Automated verification of equivalence properties in cryptographic protocols modulo AC

November 10, 2012

Contents

1 Introduction

Ensuring secure communication between two parties has always been an interesting task when the involved parties are communicating from a distance. Also when the goal is to exchange sensitive messages, this task becomes all the more important. To effectively carry out this task, one needs to take care of several aspects: The sender of the message must be sure that the message has in fact reached the intended sender, both the parties must be sure that the message has not been intercepted by a third party. Through the ages, several methods have been devised for ensuring secure communication.

With the emergence of internet and technologies like that of wireless communication, the task of ensuring secure communication has been facing new kinds of problems.

1.1 Cryptographic Primitives

One of the earliest documented uses of cryptography was by Julius Caesar, who would employ the Caesar cipher [vT03] to protect messages of military significance. The Caesar cipher and the slightly more sophisticated ciphering methods that followed are known today to be trivially broken using statistical methods. As the need for secure communication increased with the advent of electronic computers and electronic communication, the need for strong ciphers (i.e. which cannot easily be attacked) increased. In 1977 the United States Federal Information Processing Standard adopted the Data Encryption Standard (DES) [oST99] as a standard. DES enjoyed widespread use internationally, however, due to the small key-length, it quickly became vulnerable to brute-force attacks. This led to the adoption of AES (Advanced Encryption Standard) [oST01] to supersede DES.

These standards are openly available for anyone to implement and analyze. The openness of the standards follows Kerckhoffs' principle [DK01], widely followed by the scientific community. It states that a cryptosystem should be secure even if everything about the system, except the key, is public knowledge (i.e. including the algorithms for encrypting or decrypting). Due to the fact that anyone can analyze the security of these ciphers, we can be confident that they are secure.

All these are examples of what we call the symmetric key cryptography, that is the same key is used for encryption and decryption. However, this kind of system has a big disadvantage that it requires the two parties that are communicating to have the same key and securely sharing this key again becomes a problem. Another issue is the volume of keys required since each pair of communicating parties will require a unique key.

This lead to the development of public key cryptography. In public key cryptography, an agent generates a pair of keys: a public key which he shares with everyone and a private key which he keeps to himself. If someone wishes to communicate sensitive information to the agent, he encrypts the plaintext with the public key of the recipient; the recipient in turn receives the ciphertext which may be decrypted using the private key to obtain the original plain-text. It is now sufficient to authentically know the public key of the person which is to receive the sensitive information. Public/private key encryption is also called asymmetric encryption due to the fact that different keys are used for encryption and respectively decryption.

Other developments in asymmetric cryptography include digital signatures, hash function, etc. Digital signatures [RLRA83] also work using private/public key pairs. An agent signs a message using his private key to obtain a signed message. Anyone who knows the public key can verify that the resulting message was indeed signed using the associated private key, thereby allowing to establish the identity of the signer, provided that the public key is known authentically. Hash function was designed to ensure integrity of a message. Typically, a hash function is a one-way function that takes an arbitrary message and returns a fixed-size bitstring. Ideally, you cannot recover a message from its hash value.

The algorithms for symmetric key encryption, for public key encryption and decryption and for digital signatures are called *cryptographic primitives* since they are used as part of more complex protocols. The security of cryptographic primitives is analyzed individually and the security of the exchanged messages relies on the security of the cryptographic primitives involved.

1.2 Cryptographic Protocols and their Modelling

Cryptographic protocols are distributed programs which rely on the use of cryptography to secure elecetronic transactions on channels that may be controlled by an attacker. They are typically specified as a set of rules which are abstract patterns of communication, specifying which messages should be sent and how to respond to the reception of any message.

1.2.1 Need for Modelling Protocols

In order to gain confidence about the security of a protocol, the protocol should be modelled effectively, that is the model should describe the protocol as accurately as possible. Then the protocol should be formally verified for the expected security properties. This has given the rise to a class of models for security protocols generically called the *Dolev-Yao model*. The main ingredient of the Dolev-Yao model is that messages exchanged by parties are represented as terms in a term algebra, with function symbols representing cryptographic primitives. In the Dolev-Yao model, protocols are modeled in various formalisms such as a set of roles in role-based models as shown in their paper [DY83].

Dolev-Yao model also has many implications about the power of the intruder. It is assumed that the intruder can intercept all messages, block any message, construct new messages according to the deduction rules, impersonate a protocol agent and send messages of his own creation to the agents.

Once the various details described above are fixed, a model of the security protocol can be created. It remains to model the security property that we expect of the protocol. Traditionally, two security properties are verified:

1. Secrecy (or confidentiality) of some secret value, which intuitively means that the attacker should not be able to find the secret value.

2. Authentication, which means intuitively that an agent should be sure that he is talking to whom he believes to be talking.

With protocols getting more and more complex, the demands about the desired security properties are also becoming more complex. We will describe other security properties in detail in the subsequent sections.

1.2.2 Modelling Cryptographic Primitives

Consider a term like $enc(x, y)$ which models the encryption of some data x with key y. The power of the intruder can be expressed as a deduction system consisting the following inference rules:

$$\frac{enc(x,y) \qquad y}{x} \qquad \frac{x \qquad y}{enc(x,y)}$$

The inference rules allow the adversary to construct new messages from messages already known either by encrypting known data x with known key y to obtain $enc(x, y)$ or by decrypting an already-known encrypted message $enc(x, y)$ by the already-known associated encryption key y to obtain x (the fact that the intruder can learn a new message by observing network traffic is implicit and there is no need to capture this explicitly in the inference rules).

Modelling cryptographic primitives has always been important because many attacks rely only on the logical structure of the protocols and simply consist of replaying some messages at the right steps. That is why formal methods usually consider encryption schemes as black boxes and assume that an adversary cannot learn anything from an encrypted message except if he has the key. For instance, in the above example, due to the lack of other inference rules, the only way for the attacker to find the contents of an encrypted message is by knowing the encryption key. This is known as the *perfect encryption hypothesis*. Even if these assumptions are not realistic, many real attacks have been discovered using this approach.

1.2.3 Considering Some Algebraic Properties

We have seen that the *Perfect Cryptography* assumption is not very realistic. Recent works investigate how to relax the perfect cryptography assumption by refining the abstraction on cryptographic primitives. The aim is to take into account some of the algebraic properties of the cryptographic primitives. Most of the algebraic properties studied so far are properties that can be modelled using a set of equations, also known as an *Equational Theory*. The interest of studying the algebraic properties of the cryptographic primitives is that some attacks may be missed when abstracting encryption as a perfect black box.

It turns out that using equational theories allows one to entirely get rid of the deduction system. So continuing our previous example of encryption, we just need to add the function symbol *dec* and then we get our equational theory as:

$$E = \{dec(enc(x, y), y) \approx x\}$$

Another use of equational theories is to model cryptograhic primitives like *exclusive OR* (XOR). In that case E will consist of the following equations:

$$\begin{array}{cc} x \oplus 0 \approx x & x \oplus x \approx 0 \\ x \oplus (y \oplus z) \approx (x \oplus y) \oplus z & x \oplus y \approx y \oplus x \end{array}$$

Sometimes the equations in an equational theory can be oriented to form a convergent *Rewrite System* R which is nothing but a set of rules. So the equational theory E can also be expressed as:

$$R = \{dec(enc(x, y), y) \rightarrow x\}$$

We will later define equational theories and rewrite systems formally.

1.3 Verification

Once a model has been fixed for the cryptographic protocol, the next step is to formally verify the desired security properties. This is the phase wherein the attacks, if any are discovered. Verifying the security of a protocol depends on the goal of the protocol. Hence when describing a protocol, it is crucial to describe the *security properties* that the protocol is supposed to achieve.

1.3.1 Security Properties

Each cryptographic protocol may wish to achieve its own security properties, but the classical security properties have been grouped into two families [Che12]:

- Trace (Reachability) Properties
- Equivalence (Indistunguishability) Properties

Intuitively, the trace properties indicate that the protocol cannot reach a bad or undesired state. Authenticity, secrecy are trace properties. So we need to ensure that a protocol requiring authenticity cannot reach a state where an honest agent ends up communicating with someone other than who he intends it to be. Similarly we need to ensure that a protocol requiring secrecy cannot reach a state where the attacker can deduce the secret message.

Equivalence properties indicate the indistinguishability of certain instances of a protocol meaning that the attacker should not be able to distinguish the two instances of a protocol from each other. Many crucial security properties like anonymity, flavors of strong confidentiality, strong secrecy, etc. can only be modelled in terms of equivalence. So suppose we want to ensure strong secrecy in a protocol, we need to ensure that, if instead of a secret message m_1, a message m_2 is exchanged, the attacker would not be able to make out the difference.

1.3.2 Existing work and Contributions

A lot of work has been accomplished for trace properties and a lot of tools have also been developed for automated verification of trace properties for the restricted case of a *passive adversary*. But very few works focus on equivalence properties. Formally, when verifying a trace property. we have to verify that any execution of the protocol cannot reach an undesired state. On the other hand, to verify an equivalence property, we have to verify that for any execution of one of the instances of the protocol, there exists an indistinguishable execution of the other instance of the protocol. Hence the alternation of quantifiers makes the verification of equivalence properties even more difficult.

Even the modelling of indistinguishablitity is considered a difficult task and requires extensive knowledge of the capabilities of the intruder. Depending on that, several models of indistinguishability have been proposed, which include trace-equivalence [BRR02], may-equivalence [AG99], observational equivalence [AF04]. It has been showed that observational equivalence in spi calculus is undecidable as cited in [RSS12]. Also numerous results exist for unbounded number of sessions.

In [RSS12], a new procedure for verifying equivalence properties for processes specified in a crptographic process calculus has been introduced. This procedure automatically checks for two equivalences $\approx_{ct}$ and $\approx_{ft}$ which over and under approximate the standard notion of trace equivalence $\approx_t$ for cryptographic protocols. The relation $\approx_{ft}$ can thus be used to prove protocols correct while the relation $\approx_{ct}$ can thus be used to rule out incorrect protocols.

Since it has been shown that observational equivalence coincides with $\approx_t$ for the class of determinate processes in [VS09], this procedure can be used to verify observational equivalence for the whole class of determinate processes . Also the procedure is based on a fully abstract modelling of symbolic traces for a bounded number of sessions in first-order Horn clauses.

The procedure in [RSS12] is fully abstract for arbitrary cryptographic primitives that can be modelled as a convergent rewrite system. But it is possible that we have an equational theory, only part of which can be oriented as a convergent rewrite system and what remains is a non convergent equational theory like AC(Associative Commutative). Such an extension in the current procedure will enable us to analyze protocols based on exclusive OR or Diffie-Hellman exponentiations. In this thesis, we have tried to modify the existing theory and the following procedure for such a generic equational theory.

1.4 Thesis Plan

We start by describing the basic terminology that will be required throughout the thesis and then formally define equational theories and rewrite systems. Since we have a theory E which can be split into rewrite system R and a non convergent theory E', our rewrite systems have to be defined modulo E'. Similarly the *set of variants* and the *set of unifiers* have to be defined modulo E'. The goal is to reduce terms using the rewrite rules and then check equality modulo E'.

Then we go ahead and describe the cryptographic process calculus that we will be needing for our procedure. After that we show how we model the processes using Horn Clauses. In this modelling, we define some new terminology arising due to the theory E'.

We describe the actual procedure after this starting with how to get the seed statements from a trace. We also prove the soundness and completeness (partial) of the seed statements. The whole time our aim is to illustrate the parallel between the process calculus and the Horn Clause modelling.

Throughout the thesis, we have used the modified version of *Salary Sum Protocol*[VSP06] as a running example. We describe the protocol informally and show that an attack exists using the process calculus. Then we show that the same attack can also be simulated by the seed statements.

2 Preliminaries

2.1 Terms

Let F be a signature, that is, a finite set of function symbols and let ar be a function that assigns to each function symbol a natural number which is its arity. A function symbol of arity 0 is called a constant. Given a set of *atoms* A and a signature F, we denote by $T_{F,A}$ the set of terms built inductively from A by applying functions symbols in F. Given sets of atoms $A_1, A_2, ..., A_n$, we denote the set $T_{F,\cup_{1\leq i\leq n}A_i}$ by $T_{F,A_1,A_2,...,A_n}$.

