics

The judgement for the semantics still takes the form $\vdash v \rhd p : \theta$ but requires an auxiliary judgement on the form $\Gamma \vdash v \rhd t : \theta$ to be satisfied, which states that given the environment Γ that maps the variables to their corresponding sieve, then the value v matches the term t, giving rise to the variable mappings of θ. Both of these judgements are defined in Table 5.

The semantics is closely related to the semantics for the basic model, but the auxiliary judgement includes a few more rules. The rules (RPDec) and (RSign) for asymmetric decryption and validation of digital signatures are similar to symmetric decryption, except we ensure that the keys are tagged as required and that they belong to a key pair. When matching a value v against a variable x one of the rules (RBind1) and (RBind2) applies, depending on x's sieve. In the case of a restrictive sieve t the rule (RBind1) is used, this requires the value v to match t, and provided this matching gives rise to the environment θ then θ is updated with the mapping of x to v. In the case of a non-restrictive sieve $\star$ the rule (RBind2) applies, which merely returns the environment that maps x to v.

3.3 Well-formedness

In Table 6 we define the well-formedness condition for the language extension with asymmetric cryptography. The well-formedness condition of terms is analogous to the one in the basic model.

$$(\mathsf{RMatch})\quad \frac{\Gamma \vdash v \rhd t : \theta}{\vdash v \rhd t \lhd \Gamma : \theta}$$

$$(\mathsf{RName})\quad \Gamma \vdash n^{\tau} \rhd n^{\tau} : [\,] \qquad (\mathsf{RSDec})\quad \frac{\Gamma \vdash v_0 \rhd t_0 : \theta_0 \quad \Gamma \vdash v \rhd t : \theta \quad \theta_0 \doteq \theta}{\Gamma \vdash \mathsf{E}_{v_0}(v) \rhd \mathsf{E}_{t_0}(t) : \theta_0 \cup \theta}$$

$$(\mathsf{RBind1})\quad \frac{\Gamma(x) = t \quad \Gamma \vdash v \rhd t : \theta}{\Gamma \vdash v \rhd x : \theta[x \mapsto v]} \qquad (\mathsf{RPDec})\quad \frac{\Gamma \vdash n^{-} \rhd t_0 : \theta_0 \quad \Gamma \vdash v \rhd t : \theta \quad \theta_0 \doteq \theta}{\Gamma \vdash \mathsf{P}_{n^{+}}(v) \rhd \mathsf{P}_{t_0}(t) : \theta_0 \cup \theta}$$

$$(\mathsf{RBind2})\quad \frac{\Gamma(x) = \star}{\Gamma \vdash v \rhd x : [x \mapsto v]} \qquad (\mathsf{RSign})\quad \frac{\Gamma \vdash n^{+} \rhd t_0 : \theta_0 \quad \Gamma \vdash v \rhd t : \theta \quad \theta_0 \doteq \theta}{\Gamma \vdash \mathsf{P}_{n^{-}}(v) \rhd \mathsf{P}_{t_0}(t) : \theta_0 \cup \theta}$$

$$(\mathsf{RTup})\quad \frac{\Gamma \vdash v_1 \rhd t_1 : \theta_1 \quad \cdots \quad \Gamma \vdash v_k \rhd t_k : \theta_k}{\Gamma \vdash \mathsf{T}(v_1, \cdots, v_k) \rhd \mathsf{T}(t_1, \cdots, t_k) : \theta_1 \cup \cdots \cup \theta_k} \quad \text{if } \forall i, j : \theta_i \doteq \theta_j$$

Table 5
Revised semantics of pattern matching; $\vdash v \rhd p : \theta$ and $\Gamma \vdash v \rhd t : \theta$.

Well-formedness of patterns still requires an auxiliary judgement $(\mathcal{N}, \mathcal{V}) \vdash p \sim x$ for determining if a variable x can be learnt correctly from a term t given knowledge of the names $\mathcal{N}$ and variables $\mathcal{V}$, this is defined defined analogous to the auxiliary judgement in the basic model.

$$(\mathsf{RWT})\quad \frac{\mathsf{fn}(t) \subseteq \mathcal{N} \quad \mathsf{fv}(t) \subseteq \mathcal{V}}{(\mathcal{N}, \mathcal{V}) \vdash t}$$

$$(\mathsf{RWDef1})\quad \frac{x \notin \mathcal{V} \quad x \notin dom(\Gamma) \quad (\mathcal{N}, \mathcal{V}) \vdash t[t'/x] \lhd \Gamma}{(\mathcal{N}, \mathcal{V}) \vdash t \lhd ((x \mapsto t') :: \Gamma)}$$

$$(\mathsf{RWDef2})\quad \frac{x \notin \mathcal{V} \quad (\mathcal{N}, \mathcal{V}) \vdash t \sim x \quad (\mathcal{N}, \mathcal{V} \cup \{x\}) \vdash t \lhd \Gamma}{(\mathcal{N}, \mathcal{V}) \vdash t \lhd ((x \mapsto \star) :: \Gamma)}$$