We assume that we have the following countably infinite pairwise disjoint sets: a set N of *private names*, M of *public names*, a set C of *public channel names*, a set W of *parameters*, and a set χ of *message variables*. Intuitively, elements of the set N represents nonces generated by honest principals of a protocol, elements of M represent nonces available both to the adversary and to the honest participants and elements of C represent names of public channels (e.g. the name of a public network). Elements of W are pointers used by the adversary to refer to messages output on the public channel by the honest participants in a protocol. We fix an enumeration $w_1, w_2, ...$ of the elements of W. We let x, y, z range over χ. We also define the following set of terms:

- **Terms** denotes the set of all terms $T_{F,N,M,W,\chi}$
- **Messages** denotes the set of messages $T_{F,N,M}$
- **SMessages** denotes the set of symbolic messages $T_{F,N,M,\chi}$

We will let A be the entire set of atoms ($A = \chi \cup N \cup W \cup M \cup C$).

If t is a term, we denote by $vars(t)$ the set of variables appearing in t, by $names(t)$ the set of names (public or private) appearing in t. The functions $vars$, $names$ are extended to sequences and sets of terms as expected.

A *position* is a string of positive natural numbers and ϵ denotes the empty string. The set $pos(t)$ of positions of a term t is defined as usual. If $p \in pos(t)$, then $t|_p$ is the subterm of t at position p.

Example: Consider the signature modelling the asymmetric encryption, $F = \{aenc, adec, pk\}$ where $ar(aenc) = ar(adec) = 2$ and $ar(pk) = 1$. Consider a protocol participant A and by principles of public key cryptography, we have sk_A(private key of A) $\in N$. So pk is a function symbol such that given the private key of a participant, it outputs the public key.
Let term $t = adec(aenc(m, pk(sk_A)), sk_A)$ with $m \in$ **Messages**. The set of positions $pos(t) = \{\epsilon, 1, 11, 12, 2\}$ and $t|_1 = aenc(m, pk(sk_A))$, $t|_{11} = m$ and $t|_\epsilon = t$.

2.2 Substitutions

A *substitution* is a partial function $\sigma : W \cup \chi \to Terms$. We restrict substitutions to map elements of W to elements of **Messages** and elements of χ to elements of **SMessages**. The domain of σ shall be denoted by $dom(\sigma)$. We only consider substitutions with finite domains.

We let $range(\sigma) = \{\sigma(u) \in T | u \in dom(\sigma)\}$. σ is said to be *ground* if $range(\sigma) \subseteq$ **Messages**

We denote by $\sigma[\chi]$ the substitution whose domain is restricted to χ. If $dom(\sigma) \subseteq \{x_1, \ldots, x_n\}$, we may write σ as $\sigma = \{x_1 \mapsto \sigma(x_1), \ldots, x_1 \mapsto \sigma(x_1)\}$.

The homomorphic extension of a substitution σ to the set of terms $T_{F,A}$ is the function $\bar{\sigma} : T_{F,A} \to T_{F,A}$, where

$$\begin{array}{cc} \bar{\sigma}(x) = \sigma(x) & \text{for all variables } x \in \chi \\ \bar{\sigma}(a) = a & \text{for all non-variable atoms } a \in A/\chi \\ \bar{\sigma}(f(t_1, \ldots, t_k)) = f(\bar{\sigma}(t_1), \ldots, \bar{\sigma}(t_k)) & \text{for all other terms } f(t_1, \ldots, t_k) \end{array}$$

We identify as usual substitutions with their homomorphic extensions and we may write substitution application in suffix form: the term $t\sigma$ is the application of the homomorphic extension of the substitution $\bar{\sigma}$ to the term t, i.e. $t\sigma = \bar{\sigma}(t)$.

We will use the standard definitions from [RSS12] whenever not specified.

2.3 Equational Theory

Definition 1. *An equational theory* $\boldsymbol{E}$ *is a set of identities of the form* $\{s_1 \approx t_1, s_2 \approx t_2, ..., s_n \approx t_n\}$ *where for all* i *we have* $s_i, t_i \in T_{F,\chi}$ *and* $=_E$ *is the smallest equivalence relation such that*

- $(s \approx t) \in E \Rightarrow (s =_E t)$
- $(s =_E t) \Rightarrow (s\sigma =_E t\sigma)$ *for all substitutions* σ
- $(s_1 =_E t_1) \wedge ... \wedge (s_n =_E t_n) \Rightarrow f(s_1, .., s_n) =_E f(t_1, .., t_n)$ *for any k-ary function* $f \in F$

The relation $=_E$ is called the equational theory generated by the set of identities E . We will usually call E itself as the equational theory.

3 Rewriting and Unification

3.1 Rewrite Systems

A *rewrite rule* is an identity $l \approx r$ such that $l \notin \chi$ and $vars(r) \subseteq (l)$. In this case we may write $l \to r$. A *term rewriting system* $R = \{l_1 \to r_1, \ldots, l_n \to r_n\}$ is a finite set of rewrite rules. We say that s rewrites to t, denoted by $s \to t$ if there exists $p \in pos(s)$, a substitution σ and $(l \to r) \in R$ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. The reflexive, transitive closure of $\to$ is denoted by $\to^*$.

A rewrite system R implements an equational theory E if the relations $=_R$ and $=_E$ coincide, that is if $R = \{l_1 \to r_1, \ldots, l_n \to r_n\}$ is a term rewriting system and $E = \{l_1 \approx r_1, \ldots, l_n \approx r_n\}$ is the associated equational theory, it is known that R implements E. A rewrite system R is terminating if there is no infinite rewrite chain $t_1 \to t_2 \to \ldots$. A rewrite system R is confluent if for all terms s, u, v such that $s \to u$ and $s \to v$ we have that there exists a term t such that $u \to t$ and $v \to t$. A rewrite system is *convergent* if it is both terminating and confluent.

An equational theory is convergent if it is implementable by a convergent rewrite system. Otherwise the equational theory is non-convergent.

Example(continued):Consider the equational theory AC(Associative Commutative) modelled by $E' = \{x + y \approx y + x, (x + y) + z \approx x + (y + z)\}$. This cannot be implemented by a convergent rewrite system because $(x + y) \to (y + x) \to (x + y) \to \dots$ forms an infinite rewrite chain. So E' is a non-convergent equational theory.

3.1.1 Rewrite Systems modulo an Equational Theory

We have an equational theory **E** such that it can be split into two parts: a rewrite system **R** and a non-convergent equational theory **E'**. Our aim is to modify and develop all the existing procedures and theories as stated in [RSS12] for a generic theory **E**, denoted by **(R,E')**

A rewrite system R modulo equational theory E' is again a set of rules of the form $l \to r$ where $l, r \in$ **Terms**, $names(l, r) = \phi$ and $vars(r) \subseteq (l)$. We say that s rewrites to t, denoted by $s \to_{R,E'} t$ if there exists $p \in pos(s)$, a substitution σ and $(l \to r) \in R$ such that $s|_p =_{E'} l\sigma$ and $t =_{E'} s[r\sigma]_p$.

For any term s, if $t \to_{R,E'} s$ implies $t =_{E'} s$ then the term t is said to be in normal form (wrt $\to_{R,E'}$). So no rewrite rules can be applied to a term in normal form. The reflexive transitive closure of $\to_{R,E'}$ is denoted by $\to^*_{R,E'}$. If $s \to^*_{R,E'} t$ and t is in normal form then we say that t is the normal form of s. The normal form of a term will no longer be unique, but if t and t' are normal forms of term s then $t =_{E'} t'$. We denote it by $s \downarrow_{(R,E')}$, or simply $s \downarrow$ when its clear from the context. We will use $s \downarrow$ in the following text. A term t is said to be **E'-normal** if $t \downarrow_{(R,E')} =_{E'} t$.

We say that a rewrite system R is E'-confluent if for all terms s, u, v such that $s \to_{R,E'} u$ and $s \to_{R,E'} v$ we have that there exists terms t, t' such that $u \to_{R,E'} t$ and $v \to_{R,E'} t'$ and $t =_{E'} t'$. It is said to be **E'-convergent** if, in addition, the rewrite system is terminating as well. The order in which rewriting rules are applied doesn't matter.

Example: Let $R = \{adec(aenc(x, pk(sk_y)), sk_y) \to x, x + 0 \to x, x + x \to 0\}$ and $E' = \{x + y \approx y + x, (x + y) + z \approx x + (y + z$ which is AC (Associative Commutative) and the signature F as described before in 2.1.

Now let $t = adec(aenc((x + (z + 0)), pk(sk_y)), sk_y)$. So $t \to_{R,E'} adec(aenc((x + z), pk(sk_y)), sk_y) \to_{R,E'} (z + x)$ and also $t \to_{R,E'} x + (z + 0) \to_{R,E'} (x + z)$. So $(z + x)$ and $(x + z)$ are both normal forms of t, that is both are $t \downarrow$ with $(z + x) =_{E'} (x + z)$.

3.2 The equational theory (R,E')

Definition 2. *(R, E') is said to be a decomposition of an equational theory E if,*

- *R is an E'-convergent rewrite system*
- *E' is an equational theory*
- *$(s =_E t)$ iff. $s \downarrow_{(R,E')} =_{E'} t \downarrow_{(R,E')}$*

Example(continued): We will consider the equational theory of Abelian groups combined with the asymmetric encryption as a running example. So $E = \{adec(aenc(x, pk(sk_y)), sk_y) \approx x, x + 0 \approx x, x + x \approx 0, x + y \approx y + x, (x + y) + z \approx x + (y + z)\}$. Hence $E = (R, E')$ such that it can be decomposed into a rewite system $R = \{adec(aenc(x, pk(sk_y)), sk_y) \to x, x + 0 \to x, x + x \to 0\}$ and a non convergent equational theory $E' = \{[x + y] \approx [y + x], [(x + y) + z] \approx [x + (y + z)]\}$.

But, for the equational theory E in its current form, R is not $E'-$convergent. In order to make R AC-convergent, we need to add some rules as shown in [HLS05]. So now R becomes:

$$R = \begin{cases} (-x) + (-y) \to -(x + y) & (-x + y) + y \to -(x) \\ -(-x) \to x & -(-0) \to 0 \\ -(-x + y) \to (x + (-y)) & x + (-x + y) \to y \\ (-x) + (-y + z) \to -(x + y) + z & -(x + y) + (y + z) \to (-x) + z \\ x + 0 \to x & x + x \to 0 \\ adec(aenc(x, pk(sk_y)), sk_y) \to x & \end{cases}$$

3.2.1 Complete Set of Variants

Definition 3. *For an equational theory E which is composed of a convergent rewrite system R and is E′-convergent, a set of substitutions* $\boldsymbol{variants}_E(t_1, ..., t_k)$ *is called a* complete set of variants *of the terms* $t_1, ..., t_k$ *modulo E′ if for any substitution* ω*, there exists a* $\sigma \in variants(t_1, ..., t_k)$ *and a substitution* τ *such that for all* $1 \leq j \leq k$ *we have,*

- $\omega[vars(t_j)] \downarrow_{(R,E')} =_{E'} (\sigma \downarrow_{(R,E')} \tau)[vars(t_j)]$
- $(t_j\omega) \downarrow_{(R,E')} =_{E'} (t_j\sigma) \downarrow_{(R,E')} \tau$

The purpose of having such a set of substitutions is so that the normal form of any term can be obtained just by using one of the substitutions in the set so that the rewrite rules needn't be applied at a later stage.

For the equational theory E, which can be decomposed as (R, E'), we say that R has the *finite variant property modulo* E' if for any sequence of terms, a finite, complete set of variants exists.

Example(continued):Given a term like $t = adec(x, sk_y)$ and a convergent rewrite system R as described in 3.2, we want to have symbolic representation of all normal forms $t \downarrow$ of instantiations $t\sigma$ of the term t. As shown in [S.C11], the normal forms $t\sigma \downarrow$ of $t\sigma$ fall into one of two cases depending on whether x is an encrypted term of the form $aenc(s, pk(sk_y))$ for some s or not.

So the set $\{\sigma_1, \sigma_2\}$ such that $\sigma_1 = \{x \mapsto aenc(s, pk(sk_y))\}$ and σ_2 is the identity substitution, is then called a complete set of variants of t or $\mathbf{variants}_E(t)$. So for any substitution ω, any normal form $t\omega \downarrow$ of an instantiation $t\omega$ of t will be a syntactic instantiation of a term $t\sigma_i \downarrow$ (for some $1 \leq i \leq 2$).

3.2.2 Complete Set of Unifiers

Definition 4. *Two terms s and t are (syntactically) unifiable modulo E for an equational theory E, if there exists a substitution* σ *such that* $s\sigma =_E t\sigma$.

For an equational theory E,

A set of substitutions $mgu_E \left\{(s_i \stackrel{?}{=} t_i)\right\}_{i \in I}$ *is called a complete set of unifiers modulo E of the system of equations* $\left\{(s_i \stackrel{?}{=} t_i)\right\}_{i \in I}$ *if:*

- $\forall \sigma \in mgu_E \left\{(s_i \stackrel{?}{=} t_i)\right\}_{i \in I}$, $dom(\sigma) \subseteq vars(X)$
- $\forall i \in I$, $\forall \sigma \in mgu_E \left\{(s_i \stackrel{?}{=} t_i)\right\}_{i \in I}$, $s_i\sigma =_E t_i\sigma$
- $\forall \theta$ *and* $\forall i \in I$*such that* $s_i\theta =_E t_i\theta$ *, there exists a* $\sigma \in mgu_E \left\{(s_i \stackrel{?}{=} t_i)\right\}_{i \in I}$ *and there exists* τ *such that*

$$\theta[X] =_E (\sigma\tau)[X]$$

where $X = vars(\{s_i, t_i\}_{i \in I})$

The set of substitutions obtained will no longer be unique. But if S_1 and S_2 are two of such sets obtained then for every $\sigma_1 \in T_1$, there exists $\sigma_2 \in T_2$ such that $\sigma_1(x) =_{E'} \sigma_2(x)$ for all $x \in X$.

4 Frames and Deducibility

We will use the notion of a $frame$ to represent the messages that have been recorded by an attacker. We fix an enumeration $w_1, w_2, \ldots$ of the elements of the set W. The atoms $w_1, w_2, \ldots$ are not used as building blocks in messages, but are used by the attacker to point to messages exchanged during the protocol. The parameter w_1 refers to the first message exchanged, w_2 to the second and so on. This suggests a way of modeling sequences of messages exchanged during a protocol:

Definition 5 (Frame)**.** *A frame* ψ *is a substitution* $\{\omega_1 \mapsto t_1, ..., \omega_n \mapsto t_n\}$ *where* $t_i \in \boldsymbol{Messages}$ *for* $1 \leq i \leq n$

4.1 Deducibility

Note that in our definition, every frame ψ with $|dom(\psi)| = n$ has $dom(\psi) = \{w_1, \ldots, w_n\}$. The set of all frames is denoted as **Frames**. The adversary can use the messages learnt from the run of a protocol to construct new messages. This is modeled as the deducibility relation,

Definition 6. *Any term in $T_{F,M,W}$ is said to be a **Recipe**. We say that a message t is deducible from ψ with a recipe r, denoted by $(\psi \vdash^r_E t)$ if $r\psi =_E t$*

Example: Consider the frame $\psi = \{\omega_1 \mapsto aenc(a, pk(sk_y)), \omega_2 \mapsto sk_y, \omega_3 \mapsto u + (-u + v)\}$ such that $a, u, v \in N$(private names). Then we get $(\psi \vdash^{adec(\omega_1,\omega_2)}_E a)$ and $(\psi \vdash^{\omega_3}_E v)$. Hence using the frame ψ, an attacker can deduce some private names.

4.2 Static Equivalence

The deduction relation described previously is enough to model secrecy of data. We can model the fact that a private name s remains secret after the protocol run if$(\psi \not\vdash^r_E s)$ for any recipe r, where ψ represents the frame collecting all messages from the protocol. However, for more subtle properties, deduction is not enough.

Static equivalence captures indistinguishability of sequences of messages:

Definition 7. *Let $r_1, r_2 \in Recipes$. A test $r_1 \stackrel{?}{=} r_2$ holds in a frame ψ (written $(r_1 = r_2)\psi$) if $(\psi \vdash^r_E t)$ and $(\psi \vdash^r_E t)$ for some t, i.e., r_1 and r_2 are recipes for the same term in ψ.*

A frame ψ_1 is statically included in ψ_2 (written $\psi_1 \sqsubseteq_s \psi_2$) iff. for all $r_1, r_2 \in Recipes$ we have that $(r_1 = r_2)\psi_1$ implies $(r_1 = r_2)\psi_2$. Two frames ψ_1 and ψ_2 are statically equivalent (written $\psi_1 \approx_s \psi_2$) iff. $\psi_1 \sqsubseteq_s \psi_2$ and $\psi_2 \sqsubseteq_s \psi_1$.

Example: Let $a, b \in M$ and $sk_y, sk_w \in N$. We have that

$$\psi_1 = \{\omega_1 \mapsto aenc(a, pk(sk_y)), \omega_2 \mapsto sk_y\} \not\approx_s \{\omega_1 \mapsto aenc(b, pk(sk_y)), \omega_2 \mapsto sk_y\} = \psi_2$$

because $(adec(\omega_1, \omega_2) = a)\psi_1$ whereas $(adec(\omega_1, \omega_2) = b)\psi_1$. But,

$$\{\omega_1 \mapsto aenc(a, pk(sk_y)), \omega_2 \mapsto sk_w\} \approx_s \{\omega_1 \mapsto aenc(b, pk(sk_y)), \omega_2 \mapsto sk_w\}$$

5 A Cryptographic Process Calculus

We shall assume that cryptographic protocols are modelled using a simple process calculus which has similarities with the applied pi-calculus as described in [MB09]. We shall restrict our attetion to finite i.e., replication-free fragment of applied pi-calculus. This restriction is important because observational equivalence becomes undecidable with replication.

We begin defining our process calculus by defining the syntax.

5.1 Syntax of Process Calculus

We model a bounded number of instances of a cryptographic protocol as a finite set of traces. Traces are defined using sequences of *actions* generated by the following grammar:

$$a = \begin{array}{rl} in(c, x) & \text{receive action} \\ out(c, t) & \text{send action} \\ [s \stackrel{?}{=} t] & \text{test action} \end{array}$$

where $x \in \chi$, $s, t \in SMessages$, $c \in C$.

A trace T is a sequence of actions $T = a_1.a_2.....a_n$. As usual, a receive action $in(c, x)$ acts as a binding construct for x. We assume the usual definitions of free and bound variables for traces as stated in [B.B]. We also assume that each variable is bound at most once. A trace is *ground* if it does not contain any free variables. The set of ground traces shall be represented as **GndTraces**. A set of traces $P = \{T_1, ..., T_n\}$ is said to be a *process*. A process is ground if all of its traces are ground. We identify traces with singleton processes.

5.2 Semantics of Process Calculus

The semantics of a process is defined using the semantics of its traces. The semantics of a trace is given in terms of a labeled transition system T. We assume the Dolev-Yao model for the adversary as described in [DY83].Hence all interactions between protocol participants are mediated by the adversary. The labeled transition system records the interaction of the protocol participants with the adversary. The set of labels of T is defined using the set *Recipes*. Recall that the set *Recipes* is the set $T_{F,M,W}$. The set of labels, *Labels*, is

$$\{in(c,r), out(c), test | r \in Recipes, c \in C\}.$$

The labeled transition system T is a subset of $(GndTraces \times Frames) \times Labels \times (GndTraces \times Frames)$. We write $(T,\psi) \xrightarrow{l} (T',\psi')$ whenever $((T,\psi), l, (T',\psi')) \in T$. The frame in the transition system is used to record the messages that the protocol participants have sent in the past. The relation $\xrightarrow{l}$ is defined as follows:

- RECEIVE:
$$\frac{\psi \vdash^r_E t}{(in(c,x).T,\psi) \xrightarrow{in(c,r)} (T\{x \mapsto t\},\psi)}$$
- SEND:
$$\frac{}{(out(c,t).T,\psi) \xrightarrow{out(c)} (T,\psi \cup \{\omega_{|dom(\psi)|+1} \mapsto t\})}$$
- TEST:
$$\frac{s =_E t}{([s \stackrel{?}{=} t].T,\psi) \xrightarrow{test} (T,\psi)}$$

The label $in(c,r)$ indicates a message sent by the adversary over the channel c and r is the recipe that adversary uses to create this message. The label $out(c)$ indicates a message sent over the public channel c and transition rule SEND records the message sent in the frame. Finally, the rule TEST is an internal action.

As usual we shall write $(T_0,\psi_0) \xrightarrow{l_1,\ldots,l_n} (T_n,\psi_n)$ and we say that $l_1 \ldots, l_n$ is a *run* of (T_0,ψ_0). We write $(T,\psi) \stackrel{l}{\Rightarrow} (T',\psi')$ when either $(T,\psi) \xrightarrow{test^*,l,test^*} (T',\psi')$ and $l \neq test$ or $(T,\psi) \xrightarrow{test^*} (T',\psi')$ and $l = test$ where $test^*$ denotes an arbitrary number of test actions.

We also write $(T,\psi) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T_n,\psi_n)$ when $(T,\psi) \stackrel{l_1}{\Rightarrow} (T_1,\psi_1) \stackrel{l_2}{\Rightarrow} \ldots \stackrel{l_n}{\Rightarrow} (T_n,\psi_n)$. If $P = \{T_1,\ldots,T_m\}$ is a process, we write $(P,\psi) \xrightarrow{l_1,\ldots,l_n} (T',\psi')$ (resp. $\stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T',\psi')$ if there exists a trace $T \in P$ such that $(T,\psi) \xrightarrow{l_1,\ldots,l_n} (T',\psi')$ (resp. $(T,\psi) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T',\psi')$).

5.3 Process Equivalences

We will now define different flavors of trace equivalence. We first recall the standard definition of trace equivalence in cryptographic process algebras.

Definition 8 (Trace equivalence)**.** *A ground process P is said to be trace included in a ground process Q (written $P \sqsubseteq_t Q$) if whenever $(P,\phi) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T,\psi)$ then there exists T',ψ' such that $(Q,\phi) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T',\psi')$ and $\psi \approx_s \psi'$. Two processes P and Q are trace-equivalent (written $P \approx_t Q$) if $P \sqsubseteq_t Q$ and $Q \sqsubseteq_t P$.*

Definition 9 (Determinate process)**.** *A ground process P is determinate if whenever $(P,\phi) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T,\psi)$ and $(P,\phi) \stackrel{l_1,\ldots,l_n}{\Longrightarrow} (T',\psi')$ then $\psi \approx_s \psi'$.*

Intuitively, determinate processes are processes in which the adversarys static knowledge at any instance is completely determined by its past interaction with the protocol participants. Note that any ground trace is determinate.

Example(continued):Now we will consider a protocol based on the previously described equational theory E in 3.2 and we will try to model it in this process calculus. The protocol involves three participants A,B,C who want to share the sum of their salaries but not their individual salaries:

$A \to B : \{N_A + S_A\}_{pk_B}$
$B \to C : \{N_A + S_A + S_B\}_{pk_C}$
$C \to A : \{N_A + S_A + S_B + S_C\}_{pk_A}$
$A \to B, C : (S_A + S_B + S_C)$

where N_A is a nonce generated by A and S_A, S_B, S_C are individual salaries of A, B and C respectively. Formally we consider S_A, S_B, S_C to be nonces generated by the participants.

So A initiates the protocol and sends the sum of its salary and a nonce N_A to B. B adds his own salary to the sum and sends it to C. C does the same thing and sends it back to A. Finally A retrieves the sum of all salaries from what C sends and sends the sum to B and C. The protocol should ideally preserve the secrecy of S_A, S_B, S_C, that is no participant should be able to deduce the individual value of any other participant's salary.