$$(\mathsf{RWEmp})\quad \frac{(\mathcal{N}, \mathcal{V}) \vdash t}{(\mathcal{N}, \mathcal{V}) \vdash t \lhd [\,]}$$

$$(\mathsf{RWH1})\quad (\mathcal{N}, \mathcal{V}) \vdash x \sim x$$

$$(\mathsf{RWH2})\quad \frac{\mathsf{fv}(t_0) \subseteq \mathcal{V} \quad \mathsf{fn}(t_0) \subseteq \mathcal{N} \quad (\mathcal{N}, \mathcal{V}) \vdash t \sim x}{(\mathcal{N}, \mathcal{V}) \vdash \mathsf{E}_{t_0}(t) \sim x}$$

$$(\mathsf{RWH3})\quad \frac{\mathsf{fv}(t_0) \subseteq \mathcal{V} \quad \mathsf{fn}(t_0) \subseteq \mathcal{N} \quad (\mathcal{N}, \mathcal{V}) \vdash t \sim x}{(\mathcal{N}, \mathcal{V}) \vdash \mathsf{P}_{t_0}(t) \sim x}$$

$$(\mathsf{RWH4})\quad \frac{(\mathcal{N}, \mathcal{V}) \vdash t_i \sim x}{(\mathcal{N}, \mathcal{V}) \vdash \mathsf{T}(t_1, \cdots, t_k) \sim x} \quad 1 \leq i \leq k$$

Table 6
Revised well-formedness of terms and patterns; $(\mathcal{N}, \mathcal{V}) \vdash t$, $(\mathcal{N}, \mathcal{V}) \vdash p$ and $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$.

The well-formedness of patterns itself is on the form $(\mathcal{N}, \mathcal{V}) \vdash p$ stating that the pattern p is well-formed under the assumption of known sets of names $\mathcal{N}$ and variables $\mathcal{V}$. This is captured by three rules. The rule (RWDef1) shows that a variable defined with a restrictive sieve t' is checked to be fresh $x \notin \mathcal{V}$ and also required not to be defined repeatedly $x \notin dom(\Gamma)$; if this is fulfilled the rule merely replace x by its corresponding sieve t' in the pattern t. This reflects the observation, that variables defined by a restrictive sieve, i.e. a term, merely establishes a shorthand for this term. This means that the mappings of these variables rely solely on the mappings of other variables, and to validate well-formedness, we must validate the variable mappings within the structure of these variables instead of the mappings of the variables themselves. This also reduces the well-formedness requirement to rely exclusively on the variables defined with non-restrictive sieves $\star$, similarly to the basic model. The rule (RWDef2) is therefore analogous to the rule for defined

variables in the basic model.

Notice that, although there are no explicit requirement to the restrictive sieve t in the rule (RWDef1), the recursiveness of the definition enforces that the sieve only holds known names, and that free variables are either known or defined in the matching. Furthermore, the definition ensures that there are no cyclic dependencies such as $t \triangleleft [x \mapsto y, y \mapsto x]$, as these definitions, after repeated application of rule (RWDef1), result in a term that includes variables not in $\mathcal{V}$, i.e. in the pattern mentioned here both x and y will be replaced by their corresponding sieves but not included in $\mathcal{V}$, and thus the pattern will be rejected.

The examples of how well-formedness ensures that perfect cryptography is enforced in the basic model presented in Example 2.2, naturally also apply to the extended model. The extended model however, introduces new ways to circumvent perfect cryptography, and the following example shows how the well-formedness condition presented above also captures these.

Example 3.1 The following patterns are all legal:

$$\begin{array}{ll} (1) & y \triangleleft [y \mapsto \mathsf{P}_{m^-}(x), x \mapsto \star] \quad \text{where } m^- \in \mathcal{N} \\ (2) & \mathsf{T}(\mathsf{E}_x(z), y) \triangleleft [x \mapsto y, y \mapsto \star, z \mapsto \star] \\ (3) & \mathsf{T}(x, z) \triangleleft [x \mapsto \mathsf{P}_y(y), y \mapsto z, z \mapsto \star] \end{array}$$

In (1) the sieve for y is $\mathsf{P}_{m^-}(x)$ and replacing y with this in the term shows that we afterwards can learn x, since the private key m^- is known. In (2), we see that by replacing x with its corresponding sieve y, we get a pattern where we legally can learn y and then decrypt the encryption and learn z. Notice that the definition list $[y \mapsto \star, x \mapsto y, z \mapsto \star]$ would also be legal as y could be learnt prior to replacing x, although this encoding seems less intuitive. Finally in (3) we first replace x with the sieve $\mathsf{P}_y(y)$, then y with z, resulting in $\mathsf{T}(\mathsf{P}_z(z), z)$ which is an obviously legal pattern, and would for instance match a value on the form $\mathsf{T}(\mathsf{P}_{n^-}(n^+), n^+)$.