But an attack on this protocol exists. Assume that B is a dishonest agent and has control over the network. So after receiving $N_A + S_A$, B will only add his salary and send $N_A + S_A + S_B$ to A while impersonating C. So A will send $S_A + S_B$ back to B from which B can deduce S_A.

Modelling the Protocol:

Now we will model this protocol in the described process calculus. We have $F = \{aence, adec, pk, +, -\}$ and $N = \{sk_A, sk_C, S_A, S_C, N_A\}$ and $C = \{c_A, c_B, c_C\}$(individual channel for each participant in order to have a determinate process). Also $M_B = \{sk_B, S_B\}$ which is the set of public names wrt. attacker B, the names that B knows. Each participant first outputs its public key for the benefit of everyone else. Let T_A, T_B, T_C be the sequence of actions in the protocols for A, B and C respectively, given by:

$$T_A : \mathbf{out}(c_A, pk(sk_A)).\mathbf{out}(c_A, aenc((N_A{+}S_A), pk(sk_B))).\mathbf{in}(c_A, x).\mathbf{test}[x \stackrel{?}{=} aenc(y, pk(sk_A))].\mathbf{out}(c_A, (adec(x, sk_A)){+}(-N_A))$$

$$T_B : \mathbf{out}(c_B, pk(sk_B)).\mathbf{in}(c_B, x).\mathbf{test}[x \stackrel{?}{=} aenc(y, pk(sk_B))].\mathbf{out}(c_B, aenc((adec(x, sk_B) + S_B), pk(sk_C))).\mathbf{in}(c_B, z)$$

$$T_C : \mathbf{out}(c_C, pk(sk_C)).\mathbf{in}(c_C, x).\mathbf{test}[x \stackrel{?}{=} aenc(y, pk(sk_C))].\mathbf{out}(c_C, aenc((adec(x, sk_C) + S_C), pk(sk_A))).\mathbf{in}(c_C, z)$$

So the whole protocol can be considered as a process P such that $P = \{T_A, T_B, T_C\}$

Let us consider T_A and describe the labelled transitions as per the semantics of the calculus in 5.2 for this trace. We will precisely describe the transitions that lead to the attack mentioned before. We assume that B is the dishonest agent or the attacker. Initially B's frame is an empty set. After the first SEND action, we have:

$$(T_A, \phi) \stackrel{out(c_A)}{\to} (T'_A, \psi_1 = \{\omega_1 \mapsto pk(sk_A)\})$$

where $T'_A = \mathbf{out}(c_A, aenc((N_A{+}S_A), pk(sk_B))).\mathbf{in}(c_A, x).\mathbf{test}[x \stackrel{?}{=} aenc(y, pk(sk_A))].\mathbf{out}(c_A, (adec(x, sk_A)){+}(-N_A))$ and after the second SEND action,

$$(T'_A, \{\omega_1 \mapsto pk(sk_A)\}) \stackrel{out(c_A)}{\to} (T''_A, \psi_2 = \{\omega_1 \mapsto pk(sk_A), \omega_2 \mapsto aenc((N_A + S_A), pk(sk_B))\})$$

where $T''_A = \mathbf{in}(c_A, x).\mathbf{test}[x \stackrel{?}{=} aenc(y, pk(sk_A))].\mathbf{out}(c_A, (adec(x, sk_A)) + (-N_A))$
Now B can retrieve the message $(N_A + S_A)$ since B has sk_B and $\psi_2 \vdash_E^{adec(\omega_2, sk_B)} (N_A + S_A)$. B now generates S_B and using the function symbols $+, aenc$, he can create the term $aenc((N_A + S_A + S_B), pk(sk_A))$. So

$$\psi_2 \vdash_E^R aenc((N_A + S_A + S_B), pk(sk_A))$$

where $R = aenc(adec((\omega_2, sk_B) + S_B), \omega_1)$. B puts this term for A to receive on c_A.
After the subsequent RECEIVE and TEST actions in T_A, we have:

$$(T_A", \psi_2) \stackrel{in(c_A, R).test}{\Longrightarrow} (T''_A\{x \mapsto aenc((N_A + S_A + S_B), pk(sk_A))\}, \psi_2)$$

and $T_A''' = (T_A''\{x \mapsto aenc((N_A + S_A + S_B), pk(sk_A))\} = \{\mathbf{out}(c_B, (adec(aenc((N_A + S_A + S_B), pk(sk_A)), sk_A)) + (-N_A))\} = \{\mathbf{out}(c_B, (N_A + S_A + S_B) + (-N_A))\}$
Finally after the last SEND action we get,

$$(T_A''', \psi_2) \overset{out(c_A)}{\Longrightarrow} (T_A''', \psi_3 = \{\omega_1 \mapsto aenc((N_A + S_A), pk_B), \omega_2 \mapsto aenc((N_A + S_A), pk(sk_B)), \omega_3 \mapsto (S_A + S_B)\})$$

So using function symbol $-$, B calculates $(-S_B)$ and finally deduces S_A:

$$\psi_3 \vdash_E^{\omega_3 + (-S_B)} S_A$$

Thus we have $(T_A, \phi) \overset{out(c_A).out(c_A).in(c_A,R).test.out(c_A)}{\longrightarrow} (\phi, \psi_3)$ and the attacker can derive S_A from ψ_3.

6 Modelling traces as Horn Clauses

Our procedure is based on a fully abstract modeling of a trace into first-order Horn clauses. We give the details of this modeling; we start by giving some definitions that we need for defining the predicates used in the logic.

6.1 Symbolic Labels, Runs and Recipes

We define the set of *symbolic labels* as

$$SLabels = \{in(c,t), out(c), test | t \in SMessages, c \in C\}$$

and the set of *symbolic runs* as the set of finite sequences of symbolic labels. The empty sequence is denoted by ϵ. We will often write ' '(empty space) for ϵ. Intuitively, a symbolic label stands for a set of possible labels, and a symbolic run stands for a set of possible runs of the protocol.

Symbolic Recipes: We assume a set Y of *recipe variables* disjoint from χ(Set of variables). The set of terms $T_{F,M,W,Y}$ shall be called *symbolic recipes* and denoted by **SRecipes**. We use capital letters $X, ..., Z$ to range over Y. Intuitively, a symbolic recipe stands for a set of recipes.

We can extend the definition of substitutions to include variables from Y in its domain. We only consider substitutions that map variables in Y to $SRecipes$. A ground substitution must map variables in Y to $Recipes$.

6.2 Semantics of the predicates

The predicates are interpreted over a triple- (a trace T, a frame φ and a substitution σ).
For ($l, l_i \in$ SLabels, $R \in$ SRecipes, $\omega \in SRuns$, $t \in$ SMessages, $T \in$ GndTraces, $\varphi \in$ Frames, σ a ground substitution):

- **Reachability Predicate**

  $(T, \varphi_0, \sigma) \models r_{l_1,..,l_n}$ if $(T, \varphi_0) \overset{L_1}{\to} (T_1, \varphi_1) \overset{L_2}{\to} ... \overset{L_n}{\to} (T_n, \varphi_n)$ such that

  $l_i\sigma =_E L_i\varphi_{i-1}$ for all $1 \leq i \leq n$

  This says that the run $\omega = l_1, l_2, ..., l_n$ is possible.

- **Intruder Knowledge Predicate**

  $(T, \varphi_0, \sigma) \models k_{l_1,..,l_n}(R, t)$ if when $(T, \varphi_0) \overset{L_1}{\to} (T_1, \varphi_1) \overset{L_2}{\to} ... \overset{L_n}{\to} (T_n, \varphi_n)$ such that

  $l_i\sigma =_E L_i\varphi_{i-1}$ for all $1 \leq i \leq n$, then $\left(\varphi_n \vdash_E^{R\sigma} t\sigma\right)$

  This says that if the run $\omega = l_1, l_2, ..., l_n$ is possible to execute, then intruder can construct the symbolic message t from the symbolic recipe R

- **Identity Predicate**

  $(T, \varphi_0, \sigma) \models i_{l_1,..,l_n}(R, R')$ if there exists t such that

  $(T, \varphi_0, \sigma) \models k_{l_1,..,l_n}(R, t)$ and $(T, \varphi_0, \sigma) \models k_{l_1,..,l_n}(R', t)$

  This says that if the run $\omega = l_1, l_2, ..., l_n$ executes then R and R' are the recipes for the same symbolic term.

- **Reachable Identity Predicate**

  $(T, \varphi_0, \sigma) \models ri_{l_1,..,l_n}(R, R')$ if

  $(T, \varphi_0, \sigma) \models r_{l_1,..,l_n}$ and $(T, \varphi_0, \sigma) \models i_{l_1,..,l_n}(R, R')$

  This says that the run $\omega = l_1, l_2, ..., l_n$ is possible and after the execution of the run, R and R' are the recipes for the same symbolic term.

Building formulas using connectives:
We consider first order formulas [Gal03] built using the above predicates. As in case of predicates, a formula f is interpreted over the triple consisting of a trace T, a frame φ and a substitution σ, that is $(T, \varphi, \sigma) \models f$. If σ is grounding then we can write $(T, \varphi) \models f$ and if $dom(\varphi) = \phi$ then we can simply write $T \models f$.

Let (T, φ, σ) be a triple over which we will interpret the formulas f_1 and f_2. The usual connectives can then be defined as follows,

- Conjunction
  $(T, \varphi, \sigma) \models (f_1 \wedge f_2)$ iff. $((T, \varphi, \sigma) \models f_1) \wedge ((T, \varphi, \sigma) \models f_2)$
- Disjunction
  $(T, \varphi, \sigma) \models (f_1 \vee f_2)$ iff. $((T, \varphi, \sigma) \models f_1) \vee ((T, \varphi, \sigma) \models f_2)$
- Negation
  $(T, \varphi, \sigma) \models (\sim f_1)$ iff. $\sim ((T, \varphi, \sigma) \models f_1)$
- Existential Quantification
  $(T, \varphi, \sigma) \models \exists x.f$ iff. there exists a term t such that $(T, \varphi, \sigma) \models f\{x \mapsto t\}$.
- Universal Quantification
  $(T, \varphi, \sigma) \models \forall x.f$ iff. for any term t, we have $(T, \varphi, \sigma) \models f\{x \mapsto t\}$.

6.3 Equality and Membership modulo an Equational theory

In most of the places, instead of having strict equality for two terms we can allow the two terms to equal modulo our non-convergent equational theory. Same holds for checking membership of a term wrt. any set. Let us formally define these concepts.

Definition 10. *For an equational theory E and any set S, we will say that for any term t_1, $t_1 \in_E S$ (membership modulo E) iff. there exists a term t_2 such that $t_2 \in S$ and $t_1 =_E t_2$.*

Let us now define equality and membership modulo E for the following,

- Substitutions: We say that $\sigma_1 =_E \sigma_2$ if $dom(\sigma_1) = dom(\sigma_2)$ and $\sigma_1(x) =_E \sigma_2(x)$ for all $x \in dom(\sigma_1) = dom(\sigma_2)$

  Also $\sigma_1 \in_E S$ iff. there exists a σ_2 such that $\sigma_2 \in S$ and $\sigma_1 =_E \sigma_2$.

- Symbolic Runs: Let $u = l_1, \ldots, l_n$ and $u' = l'_1, \ldots, l'_n$ be symbolic runs such that $l_i \in SLabels$ for all $1 \leq i \leq n$. We say that $u =_E u'$ iff. $l_i =_E l'_i$ for all $1 \leq i \leq n$.

  Also $u \in_E S$ iff. there exists a u' such that $u' \in S$ and $u =_E u'$.