Again we can render these patterns illegal by modifying them slightly. The following patterns are all correctly disqualified by the well-formedness condition:

$$\begin{array}{ll} (1') & y \triangleleft [y \mapsto \mathsf{P}_{m^-}(x), x \mapsto \star] \quad \text{where } m^- \notin \mathcal{N} \\ (2') & \mathsf{T}(\mathsf{E}_x(z), y) \triangleleft [x \mapsto \star, y \mapsto x, z \mapsto \star] \\ (3') & \mathsf{T}(x, z) \triangleleft [x \mapsto y, y \mapsto x, z \mapsto \star] \end{array}$$

The pattern $(1')$ is illegal as it includes an unknown name, $m^- \notin \mathcal{N}$. In pattern $(2')$ the sieve for x and y are changed such that y depends on x instead, but the now reverse definition order means that we cannot learn x prior to substituting y with its pattern. Notice however that if we define y before x, the pattern becomes legal. Pattern $(3')$ defines x to depend on y and y to depend on x, obviously this is illegal, and it will also be disqualified as first substituting x with y and then y with x results in the pattern $\mathsf{T}(x, z)$ in which x is undefined, i.e. the resulting term is not well-formed as $x \notin \mathcal{V}$.

4 Modelling

In this section we will give some examples on how to use the proposed language extension to model security protocols. As the proposed extension is supposed to be

applicable to a large variety of process calculi, we cannot assume anything about the underlying semantics of the communication model. Instead, we shall formalise the protocol in an extended protocol narration [3], where we distinguish between outputs and corresponding inputs, and between encryptions and corresponding decryptions.

In the encodings we follow [3] and extend each sent message with source and destination information as the first two elements, simulating the IP address along the lines of IPv4 and IPv6. Upon receipt of a message the principal will always check whether the message is intended for it; occasionally it will also check that the sender is who it expected.

For readability we shall employ the notational convention of x instead of the more cumbersome $x \mapsto \star$ and also omit the tuple construct as this obviously can be added automatically by a parser.

4.1 Wide Mouthed Frog

The Wide Mouthed Frog protocol [6] is a symmetric key protocol, for establishing a short term key K between two principals A and B, who both trust a server S. In our modelling we shall consider the following version [1]:

$$\begin{array}{ll} 1. & A \to S \ : \ A, \mathsf{E}_{K_A}(B, K) \\ 2. & S \to B \ : \ \mathsf{E}_{K_B}(A, K) \\ 3. & A \to B \ : \ \mathsf{E}_{K}(msg) \end{array}$$

Here K_A and K_B are master keys that A and B, respectively, are assumed to share with the server S. The key K is the session key that A and B shares after completion of the protocol, such that they in the last step can communicate the message msg encrypted.

This protocol is a commonly used example because of its simplicity and it is also, since it only uses symmetric encryption, a well-suited choice for the basic model. As described above, we shall formulate the protocol in an extended protocol narration and for this we use the patterns and the matching operator $\triangleright$. A direct translation into an extended protocol narration in the style of [3] looks as follows:

$$\begin{array}{lllll} 1. & A & \to & & : A, S, A, \mathsf{E}_{K_A}(B, K) \\ 1'. & & \to & S & : x_A, S, x_A, x \triangleleft [x_A, x] \\ 1''. & & & S & : x \triangleright \mathsf{E}_{K_{x_A}}(x_B, x_K) \triangleleft [x_B, x_K] \\ 2. & S & \to & & : S, B, \mathsf{E}_{K_{x_B}}(A, K) \\ 2'. & & \to & B & : y_S, B, y \triangleleft [y_S, y] \\ 2''. & & & B & : y \triangleright \mathsf{E}_{K_B}(y_A, y_K) \triangleleft [y_A, y_K] \\ 3. & A & \to & & : A, B, \mathsf{E}_{K}(msg) \\ 3'. & & \to & B & : y_A, B, y' \triangleleft [y'] \\ 3''. & & & B & : y' \triangleright \mathsf{E}_{K}(y_{msg}) \triangleleft [y_{msg}] \end{array}$$

Each line in the simple Alice-Bob protocol narration, has been translated into three lines in the extended narration. The first line describes the actions of the sender, the second line then describe how the recipient inputs the message, and the third line describes how the recipient decrypts some of the received elements using pattern matching. Notice that this encoding assumes that the distributed key K is fresh, and that S knows the master key of A and B, called K_A and K_B respectively.