- Predicates: (Let u and u' be symbolic runs)

1. $r_u =_E r_{u'}$ iff. $u =_E u'$ and $r_u \in_E S$ iff. there exists $r_{u'} \in S$ with $u =_E u'$.
2. $k_u(R,t) =_E k_{u'}(R',t')$ iff. $u =_E u'$, $R =_E R'$, $t =_E t'$ and $k_u(R,t) \in_E S$ iff. there exists $k_{u'}(R,t) \in S$ with $u =_E u'$, $R =_E R'$, $t =_E t'$.
3. $i_u(R_x,R_y) =_E i_{u'}(R'_x,R'_y)$ iff. $u =_E u'$, $R_x =_E R'_x$, $R_y =_E R'_y$ and $i_u(R_x,R_y) \in_E S$ iff. there exists $i_{u'}(R'_x,r'_y) \in S$ with $u =_E u'$, $R_x =_E R'_x$, $R_y =_E R'_y$.
4. $ri_u(R_x,R_y) =_E ri_{u'}(R'_x,R'_y)$ iff. $u =_E u'$, $R_x =_E R'_x$, $R_y =_E R'_y$ and $ri_u(R_x,R_y) \in_E S$ iff. there exists $ri_{u'}(R'_x,R'_y) \in S$ with $u =_E u'$, $R_x =_E R'_x$, $R_y =_E R'_y$.

- Statements: Let $f = H \Leftarrow B_1,\ldots,B_n$ and $f' = H' \Leftarrow B'_1,\ldots,B'_n$. Then $f =_E f'$ iff. $H =_E H'$ and $B_i =_E B'_i$ for all $1 \le i \le n$.

  Also $f \in_E S$ if there exists $f' \in S$ such that $f =_E f'$.

Proposition 1. *Given a trace T, frame φ and substitution σ, if we have that $(T,\varphi,\sigma) \models f$ and $f =_E f'$ for first order formulas f and f' then $(T,\varphi,\sigma) \models f'$.*

Proof: We will prove by induction on formulas f and f'. The base case is when $f = H$ and $f' = H'$ such that H and H' can be any of the four predicates, we will consider each case,

1. Let $H = r_{l_1,..,l_n}$ and since $f =_E f'$, we have $H' = r_{l'_1,..,l'_n}$. Given that $(T,\varphi,\sigma) \models r_{l_1,..,l_n}$, by the semantics of reachability, there exists $L_1,\ldots L_n$ such that,

   $$(T,\varphi) \xrightarrow{L_1} (T_1,\varphi_1) \xrightarrow{L_2} ..... \xrightarrow{L_n} (T_n,\varphi_n)$$

   and $l_i\sigma =_E L_i\varphi_{i-1}$ for all $1 \le i \le n$

   Since we have $l_i =_E l'_i$ for all $1 \le i \le n$ by previous definition, $l'_i\sigma =_E L_i\varphi_{i-1}$ for all $1 \le i \le n$ and hence

   $$(T,\varphi,\sigma) \models r_{l'_1,..,l'_n}$$

   implying that $(T,\varphi,\sigma) \models f'$ as we wanted to prove.

2. Let $H = k_{l_1,..,l_n}(R,t)$ and since $f =_E f'$, we have $H' = k_{l'_1,..,l'_n}(R',t')$. Given that $(T,\varphi,\sigma) \models k_{l_1,..,l_n}(R,t)$, by the semantics of knowledge, there exists $L_1,\ldots L_n$ such that if,

   $$(T,\varphi) \xrightarrow{L_1} (T_1,\varphi_1) \xrightarrow{L_2} ..... \xrightarrow{L_n} (T_n,\varphi_n)$$

   and $l_i\sigma =_E L_i\varphi_{i-1}$ for all $1 \le i \le n$ then $\left(\varphi_n \vdash^{R\sigma}_E t\sigma\right)$

   Since we have $l_i =_E l'_i$ for all $1 \le i \le n$ by previous definition, $l'_i\sigma =_E L_i\varphi_{i-1}$ for all $1 \le i \le n$ and since we have that $R'\sigma\varphi_n =_E t'\sigma$, we get $\left(\varphi_n \vdash^{R'\sigma}_E t'\sigma\right)$. Hence,

   $$(T,\varphi,\sigma) \models k_{l'_1,..,l'_n}(R',t')$$

   implying that $(T,\varphi,\sigma) \models f'$ as we wanted to prove.

3. Let $H = i_{l_1,..,l_n}(R,R')$ and since $f =_E f'$, we have $H' = i_{l'_1,..,l'_n}(S,S')$. Given that $(T,\varphi,\sigma) \models i_{l_1,..,l_n}(R,R')$, by the semantics of identity, we have that there exists t such that,

   $(T,\varphi,\sigma) \models k_{l_1,..,l_n}(R,t)$ and $(T,\varphi,\sigma) \models k_{l_1,..,l_n}(R',t)$ From the proof of the knowledge predicate we can say that $(T,\varphi,\sigma) \models k_{l'_1,..,l'_n}(S,t)$ and $(T,\varphi,\sigma) \models k_{l'_1,..,l'_n}(S',t)$ and hence,

   $$(T,\varphi,\sigma) \models i_{l'_1,..,l'_n}(S,S')$$

   implying that $(T,\varphi,\sigma) \models f'$ as we wanted to prove.

4. Let $H = ri_{l_1,..,l_n}(R, R')$ and since $f =_E f'$, we have $H' = ri_{l'_1,..,l'_n}(S, S')$. Given that $(T, \varphi, \sigma) \models ri_{l_1,..,l_n}(R, R')$, by the semantics of reachable identity, we have that,

   $(T, \varphi, \sigma) \models r_{l_1,..,l_n}$ and $(T, \varphi, \sigma) \models i_{l_1,..,l_n}(R, R')$. From the proof of the reachability and identity predicate we can say that $(T, \varphi, \sigma) \models r_{l'_1,..,l'_n}$ and $(T, \varphi, \sigma) \models i_{l'_1,..,l'_n}(S, S')$ and hence,

$$(T, \varphi, \sigma) \models ri_{l'_1,..,l'_n}(S, S')$$

   implying that $(T, \varphi, \sigma) \models f'$ as we wanted to prove.

Now that we have proved the base case for all the four predicates, we will prove inductively for each of the connectives. By the induction hypothesis we will assume that f and g be any two first order formulas for which our proposition holds. So, if we have $(T, \varphi, \sigma) \models f$, $(T, \varphi, \sigma) \models g$ and $f =_E f'$, $g =_E g'$ respectively, then we have $(T, \varphi, \sigma) \models f'$ and $(T, \varphi, \sigma) \models g'$.

We will prove for each connective,

- Conjunction: Let $(T, \varphi, \sigma) \models (f \wedge g)$, which means $((T, \varphi, \sigma) \models f) \wedge ((T, \varphi, \sigma) \models g)$ by the semantics of conjunction. But by induction hypothesis we have $((T, \varphi, \sigma) \models f') \wedge ((T, \varphi, \sigma) \models g')$ and hence, again by semantics $(T, \varphi, \sigma) \models (f' \wedge g')$

- Disjunction: Let $(T, \varphi, \sigma) \models (f \vee g)$, which means $((T, \varphi, \sigma) \models f) \vee ((T, \varphi, \sigma) \models g)$ by the semantics of conjunction. So at least one of the two things is true. If $((T, \varphi, \sigma) \models f)$ then by induction hypothesis we have $((T, \varphi, \sigma) \models f')$ and hence $((T, \varphi, \sigma) \models f') \vee ((T, \varphi, \sigma) \models g')$ is true. The same argument holds if $((T, \varphi, \sigma) \models g)$ and hence, again by semantics $(T, \varphi, \sigma) \models (f' \vee g')$

- Negation: Let $(T, \varphi, \sigma) \models (\sim f)$, which means $\sim ((T, \varphi, \sigma) \models f)$ by the semantics of negation. By induction hypothesis we have that if $(T, \varphi, \sigma) \models f$ and $f =_E f'$, then $(T, \varphi, \sigma) \models f'$. We can use the same hypothesis for f' and say that if $(T, \varphi, \sigma) \models f'$ and $f =_E f'$, then $(T, \varphi, \sigma) \models f$. So if we have $f =_E f'$, then we get

$$((T, \varphi, \sigma) \models f) \Leftrightarrow ((T, \varphi, \sigma) \models f')$$

  So, $\sim ((T, \varphi, \sigma) \models f)$ implies $\sim ((T, \varphi, \sigma) \models f')$ and hence, again by semantics $(T, \varphi, \sigma) \models (\sim f')$

- Existential Quantification: Let $(T, \varphi, \sigma) \models \exists x.f$ and let f'' be such that $\exists x.f =_E f''$. Hence f'' is also of the form $\exists x.f'$ for some $f' =_E f$.

  $(T, \varphi, \sigma) \models \exists x.f$ means there exists a term t such that $(T, \varphi, \sigma) \models f\{x \mapsto t\}$. Let t_1 be that term and let $f_1 = f\{x \mapsto t_1\}$. Also let $f'_1 = f'\{x \mapsto t_1\}$, so we have $f_1 =_E f'_1$. Therefore, by induction hypothesis, $(T, \varphi, \sigma) \models f'_1$. So there exists such a term for f' also since $(T, \varphi, \sigma) \models f'\{x \mapsto t_1\}$. Hence we get $(T, \varphi, \sigma) \models f''$.

- Universal Quantification: Let $(T, \varphi, \sigma) \models \forall x.f$ and let f'' be such that $\forall x.f =_E f''$. Hence f'' is also of the form $\forall x.f'$ for some $f' =_E f$.

  $(T, \varphi, \sigma) \models \forall x.f$ means that for any term t, we have $(T, \varphi, \sigma) \models f\{x \mapsto t\}$. Let t_1 be any term and let $f_1 = f\{x \mapsto t_1\}$. Also let $f'_1 = f'\{x \mapsto t_1\}$, so we have $f_1 =_E f'_1$. Therefore, by induction hypothesis, $(T, \varphi, \sigma) \models f'_1$. Since t_1 was chosen arbitrarily, $(T, \varphi, \sigma) \models f'\{x \mapsto t\}$ for any term t . Hence we get $(T, \varphi, \sigma) \models f''$.

Hence proved.

7 The Seed Statements

Given a trace T, we will give a set of seed statements $seed(T)$ which will serve as a starting point for our modelling. We shall also establish that the set $seed(T)$ is a sound and (partially) complete abstraction of tha trace T.

Definition 11 (Statement)**.** *A Horn Clause of the form $H \Leftarrow B_1, \ldots, B_n$ is called a* Statement *if,*

1. $H \in \{r_u, k_u(R, t), i_u(R, R'), ri_u(R, R')\}$

2. *For each* $1 \leq i \leq n$, $B_i = k_{u_i}(X_i, t_i)$

where, $u, u_i \in SLabels$, $t, t_i \in SMessages$, $R, R' \in SRecipes$ *and* $X_i \in$ *Recipe Variables for all* $1 \leq i \leq n$.

We implicitly assume that in a Horn Clause, all variables are universally quantified. Hence, all statements are closed formulas. Also $X_1, \ldots, X_n$ are distinct variables.

7.1 Some Conventions

In order to formally define $seed(T)$, we start by fixing some conventions. Let $T = a_1.a_2....a_n$ be a ground trace. We assume the following naming conventions:

1. If a_i is a receive action then $a_i = in(c_i, x_i)$
2. $x_i \neq x_j$ for any $i \neq j$
3. If a_i is a send action then $a_i = out(c_i, t_i)$
4. If a_i is a test action then $a_i = [s_i \stackrel{?}{=} t_i]$

Moreover, for each $1 \leq i \leq n$ let $l_i \in SLabels$ be as follows,

$$l_i = \begin{cases} in(c_i, x_i) & \text{if } a_i = in(c_i, x_i) \\ out(c_i) & \text{if } a_i = out(c_i, t_i) \\ test & \text{if } a_i = [s_i \stackrel{?}{=} t_i] \end{cases}$$

For each $0 \leq m \leq n$, let the sets $R(m)$, $S(m)$ and $T(m)$ respectively denote the indices of the receive, send and test actions amongst $a_1.a_2....a_m$. Formally,

$$\begin{cases} R(m) = \{i | 1 \leq i \leq m, a_i = in(c_i, x_i)\} \\ S(m) = \{i | 1 \leq i \leq m, a_i = out(c_i, t_i)\} \\ T(m) = \{i | 1 \leq i \leq m, a_i = [s_i \stackrel{?}{=} t_i]\} \end{cases}$$

Given a set of public names $M_0 \subseteq M$, the *set of seed statements* associated to T and M_0, denoted by $seed(T, M_0)$, is defined to be the set of following statements. If $M_0 = M$, then $seed(T, M)$ is simply denoted by $seed(T)$.