However, this encoding can be optimised as all the decryptions can be incorporated into the input patterns directly. And the resulting narration looks as follows:

$$
\begin{array}{lll}
1. & A \rightarrow & : A, S, A, \mathsf{E}_{K_A}(B, K) \\
1'. & \rightarrow S & : x_A, S, x_A, \mathsf{E}_{K_{x_A}}(x_B, x_K) \triangleleft [x_A, x_B, x_K] \\
2. & S \rightarrow & : S, B, \mathsf{E}_{K_{x_B}}(A, K) \\
2'. & \rightarrow B & : y_S, B, \mathsf{E}_{K_B}(y_A, y_K) \triangleleft [y_S, y_A, y_K] \\
3. & A \rightarrow & : A, B, \mathsf{E}_K(msg) \\
3'. & \rightarrow B & : y_A, B, \mathsf{E}_K(y_{msg}) \triangleleft [y_{msg}]
\end{array}
$$

This optimised encoding is as detailed about the actions of the participants as the former encoding, but the readability is much higher. It is also noticeable that this encoding uses less variables, as it is not necessary to store temporary values in temporary variables.

4.2 ISO Three-Pass Mutual Authentication Protocol

The ISO protocols were proposed in Part 3 of the ISO/IEC 9798 Standard [8]. These protocols use asymmetric cryptography for authentication, and thus they are all good choices for the extended model. We have chosen to model the three-pass version, which in [7] is formalised as follows:

$$
\begin{array}{ll}
1. & B \rightarrow A : B, Rb, T1 \\
2. & A \rightarrow B : CertA, Ra, Rb, B, T3, \mathsf{P}_{K_A^-}(Ra, Rb, B, T2) \\
3. & B \rightarrow A : CertB, Rb, Ra, A, T5, \mathsf{P}_{K_B^-}(Rb, Ra, A, T4)
\end{array}
$$

The protocol authenticates A and B to each other. Apart from the nonces Ra and Rb and the text fields $T1 - T5$, the protocol relies on the certificates $CertA$ and $CertB$. These certificates are supposed to be generated by a trusted certificate authority CA, and usually include the name of the certificate authority, the name of the certified principal and its public key, all signed by the certificate authority's private key. Hence A's certificate for instance is directly translated into our syntax as $\mathsf{P}_{K_{CA}^-}(A, CA, K_A^+)$, and when B receives A's certificate in step 2, he will be able to learn A's public key and thereby ensure that the last part of the message is actually signed by A.

When modelling this protocol, we must also model the distribution of the certificates, as these should originate from the certificate authority. Thus the resulting

extended protocol narration will look as follows:

$$
\begin{array}{llll}
0_A. & CA \rightarrow & : & \mathsf{P}_{K^-_{CA}}(A, CA, K^+_A) \\
0'_A. & \rightarrow A & : & x_{cert} \triangleleft [x_{cert} \mapsto \mathsf{P}_{K^+_{CA}}(A, CA, K^+_A)] \\
0_B. & CA \rightarrow & : & \mathsf{P}_{K^-_{CA}}(B, CA, K^+_B) \\
0'_B. & \rightarrow B & : & y_{cert} \triangleleft [y_{cert} \mapsto \mathsf{P}_{K^+_{CA}}(B, CA, K^+_B)] \\
1. \quad B & \rightarrow & : & B, Rb, T1 \\
1'. & \rightarrow A & : & x_B, x_{Rb}, x_{T1} \triangleleft [x_B, x_{RB}, x_{T1}] \\
2. \quad A & \rightarrow & : & x_{cert}, Ra, x_{Rb}, B, T3, \mathsf{P}_{K^+_A}(Ra, x_{Rb}, B, T2) \\
2'. & \rightarrow B & : & \mathsf{P}_{K^+_{CA}}(A, CA, y_{K^+_A}), y_{Ra}, Rb, B, y_{T3}, \mathsf{P}_{y_{K^+_A}}(y_{Ra}, Rb, B, y_{T2}) \\
 & & & \quad \triangleleft [y_{K^+_A}, y_{Ra}, y_{T3}, y_{T2}] \\
3. \quad B & \rightarrow & : & y_{cert}, Rb, y_{Ra}, A, T5, \mathsf{P}_{K^+_B}(Rb, y_{Ra}, A, T4) \\
3'. & \rightarrow A & : & \mathsf{P}_{K^+_{CA}}(B, CA, x_{K^+_B}), x_{Rb}, Ra, A, x_{T5}, \mathsf{P}_{x_{K^+_B}}(x_{Rb}, Ra, A, x_{T4}) \\
 & & & \quad \triangleleft [y_{K^+_A}, y_{T5}, y_{T4}]
\end{array}
$$

The encoding assumes that both A and B trust CA, and that they both know its public key K^+_{CA}. The first 4 lines then describe the distribution of the certificates, and it important to observe that A and B can only verify that the certificate is authentic but not recreate it themselves, and they therefore have to use the restrictive sieves to ensure that they only accept a correct certificate. The remaining part of the encoding is relatively trivial, note however that the patterns in $2'$ and $3'$ allow the receiver to learn the public key of the sender prior to decrypting the last encrypted part of the tuple, and this allows for the patterns to be well-formed.

5 Conclusion

This paper has developed an expressive syntax, general semantics and a notion of well-formedness for capturing the assumptions of perfect cryptography. The extended syntax builds on and refines ideas that can be found in the Spi-calculus, in LySA and in LySA$^{\mathrm{NS}}$ [5] in order to express patterns that enable to learn a number of secrets from a single transmitted message, as is common in security protocols.

The formal development has taken the form of defining a semantics that can also deal with imperfect cryptography (as when secret keys can be broken by brute force attack or successful guessing) which has been supplemented by general well-formedness conditions for ruling out those behaviours not allowed when assuming perfect cryptography.

The theo-rems es-tab-lis-hed in Ap-pendix A aim at show-ing that the well-formedness conditions are sufficiently restrictive that no improper behaviour is admitted; the example protocols modelled in Section 4 aim at showing that the well-formedness conditions are sufficiently flexible to be of widespread interest.

Clearly there is more to a process calculus than the modelling of cryptography, but as we already said in the introduction, we believe the other features to be somewhat orthogonal to this. We are currently working [9] on embedding the ideas exposed here into the process calculus KLAIM [2] that deals with communication by means of distributed tuple spaces - so far without any surprises. Future work

involves showing that this development can also be integrated with the calculi for service-orientation and orchestration developed in the Sensoria project.

References

[1] Abadi, M. and A. D. Gordon, *A calculus for cryptographic protocols: the spi calculus*, in: *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security* (1997), pp. 36–47.

[2] Bettini, L., V. Bono, R. De Nicola, G. Ferrari, D. Gorla, M. Loreti, E. Moggi, R. Pugliese, E. Tuosto and B. Venneri, *The Klaim Project: Theory and Practice*, in: *Global Computing. Programming Environments, Languages, Security, and Analysis of Systems, IST/FET International Workshop, GC 2003, Revised Papers*, LNCS **2874** (2003), pp. 88–150.

[3] Bodei, C., M. Buchholtz, P. Degano, F. Nielson and H. R. Nielson, *Automatic Validation of Protocol Narration.*, in: *CSFW*, 2003, pp. 126–140.

[4] Bodei, C., M. Buchholtz, P. Degano, H. R. Nielson and F. Nielson, *Static Validation of Security Protocols*, Journal of Computer Security (2004).

[5] Buchholtz, M., H. R. Nielson and F. Nielson, *A calculus for control flow analysis of security protocols.*, Int. J. Inf. Sec. **2** (2004), pp. 145–167.

[6] Burrows, M., M. Abadi and R. Needham, *A logic of authentication*, ACM Trans. Comput. Syst. **8** (1990), pp. 18–36.

[7] Clark, J. and J. Jacob, *A survey of authentication protocol literature: Version* (1997).
URL citeseer.ifi.unizh.ch/clark97survey.html

[8] ISO, *ISO/IEC 9798-3. Information technology - Security techniques - Entity authentication mechanisms - Part 3: Entity authentication using a public-key algorithm*, Technical report, ISO, Geneva, Switzerland (1993), first edition.

[9] Nielsen, C. R., H. R. Nielson and F. Nielson, *CryptoKlaim*, work in progress.

A Validating the Design

In this appendix we shall show some important properties of our language extension. The first is that well-formedness is preserved in the semantics, and the second that a well-formed pattern guarantees perfect cryptography. As was the case for the development of the language extension, we shall proceed in two phases. First we shall prove that the results apply to the basic model, and then using the same proof technique we shall extend the results to the extended model.

A.1 Basic Model

That well-formedness is preserved in the semantics, is captured by Proposition A.1.

Proposition A.1 *If* $\vdash v \rhd p : \theta$ *and* $(\mathcal{N}, \mathcal{V}) \vdash v$ *then it follows that also* $\forall x \in dom(\theta) : (\mathcal{N}, \mathcal{V}) \vdash \theta x$.

Proof. The proof proceeds by induction in the definition of $\mathcal{X} \vdash v \rhd t : \theta$ where each case follows directly from the induction hypothesis. □

Notice that the result does *not* require the pattern to be well-formed but only the value, as well-formed patterns only enforce the assumption of perfect cryptography.

This leads us to the next and less trivial property, namely that well-formed patterns enforce the assumption of perfect cryptography.