7.2 Set of Seed Statements and H(K)

Definition 12. *For a ground trace* $T = a_1.a_2...a_n$ *the set of seed statements is,*

1. $r_{l_1\sigma\tau\downarrow,\ldots,l_m\sigma\tau\downarrow} \Leftarrow \{k_{l_1\sigma\tau\downarrow,\ldots,l_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\}_{j \in R(m)}$
   for all $0 \leq m \leq n$
   for all $\sigma \in mgu_E \left\{(s_k \stackrel{?}{=} t_k)\right\}_{k \in T(m)}$
   for all $\tau \in \boldsymbol{variants}_E(l_1\sigma, \ldots, l_m\sigma)$
2. $k_{l_1\tau\downarrow,\ldots,l_m\tau\downarrow}(w_{|S(m)|}, l_m\tau\downarrow) \Leftarrow k_{l_1\tau,\ldots,l_{j-1}\tau\downarrow}(X_j, x_j\tau\downarrow)\}_{j \in R(m)}$
   for all $m \in S(n)$
   for all $\tau \in \boldsymbol{variants}_E(l_1, \ldots, l_m, t_m)$
3. $k(c, c) \Leftarrow$
   for all public names $c \in M_o$
4. $k_{l_1,\ldots,l_m}(f(Y_1, \ldots, Y_k), f(y_1, \ldots, y_k)\tau\downarrow) \Leftarrow \{k_{l_1,\ldots,l_m}(Y_j, y_j\tau\downarrow)\}_{j \in \{1,\ldots,k\}}$
   for all $0 \leq m \leq n$
   for all function symbols f *of arity* k
   for all $\tau \in \boldsymbol{variants}_E(f(y_1, \ldots, y_k))$

Example(continued):For our protocol as described in 5.3, we will find seed statements for T_A wrt. B, that is $seed(T_A, M_B)$. We have

$$T_A : \mathbf{out}(c_A, pk(sk_A)).\mathbf{out}(c_A, aenc((N_A{+}S_A), pk(sk_B))).\mathbf{in}(c_A, x).\mathbf{test}[x \stackrel{?}{=} aenc(y, pk(sk_A))].\mathbf{out}(c_A, (adec(x, sk_A)){+}(-N_A))$$

The corresponding symbolic run $(l_1.l_2.l_3.l_4.l_5)$ is:

$$\mathbf{out}(c_A).\mathbf{out}(c_A)\mathbf{in}(c_A, x).\mathbf{test}.\mathbf{out}(c_A)$$

Also, $R(5) = \{3\}$(indices of RECEIVE actions), $S(5) = \{1, 2, 5\}$(indices of SEND actions) and $T(5) = \{4\}$(indices of TEST actions).
We will consider examples of seed statements of each type:

1. $r_{out(c_A)\sigma\tau\downarrow.out(c_A)\sigma\tau\downarrow.in(c_A,x)\sigma\tau\downarrow.test\sigma\tau\downarrow.out(c_A)\sigma\tau\downarrow} \Leftarrow k_{out(c_A)\sigma\tau\downarrow.out(c_A)\sigma\tau\downarrow}(X, x\sigma\tau\downarrow)$
   for all $\sigma \in mgu_E\{x \stackrel{?}{=} aenc(y, pk(sk_A))\}$
   for all $\tau \in \mathbf{variants}_E(out(c_A)\sigma.out(c_A)\sigma.in(c_A, x)\sigma.(test)\sigma.out(c_A)\sigma)$

   Now $mgu_E\{x \stackrel{?}{=} aenc(y, pk(sk_A))\} = \{\sigma\}$ where $\sigma = \{x \mapsto aenc(y, pk(sk_A))\}$.
   Also $\mathbf{variants}_E(out(c_A)\sigma.out(c_A)\sigma.in(c_A, x)\sigma.(test)\sigma.out(c_A)\sigma)$
   $= \mathbf{variants}_E(out(c_A).in(c_A, aenc(y, pk(sk_A))).(test).out(c_A).out(c_A)) = \{\tau\}$ where τ is the identity substitution.

   So now that we know the substitutions σ and τ, the seed statement becomes:

   $$r_{out(c_A).out(c_A).in(c_A,aenc(y\downarrow,pk(sk_A))).test.out(c_A)} \Leftarrow k_{out(c_A).out(c_A)}(X, aenc(y, pk(sk_A)))$$

2. $k_{out(c_A)\tau\downarrow}(\omega_1, pk(sk_A))\tau\downarrow) \Leftarrow$
   for all $\tau \in \mathbf{variants}_E(out(c_A), pk(sk_A))$

   Now $\mathbf{variants}_E(out(c_A), pk(sk_A)) = \{\tau\}$ where τ is the identity substitution. So the seed statement becomes:

   $$k_{out(c_A)}(\omega_1, pk(sk_A)) \Leftarrow$$

   Similarly we get
   $$k_{out(c_A)}(\omega_2, aenc((N_A + S_A), pk(sk_B))) \Leftarrow$$

3. $k(sk_B, sk_B) \Leftarrow$
   $k(S_B, S_B) \Leftarrow$
   where $sk_B, S_B \in M_B$
   B can always derive any of the public names.

4. $k(adec(Y_1, Y_2), adec(y_1, y_2)\tau\downarrow) \Leftarrow k(Y_1, y_1\tau\downarrow) \wedge k(Y_2, y_2\tau\downarrow)$
   for $\tau \in \mathbf{variants}_E(adec(y_1, y_2))$ for the function $adec \in$ the signature F.

   We know $\mathbf{variants}_E(adec(y_1, y_2) = \{\sigma_1 = \{y_1 \mapsto aenc(s, pk(sk_z)), y_2 \mapsto sk_z\}, \sigma_2 = \{\}\}$ for some tern s as we calculated in 3.2.1.

   So the corresponding seed statements are:

   $$k(adec(Y_1, Y_2), s\downarrow) \Leftarrow k(Y_1, aenc(s\downarrow, pk(sk_z))) \wedge k(Y_2, sk_z)$$

   $$k(adec(Y_1, Y_2), adec(y_1, y_2)) \Leftarrow k(Y_1, y_1) \wedge k(Y_2, y_2)$$

   Similarly the attacker can use this type of seed statements to calculate other functions like $+, - \in F$:

   $$k(Y_1 + Y_2, y_1 + y_2) \Leftarrow k(Y_1, y_1) \wedge k(Y_2, y_2)$$

   $$k(-(Y_1), -(y_1)) \Leftarrow k(Y_1, y_1)$$

Definition 13. *Let K be a set of statements. We define $H(K)$ to be the smallest set of ground terms such that,*

- *SIMPLE CONSEQUENCE:*
$$\frac{f = (H \Leftarrow B_1, \ldots, B_n) \in K \qquad \sigma \text{ grounding for } f \qquad B_1\sigma \in_{E'} H(K) \qquad \ldots \qquad B_n\sigma \in_{E'} H(K)}{H\sigma \in H(K)}$$
- *EXTENDK:* $\dfrac{k_u(R,t) \in H(K)}{k_{uv}(R,t) \in H(K)}$
- *EXTENDI:* $\dfrac{i_u(R,R') \in H(K)}{i_{uv}(R,R') \in H(K)}$

7.3 Abstraction of the Trace

Theorem 1. *Let T be a ground trace.*

- *(Soundness) For any statement $f \in seed(T) \cup H(seed(T)), T \models f$*
- *(Completeness) If $(T,\phi) \xrightarrow{L_1,\ldots,L_n} (S,\varphi)$ then :*
  1. $r_{L_1\varphi\downarrow,..,L_i\varphi\downarrow} \in_{E'} H(seed(T))$
  2. *if $\varphi \vdash_E^R t$ then $k_{L_1\varphi\downarrow,..,L_i\varphi\downarrow}(R, t\downarrow) \in_{E'} H(seed(T))$*

We will now illustrate the completeness of the seed statements with the help of our running example. We had shown in 5.3 that

$$(T_A,\phi) \xrightarrow{out(c_A).out(c_A).in(c_A,R).test.out(c_A)} (\phi,\psi_3)$$

and that $\psi_3 \vdash_E^{\omega_3+(-S_B)} S_A$. We will show that

$$r_{out(c_A)\psi_3\downarrow.out(c_A)\psi_3\downarrow.in(c_A,R)\psi_3\downarrow.test\psi_3\downarrow.out(c_A)\psi_3\downarrow} \in_{E'} H(seed(T_A,M_B))$$

and

$$k_{out(c_A)\psi_3\downarrow.out(c_A)\psi_3\downarrow.in(c_A,R)\psi_3\downarrow.test\psi_3\downarrow.out(c_A)\psi_3\downarrow}(\omega_3+(-S_B),S_A) \in_{E'} H(seed(T_A,M_B))$$

We will use the seed statements computed in 7.2 and the definition of $H(K)$.

- We have $(k_{out(c_A)}(\omega_2, aenc((N_A+S_A), pk(sk_B))) \Leftarrow)$ and $(k(sk_B,sk_B) \Leftarrow)$. So using the seed statement

  $$k_{out(c_A)}(adec(\omega_2,sk_B),(N_A+S_A)) \Leftarrow k_{out(c_A)}(\omega_2, aenc((N_A+S_A),pk(sk_B))) \wedge k_{out(c_A)}(sk_B,sk_B)$$

  and the rule of SIMPLE CONSEQUENCE (with σ as the identity substitution), we get

  $$k_{out(c_A)}(adec(\omega_2,sk_B),(N_A+S_A)) \in H(seed(T_A,M_B))$$

  So this models how B is able to retrieve the term (N_A+S_A) from the message outputted by A.
- Now we have $k_{out(c_A)}(adec(\omega_2,sk_B)+S_B,(N_A+S_A+S_B)) \in H(seed(T_A,M_B))$ using

  $k_{out(c_A)}(adec(\omega_2,sk_B)+S_B,(N_A+S_A+S_B)) \Leftarrow k_{out(c_A)}(adec(\omega_2,sk_B),(N_A+S_A) \wedge k_{out(c_A)}(S_B,S_B) \in (seed(T_A,M_B)$

  and the identity substitution again for SIMPLE CONSEQUENCE.
  This models how B is able to generate the term $(N_A+S_A+S_B)$.
- Since we have $k_{out(c_A)}(\omega_1, pk(sk_A))$ and the seed statement for the encryption function, we obtain

  $$k_{out(c_A)}(aenc(S,\omega_1), aenc((N_A+S_A+S_B),pk(sk_A))) \in H(seed(T_A,M_B))$$

  where $S = adec(\omega_2,sk_B)+S_B$. This models how B encrypts the term to be given as input for A.

- By the EXTENDK rule we get

$$k_{out(c_A).out(c_A)}(aenc(S,\omega_1),u) \in H(seed(T_A,M_B))$$

where $u = aenc((N_A + S_A + S_B), pk(sk_A))$ and $R\psi_3 \downarrow= u$ where $R = aenc(adec((\omega_2, sk_B) + S_B), \omega_1)$ and $\psi_3 = \{\omega_1 \mapsto aenc((N_A + S_A), pk_B), \omega_2 \mapsto aenc((N_A + S_A), pk(sk_B)), \omega_3 \mapsto (S_A + S_B)\}$. So using the seed statement

$$r_{out(c_A).out(c_A).in(c_A,u).test.out(c_A)} \Leftarrow k_{out(c_A).out(c_A)}(aenc(S, omega_1), u)$$

we get the desired result that

$$r_{out(c_A).out(c_A).in(c_A,u).test.out(c_A)} \in H(seed(T_A,M_B))$$

- Finally we use the seed statement of type 2

$$k_{out(c_A).out(c_A).in(c_A,u).test.out(c_A)}(\omega_3, (adec(z, sk_A) + (-N_A))\downarrow) \Leftarrow k_{out(c_A).out(c_A)}(aenc(S, omega_1), z)\downarrow)$$

where $z = aenc((N_A + S_A + S_B), pk(sk_A))$. So

$$(adec(z, sk_A) + (-N_A))\downarrow= ((N_A + S_A + S_B) + (-N_A))\downarrow= (S_A + S_B)$$

So $k_{out(c_A).out(c_A).in(c_A,R).test.out(c_A)}(\omega_3, (S_A + S_B)) \in H(seed(T_A, M_B))$ and we use the following seed statement to get the final result in our set

$k_{out(c_A).out(c_A).in(c_A,u).test.out(c_A)}(\omega_3 + (-S_B), (S_A + S_B + (-S_B))\downarrow)$
$\Leftarrow k_{out(c_A).out(c_A).in(c_A,u).test.out(c_A)}(\omega_3, (S_A + S_B)) \wedge k_{out(c_A).out(c_A).in(c_A,u).test.out(c_A)}(-(S_B), -(S_B))$
and get

$$k_{out(c_A).out(c_A).in(c_A,u).test.out(c_A)}(\omega_3 + (-S_B), S_A) \in H(seed(T_A, M_B))$$

as we wanted to show.

7.4 Soundness proofs

We will now formally prove the above stated theorem.
Proof: First we prove the soundness of the seed statements. We prove that for each statement $f \in seed(T)$, we have that $T \models f$. There are four kinds of seed statements and so we will prove for each type,

1. Let m be such that $0 \leq m \leq n$ and let σ, τ be substitutions such that $\sigma \in mgu_E \left\{(s_i \stackrel{?}{=} t_i)\right\}_{i \in T(m)}$ and $\tau \in variants(l_1\sigma, \ldots, l_m\sigma)$. We show that,

$$f = ((r_{l_1\sigma\tau\downarrow,\ldots,l_m\sigma\tau\downarrow} \Leftarrow \left\{k_{l_1\sigma\tau\downarrow,\ldots,l_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\right\}_{j \in R(m)}))$$

is a statement true in T.

Let ω be an arbitrary substitution, grounding for f. Assume also that $T \models (k_{l_1\sigma\tau\downarrow,\ldots,l_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow))\omega$ for all $j \in R(m)$. We show that $T \models (r_{l_1\sigma\tau\downarrow,\ldots,l_m\sigma\tau\downarrow})\omega$. In fact, we will show a stronger statement that for all p such that $0 \leq p \leq m$,

$$T \models ((r_{l_1\sigma\tau\downarrow,\ldots,l_p\sigma\tau\downarrow})\omega)$$

We proceed by induction on p
Base Case: For $p = 0$, we have $(r_{l_1\sigma\tau\downarrow,\ldots,l_p\sigma\tau\downarrow})\omega = \mathbf{r}_\epsilon$ which is the empty run. This holds trivially for any trace T.

Inductive Case: For $p > 0$, assume that $T \models (r_{l_1\sigma\tau\downarrow,\ldots,l_{p-1}\sigma\tau\downarrow})\omega$ and we have to prove $T \models (r_{l_1\sigma\tau\downarrow_R,\ldots,l_p\sigma\tau\downarrow_R})\omega$. We consider different cases for the p^{th} label (SEND, TEST or RECEIVE) in the trace and proceed with the case analysis.

First, we need to fix some notations. Let $T_1 = T$ and $\varphi_1 = \phi$ where (T, ϕ) is the initial trace and frame. As $T \models (r_{l_1\sigma\tau\downarrow,\ldots,l_{p-1}\sigma\tau\downarrow})\omega$, we have that there exist $L_1, \ldots, L_{p-1}$ such that

$$(T_i, \varphi_i) \xrightarrow{L_i} (T_{i+1}, \varphi_{i+1})$$

and $(l_i\sigma\tau)\omega =_E L_i\varphi_i$ for all $1 \leq i < p$ where $T_i = (a_i \ldots a_n)\{x_j \mapsto x_j\sigma\tau \downarrow_R \omega\}_{j\in R(i-1)}$ and where φ_i extends φ_{i-1} for all i. We can now do the case analysis-

- (SEND) If $a_p = \mathbf{out}(c_p, t_p)$, then $l_p = \mathbf{out}(c_p)$ by definition. Let $T_{p+1} = (a_{p+1} \ldots a_n)\{x_j \mapsto x_j\sigma\tau \downarrow \omega\}_{j\in R(p)}$ and let $\varphi_{p+1} = \varphi_p \cup \{\omega_{dom(\varphi_p)+1} \mapsto t_p\sigma\tau \downarrow \omega\}$. Let $L_p = \mathbf{out}(c_p)$, then by definition we have-

  $$(T_p, \varphi_p) \xrightarrow{L_p} (T_{p+1}, \varphi_{p+1})$$

  This is what we wanted to prove.

- (TEST) If $a_p = [s_p \stackrel{?}{=} t_p]$, then $l_p = \mathbf{test}$. Let $T_{p+1} = (a_{p+1} \ldots a_n)\{x_j \mapsto x_j\sigma\tau \downarrow \omega\}_{j\in R(p)}$ and let $\varphi_{p+1} = \varphi_p$. As $\sigma \in mgu_E(\left\{s_k \stackrel{?}{=} t_k\right\}_{k\in T(m)})$, we have that $s_p\sigma =_E t_p\sigma$ and therefore

  $$(s_p\sigma\tau)\omega =_E (t_p\sigma\tau)\omega$$

  . Hence, $(T_p, \varphi_p) \xrightarrow{\mathbf{test}} (T_{p+1}, \varphi_{p+1})$ as we wanted to prove.

- (RECEIVE) If $a_p = \mathbf{in}(c_p, x_p)$, we know that $p \in R(p)$. Let $T_{p+1} = (a_{p+1} \ldots a_n)\{x_j \mapsto x_j\sigma\tau \downarrow \omega\}_{j\in R(p)}$ and let $\varphi_{p+1} = \varphi_p$. As $p \in R(p)$, we have that $T \models ((k_{l_1\sigma\tau\downarrow,\ldots,l_{p-1}\sigma\tau\downarrow}(X_p, x_p\sigma\tau \downarrow))\omega$. Therefore $\varphi_p \vdash_E^{X_p\omega} x_p\sigma\tau \downarrow \omega$ and, by letting $L_p = \mathbf{in}(c_p, X_p\omega)$, by definition we obtain-

  $$(T_p, \varphi_p) \xrightarrow{L_p} (T_{p+1}, \varphi_{p+1})$$

  as we wanted to prove.

2. Let $m \in S(n)$ and $\sigma \in variants(l_1, \ldots, l_m, t_m)$. We show that the statement

   $$f = (k_{l_1\sigma\downarrow,\ldots,l_m\sigma\downarrow}(\omega_{|S(m)|}, (t_m\sigma)\downarrow) \Leftarrow \left\{k_{l_1\sigma\downarrow,\ldots,l_{j-1}\sigma\downarrow}(X_j, x_j\sigma\downarrow)\right\}_{j\in R(m)}$$

   is true in T. Let w be a substitution grounding for f. We assume that $T \models (k_{l_1\sigma\downarrow,\ldots,l_{j-1}\sigma\downarrow}(X_j, x_j\sigma\downarrow))w$ for all $j \in R(m)$ and we show that $T \models (k_{l_1\sigma\downarrow,\ldots,l_m\sigma\downarrow}(\omega_{|S(m)|}, (t_m\sigma)\downarrow))w$

   Let $T_i = (a_{i+1} \ldots a_n)\{x_j \mapsto x_j\sigma w\}_{j\in R(i-1)}$ and $\varphi_i = \cup_{1\leq j\leq |S(i-1)|}\{\omega_j \mapsto t_{o(j)}\sigma w\}$ where $o(j)$ denotes the index of the j^{th} SEND action.

   We distinguish two cases:

   - If there exist $L_1, \ldots, L_m$ such that $(T_1, \varphi_1) \xrightarrow{L_1} (T_2, \varphi_2) \xrightarrow{L_2} \ldots \xrightarrow{L_m} (T_{m+1}, \varphi_{m+1})$ such that

     $(l_i\sigma)w =_E L_i\varphi_i$ for all $1 \leq i \leq m$, we have that

     $$\varphi_m(\omega_{|S(m)|}) = t_m\sigma w$$

     and we have that $(\varphi \vdash_E^{\omega_{|S(m)|}} t_m\sigma w)$ and therefore $(\varphi \vdash_E^{\omega_{|S(m)|}} (t_m\sigma) \downarrow_{(R,E')} w)$ which implies $T \models (k_{l_1\sigma\downarrow,\ldots,l_m\sigma\downarrow}(\omega_{|S(m)|}, (t_m\sigma)\downarrow))w$
   - Otherwise, it trivially holds that $T \models (k_{l_1\sigma\downarrow,\ldots,l_m\sigma\downarrow}(\omega_{|S(m)|}, (t_m\sigma)\downarrow))w$

3. If c is a public name, we have that $f = (k(c, c) \Leftarrow)$ is true in T because $\phi \vdash_E^c c$

4. Let g be a function symbol of arity k and let $\sigma \in variants(g(x_1, \ldots, x_k))$. We show that the following statement is true in T,

$$f = (k_{l_1,\ldots,l_m}(g(X_1, \ldots, X_k), g(x_1, \ldots, x_k)\sigma\downarrow) \Leftarrow \{k_{l_1,\ldots,l_m}(X_j, x_j\sigma\downarrow)\}_{j\in\{1,\ldots,k\}})$$

for any run $l_1, \ldots, l_m$. Let w be a substitution grounding for f. We assume that $T \models k_{l_1,\ldots,l_m}(X_j, x_j\sigma\downarrow)w$ for all $1 \leq j \leq k$.

So by our hypothesis, if there exist $L_1, \ldots, L_{n-1} \in Labels$ such that

$$(T, \phi) \xrightarrow{L_1} (T_2, \varphi_2) \xrightarrow{L_2} ..... \xrightarrow{L_{n-1}} (T_n, \varphi_n)$$

and $l_i\sigma =_E L_i\varphi_{i-1}$ for all $1 \leq i \leq n$, then $X_j\omega$ is the recipe for $(x_j\sigma)\downarrow w$ for all $1 \leq j \leq k$, by sematics of the knowedge predicate. Hence we have that $\psi_n \vdash_E^{X_j w} (x_j\sigma)\downarrow w$ for all $1 \leq j \leq k$. But this implies

$$\psi_n \vdash_E^{g(X_1 w,\ldots,X_k w)} g((x_1\sigma)\downarrow w, ..., (x_k\sigma)\downarrow w) = g(x_1, ..., x_k)\sigma)\downarrow w$$

which immediately implies $T \models (k_{l_1,\ldots,l_m}(g(X_1, \ldots, X_k), g(x_1, \ldots, x_k)\sigma\downarrow))w$

If there does not exist such $L_1, \ldots, L_{n-1}$ then by semantics of the knowledge predicate, $T \models (k_{l_1,\ldots,l_m}(g(X_1, \ldots, X_k), g(x_1, ..$ $))w$ holds.

Hence, we have shown that for every statement $f \in seed(T)$, $T \models f$.

Soundness of H(seed(T)):

Proposition: *Let T be a ground trace and K be a set of statements such that for all $f \in K$ we have that $T \models f$. Then for all $f \in H(K)$ we also have that $T \models f$*

Proof:

This proof will again use induction on the size of smallest proof of $f \in H(K)$.

Base Case: Consider any statement $f' \in K$ such that $(n = 0)$ for the rule of SIMPLE CONSEQUENCE. Therefore $f' = (H \Leftarrow)$ and let $f = f'\sigma$ where σ is a grounding substitution for f'. By the rule, we have that $f \in H(K)$

Now, since $f' \in K$, we have that $T \models f'$ and as variables in f' are universally quantified, we have that $T \models f'\sigma$, that is $T \models f$ which is what we wanted to prove.

Inductive Case: We have different cases depending on the last rule applied in the proof of f.