We shall prove this result by first introducing a principal I that plays according to the rules of perfect cryptography. Formally, we shall define I's knowledge $I(\mathcal{K})$ as listed in Table A.1. Here rule (I1) shows that I has some initial knowledge $\mathcal{K}$.

$$(\mathsf{I1})\ \frac{t \in \mathcal{K} \quad \mathsf{fv}(t) = \emptyset}{t \in I(\mathcal{K})}$$

$$(\mathsf{I2})\ \frac{\mathsf{T}(t_1, \ldots, t_k) \in I(\mathcal{K})}{t_1, \ldots, t_k \in I(\mathcal{K})} \qquad (\mathsf{I3})\ \frac{\mathsf{E}_{t_0}(t) \in I(\mathcal{K}) \quad t_0 \in I(\mathcal{K})}{t \in I(\mathcal{K})}$$

$$(\mathsf{I4})\ \frac{t_1, \ldots, t_k \in I(\mathcal{K})}{\mathsf{T}(t_1, \ldots, t_k) \in I(\mathcal{K})} \qquad (\mathsf{I5})\ \frac{t_0, t \in I(\mathcal{K})}{\mathsf{E}_{t_0}(t) \in I(\mathcal{K})}$$

Table A.1
The knowledge of I; $I(\mathcal{K})$.

It can extend its knowledge by decomposing the values according to perfect cryptography (rules (I2) and (I3)), i.e. decrypting using known keys and decomposing tuples. Furthermore, I has the ability to construct new values from its knowledge by concatenating known values into new tuples and creating new encryptions from known keys and contents (rules (I4) and (I5)).

Apart from the knowledge of the principal I, we must also keep track of the variable mappings that I knows. We do this by introducing a mapping Θ defined analogous to θ. Given these ingredients we now define that a pattern p guarantees perfect cryptography if and only if allowing I to use pattern matching of any value $v \in I(\mathcal{K})$ against p would never produce variable mappings not already in $I(\mathcal{K})$. More formally we have:

Definition A.2 The pattern $t \triangleleft \mathcal{X}$ guarantees perfect cryptography if and only if for all v and Θ where $(\mathsf{fv}(t) \backslash \mathcal{X}) \subseteq dom(\Theta)$, $range(\Theta) \subseteq I(\mathcal{K})$ and $v \in I(\mathcal{K})$ then $\mathcal{X} \vdash v \rhd (\Theta t) : \theta$ implies $range(\theta) \subseteq I(\mathcal{K})$.

This provides us with a concrete requirement for patterns to guarantee perfect cryptography, which we now must prove is ensured by the well-formedness condition.

First we shall state some facts which clarifies the proofs. All of the facts follow directly from straightforward induction and we shall therefore omit the proofs.

Fact A.3 *If $\mathcal{X} \vdash v \rhd t : \theta$ then $dom(\theta) \subseteq \mathcal{X}$.*

Fact A.4 *If $\mathcal{X} \vdash v \rhd t : \theta$ where $\mathsf{fn}(t) \subseteq I(\mathcal{K})$ and $\mathsf{fv}(t) = [\,]$ then $v \in I(\mathcal{K})$.*

Fact A.5 *If $(x :: \mathcal{X}) \vdash v \rhd t : \theta$ and $x \in dom(\theta)$ then $\mathcal{X} \vdash v \rhd t[\theta x/x] : \theta \backslash \{x \mapsto \theta x\}$.*

Before we proceed to main result itself, we need an auxiliary result, namely that $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$ is only satisfied if x can be learnt from t without violating perfect cryptography.

Lemma A.6 *The judgement $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$ ensures that for all v and Θ where $\mathcal{V} \subseteq dom(\Theta)$, $x \notin dom(\Theta)$, $v \in I(\mathcal{K})$ and $\mathcal{N} \cup range(\Theta) \subseteq I(\mathcal{K})$ then $\mathcal{X} \vdash v \rhd (\Theta t) : \theta$ implies $x \in dom(\theta)$ and $\theta x \in I(\mathcal{K})$*

Proof. The proof proceeds by induction in the structure of $\mathcal{X} \vdash v \triangleright (\Theta t) : \theta$, where the interesting cases are (Bind) and (SDec):

Case (Bind). Assuming that $\mathcal{X} \vdash v \triangleright x : [x \mapsto v]$ by (Bind) because $x \in \mathcal{X}$, the result follows directly from the assumption $v \in I(\mathcal{K})$.

Case (SDec). Assume that $\mathcal{X} \vdash \mathsf{E}_{v_0}(v_1) \triangleright (\Theta \mathsf{E}_{t_0}(t_1)) : \theta_0 \cup \theta_1$ by (SDec) because $\mathcal{X} \vdash v_0 \triangleright (\Theta t_0) : \theta_0$ and $\mathcal{X} \vdash v_1 \triangleright (\Theta t_1) : \theta_1$. Assume furthermore that $(\mathcal{N}, \mathcal{V}) \vdash \mathsf{E}_{t_0}(t_1) \sim x$, $\mathcal{V} \subseteq dom(\Theta)$, $x \notin dom(\Theta)$, $v \in I(\mathcal{K})$ and $\mathcal{N} \cup range(\Theta) \subseteq I(\mathcal{K})$. From (WH2) we now get $(\mathcal{N}, \mathcal{V}) \vdash t_1 \sim x$ and also that $\mathsf{fn}(t_0) \subseteq \mathcal{N} \subseteq I(\mathcal{K})$ and $\mathsf{fv}(t_0) \subseteq \mathcal{V} \subseteq dom(\Theta)$. The latter implies that $\mathsf{fv}(\Theta t_0) = [\,]$ and thus from Fact A.4 we get $v_0 \in I(\mathcal{K})$. Finally by (I3) we have $v_1 \in I(\mathcal{K})$, and the induction hypothesis then gives us that $x \in \theta_1$ and $\theta_1 x \in I(\mathcal{K})$ which concludes the proof. □