- SIMPLE CONSEQUENCE: Let $f' = (H \Leftarrow B_1, \ldots, B_n)$ and we have $T \models f'$. Let σ be a substitution grounding for f'. So by the definition of the rule we have that $B_i\sigma \in_{E'} H(K)$, that is there exists C_i such that $B_i\sigma =_E C_i$ and $C_i \in H(K)$. By induction hypothesis, we have that since $C_i \in H(K)$, $T \models C_i$ for all i. By proposition 1 we get that $T \models B_i\sigma$ for all i. So we have $T \models f$ where $f = H\sigma$.
- EXTENDK: We have $k_u(R, t) \in H(K)$ and by induction hypothesis we have $T \models k_u(R, t)$. So by the semantics of the knowledge predicate, $T \models k_{uv}(R, t)$.
- EXTENDI: Similar to EXTENDK

7.5 Completeness proof

Now we prove the **completeness of the seed statements**. We have to prove-

Let T and S be traces and let φ be a frame. If $(T, \phi) \xrightarrow{L_1,\ldots,L_n} (S, \varphi)$ then :

- $r_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow} \in_{E'} H(seed(T))$
- if $\varphi \vdash_E^R t$ then $k_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in_{E'} H(seed(T))$

Proof: We prove both statements by induction on n. For the base case we assume the empty run, that is, $(T,\phi) \longrightarrow (T,\phi)$. Hence by definition we have that $r_\epsilon \in H(seed(T)$.

Also if $\phi \vdash_E^R t$ then t is a term formed by combination of public constants and function symbols and $R =_E t$. So let us prove this base case again by induction on the size of R. For the base case of this induction, we consider the following cases,

1. If $R = c$ is a public name, we have that the statement $f = (k(c,c) \Leftarrow)$ is in the set of seed statements by definition. So, $k(R, t\downarrow) = (k(c,c) \in H(seed(T))$ by definition.

   If $R = f(R_1, \ldots, R_k)$, let γ be a substitution with $dom(\gamma) \subseteq \{y_1, \ldots, y_k\}$ that maps y_j to $R_j\varphi\downarrow$ for all $1 \leq j \leq k$

   By definition of seed knowledge base, we have that,

   $$((k(f(Y_1,\ldots,Y_k), f(y_1,\ldots,y_k)\tau_v\downarrow) \Leftarrow \{k(Y_j, y_j\tau_v\downarrow)\}_{j\in\{1,\ldots,k\}})) \in seed(T)$$

   for all $\tau_v \in V$ where V is one of the possible sets of $\mathbf{variants}_E(f(y_1,\ldots,y_k))$

   By definition of variants, there exist $\tau \in V$ and τ' such that $(f(y_1,\ldots,y_k)\gamma)\downarrow =_{E'} (f(y_1,\ldots,y_k)\tau)\downarrow\tau'$. Also $(f(y_1,\ldots,y_k)\gamma)\downarrow = R\varphi\downarrow$. Hence, $R\varphi\downarrow =_{E'} (f(y_1,\ldots,y_k)\tau)\downarrow\tau'$.

   Let

   $$g = ((k(f(Y_1,\ldots,Y_k), f(y_1,\ldots,y_k)\tau\downarrow) \Leftarrow \{k(Y_j, y_j\tau\downarrow)\}_{j\in\{1,\ldots,k\}}))$$

   Let $\tau'' = \omega\downarrow \cup \tau' \cup \{Y_j \mapsto R_j\}_{j\in\{1,\ldots,k\}}$. We have that all antecedents of $g\tau''$ are in $H(seed(T))$ by induction hypothesis. This is because, by definition of γ and that of $\mathbf{variants}_E(f(y_1,\ldots,y_k))$, we get $y_j\gamma\downarrow =_{E'} y_j\tau\downarrow \tau' =_{E'} R_j\varphi\downarrow$ and so we have that R_j is the recipe for $y_j\tau\downarrow\tau'$. Hence the head of $g\tau''$ is also in $H(seed(T))$.

   Hence, for any size of R such that $R =_E t$, we have $k_\epsilon(R, t\downarrow) \in H(seed(T))$.

We assume that the statements hold for any index of a run less than n. As $(T,\phi) \xrightarrow{L_1,\ldots,L_n} (S,\varphi)$, we have that there exists a substitution ω such that $(s_k\omega)\downarrow =_{E'} (t_k\omega)\downarrow$ for all $k \in T(n)$ and $L_1\varphi\downarrow, \ldots, L_n\varphi\downarrow = (l_1,\ldots,l_n)\omega$ for ω

We prove for each of the statements,

- As $(s_k\omega)\downarrow =_{E'} (t_k\omega)\downarrow$ for all $k \in T(n)$, it follows from definition that there exists the set $mgu_E\left\{(s_i \stackrel{?}{=} t_i)\right\}_{i\in T(n)}$.

  Also by definition of $seed(T)$ we have that

  $$((r_{l_1\sigma_M\tau_V\downarrow,\ldots,l_n\sigma_M\tau_V\downarrow} \Leftarrow \left\{k_{l_1\sigma_M\tau_V\downarrow,\ldots,l_{j-1}\sigma_M\tau_V\downarrow}(X_j, x_j\sigma_M\tau_V\downarrow)\right\}_{j\in R(n)})) \in seed(T)$$

  for all $\sigma_M \in M$ where M is one of the possible sets arising from the computation of $mgu_E\left\{(s_k \stackrel{?}{=} t_k)\right\}_{k\in T(m)}$

  for all $\tau_V \in V$ where V is one of the possible sets arising from the computation of $\mathbf{variants}_E(l_1\sigma,\ldots,l_m\sigma)$

  By definition of mgu_E, given a substitution ω, there exists a $\sigma \in M$ such that,

  (a) $dom(\sigma) \subseteq X$
  (b) $(s_k\sigma) =_E (t_k\sigma)$ for all $k \in T(n)$
  (c) $\omega[X] =_\prime (\sigma\pi)[X]$ for some substitution π

  where $X = vars(\{s_k, t_k\}_{k\in T(n)})$

  It follows that $(l_1,\ldots,l_n)\omega =_E (l_1,\ldots,l_n)\sigma\pi$ for some substitution π.

  By definition of $variants((l_1,\ldots,l_n)\sigma)$, we have that there exists $\tau \in V$ such that $(l_1,\ldots,l_n)\sigma\pi\downarrow_{(R,E')} =_{E'} ((l_1,\ldots,l_n)\sigma\tau\downarrow_{(R,E')})\tau'$ for some τ'.

  Let

  $$f = ((r_{l_1\sigma\tau\downarrow,\ldots,l_n\sigma\tau\downarrow} \Leftarrow \left\{k_{l_1\sigma\tau\downarrow,\ldots,l_{j-1}\sigma\tau\downarrow}(X_j, x_j\sigma\tau\downarrow)\right\}_{j\in R(n)}))$$

  Let τ" be the substitution obtained by extending τ' such that $\{X_j \mapsto R_j\}$ for all $j \in R(n)$ where $R_j s$ are recipes for $x_j\omega$. Such $R_j s$ exist because since we have that X_j is the recipe for $x_j\sigma\tau\downarrow$ for all $j \in R(n)$ and hence by semantics of RECEIVE in the proces calculus, $X_j\tau'$ is the recipe for $x_j\omega\downarrow$

  By induction hypothesis we have that each component on right hand side of $f\tau$" $\in_{E'} H(seed(T))$ and hence the head of $f\tau$" is also in $H(seed(T))$.

- Again on induction on size of R, we will show that

$$k_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, t\downarrow) \in H(seed(T))$$

1. If $R = c$ is a public name, we have that the statement $f = (k(c,c) \Leftarrow)$ is in the set of seed statements by definition. So, $k(R, t\downarrow) = (k(c,c) \in H(seed(T))$ by definition and hence $k_{L_1\varphi\downarrow,\ldots,L_n\varphi\downarrow}(R, R\varphi\downarrow) \in H(seed(T))$ by the EXTENDK rule.
2. If $R = f(R_1, \ldots, R_k)$, the proof is same as the case 2 stated above for the base case.
3. If $R = w_j$, let m be the index of the output action $a_m = \mathbf{out}(c_m, t_m)$ which maps t_m to w_j.
   By definition of seed knowledge base we have that,

$$(k_{l_1\tau\downarrow,\ldots,l_m\tau_V\downarrow}(w_j, t_m\tau_V\downarrow) \Leftarrow \left\{k_{l_1\tau_V\downarrow,\ldots,l_{k-1}\tau_V\downarrow}(X_k, x_k\tau_V\downarrow)\right\}_{k\in R(m)}) \in seed(T)$$

   for all $\tau_V \in V$ where V is one of the possible sets arising from the computation of $\mathbf{variants}_E(l_1, \ldots, l_m, t_m)$ Let $\tau \in V$ and τ' be such that $(l_1, \ldots, l_m, t_m)\omega\downarrow =_{E'} ((l_1, \ldots, l_m, t_m)\tau)\downarrow \tau'$ Let

$$h = (k_{l_1\tau\downarrow,\ldots,l_m\tau\downarrow}(w_j, t_m\tau\downarrow) \Leftarrow \left\{k_{l_1\tau\downarrow,\ldots,l_{k-1}\tau\downarrow}(X_k, x_k\tau\downarrow)\right\}_{k\in R(m)})$$

   Let R_k be recipes of $(x_k\tau)\downarrow \tau' =_{E'} x_k\omega$ in the smallest possible prefix of φ, that is the first time it appears in the frame. Let $\tau'' = \tau' \cup \{X_k \mapsto R_k\}_{k\in R(m)}$. We have that the antecedents of $h\tau''$ are in $H(seed(T))$ by induction hypothesis and hence so is head of $h\tau''$.

Hence we have now proved the soundness and completeness of the seed statements.

8 Conclusion

References

[AF04] Martin Abadi and Cdric Fournet. Private authentication. In *Theoretical Computer Science*, pages 427–476, 2004.

[AG99] Martin Abadi and Andrew Gordon. A calculus for cryptographic protocols: The spi calculus. In *Information and Computing (148)*, pages 1–70, 1999.

[B.B] S.Delaune C.Fournet S.Kremer D.Pointcheval B.Blanchet, H.Comon-Lundh. Cryptographic protocols: Formal and computational proofs. *MPRI Lecture Notes*, 2-30.

[BRR02] M. Boreale, R.Nicola, and R.Pugliese. Proof techniques for cryptographic processes. In *SIAM journal on Computing 31(3)*, pages 947–986, 2002.

[Che12] Vincent Cheval. Automated verification of cryptographic protocols: Privacy-type properties. *Ph.D Thesis at ENS Cachan*, 2012.

[DK01] H. Delfs and H. Kneb. *Introduction to cryptography: principles and applications.* New York, NY, USA. Springer-Verlag, 2001.

[DY83] D. Dolev and A. C. Yao. On the security of public-key protocols. In *IEEE Transactions on Information Theory*, page 198208, 1983.

[Gal03] Jean H. Galier. *Logic for Computer Science.* Undergraduate texts in mathematics. Springer, 2003.

[HLS05] H.Comon-Lundh and S.Delaune. The finite variant property: How to get rid of some algebraic properties. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA'05)*, pages 294–307, 2005.

[MB09] M.Ryan and B.Smith. Applied pi calculus. In *In CSF09: Proceedings of the 22nd IEEE Computer Security Foundations Symposium*, pages 355–370, 2009.

[oST99] National Institute of Standards and Technology. Federal information processing standards. *FIPS PUB 46-3*, 1999.

[oST01] National Institute of Standards and Technology. Federal information processing standards. *FIPS*, 2001.

[RLRA83] A. Shamir R. L. Rivest and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Commun. ACM*, pages 26–98, 1983.

[RSS12] R.Chadha, S.Ciobaca, and S.Kremer. Automated verification of equivalence properties of cryptographic protocols. In *21st European Symposium on Programming (ESOP)*, 2012.

[S.C11] S.Ciobaca. Verification and composition of security protocols with applications to electronic voting. *Ph.D Thesis at ENS Cachan*, 2011.

[VS09] V.Cortier and S.Delaune. A method for proving observational equivalence. In *22nd IEEE Computer Security Foundations Symposium*, pages 266–276, 2009.

[VSP06] V.Cortier, S.Delaune, and P.Lafourcade. A survey of algebraic properties used in cryptographic protocols. In *Journal of Computer Security 14(1)*, pages 1–43, 2006.

[vT03] H. C. A. van Tilborg. *Encyclopedia of Cryptography and Security*. Encyclopedia. Springer, 2003.