This leads us to the main result itself, namely that $(\mathcal{N}, \mathcal{V}) \vdash p$ ensures that the pattern p guarantees perfect cryptography.

Theorem A.7 *If $(\mathcal{N}, \mathcal{V}) \vdash t \triangleleft \mathcal{X}$ then $t \triangleleft \mathcal{X}$ guarantees perfect cryptography.*

Proof. This is proven by induction in the structure of $\mathcal{X}$:

Case $[\,]$ is trivially true due to Definition A.2 and Fact A.3.

Case $x :: \mathcal{X}'$. Assume that $(\mathcal{N}, \mathcal{V}) \vdash t \triangleleft (x :: \mathcal{X}')$ by (WPDef) because $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$, $(\mathcal{N}, \mathcal{V} \cup \{x\}) \vdash t \triangleleft \mathcal{X}'$ and $x \notin \mathcal{V}$. Now assume v and Θ where $\mathcal{V} \subseteq dom(\Theta)$, $x \notin dom(\Theta)$, $v \in I(\mathcal{K})$ and $\mathcal{N} \cup range(\Theta) \subseteq I(\mathcal{K})$ such that $(x :: \mathcal{X}') \vdash v \triangleright (\Theta t) : \theta$. From Lemma A.6 we have that $x \in dom(\theta)$ and $\theta x \in I(\mathcal{K})$ and due to Fact A.5 this implies that $\mathcal{X} \vdash v \triangleright (\Theta' t) : \theta \backslash \{x \mapsto \theta x\}$ where $\Theta' = \Theta[x \mapsto \theta x]$. Thus from the induction hypothesis we get that also $range(\theta \backslash \{x \mapsto \theta x\}) \subseteq I(\mathcal{K})$ and according Definition A.2 and Fact A.3 this ensures perfect cryptography and concludes the proof. □

A.2 Extended Model

We shall now proceed by showing that the two results also apply to the extended model. Again in this case the first result follows directly from the definition of the semantics of pattern matching.

Proposition A.8 *If $\vdash v \triangleright p : \theta$ and $(\mathcal{N}, \mathcal{V}) \vdash v$ then it follows that also $\forall x \in dom(\theta) : (\mathcal{N}, \mathcal{V}) \vdash \theta x$.*

Proof. The proof proceeds by induction in the definition of $\Gamma \vdash v \triangleright t : \theta$ where each case follows directly from the induction hypothesis. □

We then introduce a new principal I_x, which similarly to I acts according to the assumption of perfect cryptography, but in the extended model. This is captured by the definition in Table A.2, where we merely extended the definition of I with the rules (RI4), (RI5) and (RI7) for public key encryption and digital signatures.

$$(\mathsf{RI1})\quad \frac{t \in \mathcal{K} \quad \mathsf{fv}(t) = \emptyset}{t \in I_x(\mathcal{K})}$$

$$(\mathsf{RI2})\quad \frac{\mathsf{T}(t_1, \ldots, t_k) \in \mathcal{K}}{t_1, \ldots, t_k \in I_x(\mathcal{K})} \qquad (\mathsf{RI3})\quad \frac{\mathsf{E}_{t_0}(t) \in \mathcal{K} \quad t_0 \in \mathcal{K}}{t \in I_x(\mathcal{K})}$$

$$(\mathsf{RI4})\quad \frac{\mathsf{P}_{n^+}(t) \in \mathcal{K} \quad n^- \in \mathcal{K}}{t \in I_x(\mathcal{K})} \qquad (\mathsf{RI5})\quad \frac{\mathsf{P}_{n^-}(t) \in \mathcal{K} \quad n^+ \in \mathcal{K}}{t \in I_x(\mathcal{K})}$$

$$(\mathsf{RI6})\quad \frac{t_0, t \in \mathcal{K}}{\mathsf{E}_{t_0}(t) \in I_x(\mathcal{K})} \qquad (\mathsf{RI7})\quad \frac{t_0, t \in \mathcal{K}}{\mathsf{P}_{t_0}(t) \in I_x(\mathcal{K})}$$

$$(\mathsf{RI8})\quad \frac{t_1, \ldots, t_k \in \mathcal{K}}{\mathsf{T}(t_1, \ldots, t_k) \in I_x(\mathcal{K})}$$

Table A.2
The knowledge of I_x; $I_x(\mathcal{K})$.

We can now define how a pattern p can ensure perfect cryptography in the extended model.

Definition A.9 The pattern $t \triangleleft \Gamma$ guarantees perfect cryptography if and only if for all v and Θ where $(\mathsf{fv}(t) \backslash dom(\Gamma)) \subseteq dom(\Theta)$, $range(\Theta) \subseteq I(\mathcal{K})$ and $v \in I(\mathcal{K})$ then $\Gamma \vdash v \rhd (\Theta t) : \theta$ implies $range(\theta) \subseteq I(\mathcal{K})$.

Analogous to the basic model, we shall first state some facts. These facts follow directly from straightforward induction, and thus we shall omit the proofs.

Fact A.10 *If* $\Gamma \vdash v \rhd t : \theta$ *then* $dom(\theta) \subseteq dom(\Gamma)$.

Fact A.11 *If* $\Gamma \vdash v \rhd t : \theta$ *where* $\mathsf{fn}(t) \subseteq I(\mathcal{K})$ *and* $\mathsf{fv}(t) = [\,]$ *then* $v \in I(\mathcal{K})$.

Fact A.12 *If* $((x \mapsto \star) :: \Gamma) \vdash v \rhd t : \theta$ *and* $x \in dom(\theta)$ *then* $\Gamma \vdash v \rhd t[\theta x/x] : \theta \backslash \{x \mapsto \theta x\}$.

Fact A.13 *If* $((x \mapsto t') :: \Gamma) \vdash v \rhd t : \theta$ *then* $\Gamma \vdash v \rhd t[t'/x] : \theta \backslash \{x \mapsto \theta x\}$.

Notice the last fact, which we did not need for the basic model.

Before we proceed to the main result itself, we shall prove that $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$ is only satisfied if x can be learnt from t without violating perfect cryptography.

Lemma A.14 *The judgement* $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$ *ensures that for all* v *and* Θ *where* $\mathcal{V} \subseteq dom(\Theta)$, $x \notin dom(\Theta)$, $v \in I(\mathcal{K})$ *and* $\mathcal{N} \cup range(\Theta) \subseteq I(\mathcal{K})$ *then* $\Gamma \vdash v \rhd (\Theta t) : \theta$ *implies* $x \in dom(\theta)$ *and* $\theta x \in I(\mathcal{K})$

Proof. The proof proceeds by induction in the structure of $\Gamma \vdash v \rhd (\Theta t) : \theta$ follows closely the proof for Lemma A.6. In particular are the cases (RBind1) and (RBind2) analogous to the case (Bind), and the cases (RPDec) and (RSDec) analogous to the case (SDec), and the remaining cases are trivial. □

And now we can prove the main result that well-formedness of the patterns in the extended model, ensures the assumption of perfect cryptography.

Theorem A.15 *If* $(\mathcal{N}, \mathcal{V}) \vdash t \triangleleft \Gamma$ *then* $t \triangleleft \Gamma$ *guarantees perfect cryptography.*

Proof. This is proven by induction in the structure of Γ:

Case $[\,]$ is trivially true due to Definition A.9 and Fact A.10.

Case $(x \mapsto s) :: \Gamma'$. Assume that $(\mathcal{N}, \mathcal{V}) \vdash t \triangleleft ((x \mapsto s) :: \Gamma')$ then this may be due to one of two rules:

Sub-case (RWDef1) because $s = t'$ and $(\mathcal{N}, \mathcal{V}) \vdash t \sim x$, $(\mathcal{N}, \mathcal{V}) \vdash t[t'/x] \triangleleft \Gamma'$. Now assume v and Θ where $\mathcal{V} \subseteq dom(\Theta)$, $x \notin dom(\Theta)$, $v \in I(\mathcal{K})$ and $\mathcal{N} \cup range(\Theta) \subseteq I(\mathcal{K})$ such that $(x \mapsto t') :: \Gamma') \vdash v \triangleright (\Theta t) : \theta$. From Fact A.13 we get $\Gamma' \vdash v \triangleright t[t'/x] : \theta \backslash \{x \mapsto \theta x\}$ and thus by the induction hypothesis we get that $range(\theta \backslash \{x \mapsto \theta x\}) \in I_x(\mathcal{K})$ which again means that $\forall x' \in \mathsf{fv}(t') : \theta x' \in I_x(\mathcal{K})$, and as we know from the recursiveness of the definition and rule (RWEmp) that $\mathsf{fn}(t') \subseteq \mathcal{N} \subseteq I_x(\mathcal{K})$, we can conclude from Fact A.11 that also $\theta x \in I_x(\mathcal{K})$. Thus according to Definition A.9 and Fact A.10, we have that the pattern ensures cryptography.

Sub-case (RWDef2) is similar to the case (WPDef) in the basic model.

And that concludes the proof.

□