Identity Escrow Protocol and Anonymity Analysis in the Applied pi-calculus

AYBEK MUKHAMEDOV AND MARK D. RYAN
University of Birmingham

Anonymity with identity escrow attempts to allow users of an on-line service to remain anonymous, while providing the possibility that the service owner can break the anonymity in exceptional circumstances, such as to assist in a criminal investigation. In the paper, we propose an identity escrow protocol that distributes user identity among several escrow agents. The main feature of our scheme is it is based on standard encryption algorithms and it provides user anonymity even if all but one escrow holders are dishonest acting in a coalition. We also present analysis of the anonymity property of our protocol in the applied pi calculus. We review a related scheme by Marshall and Molina-Jiminez that aimed to achieve goals similar to ours, and show that their scheme suffers from serious weaknesses.

Categories and Subject Descriptors: C.2.2 [**Network Protocols**]: Protocol verification; K.4.4 [**Electronic Commerce (J.1)**]: Security

General Terms: Security, Verification

Additional Key Words and Phrases: identity escrow protocol, applied π-calculus

1. INTRODUCTION

With the increasing sophistication and adoption of communication systems in businesses and personal use, privacy and anonymity has become a concern among users [BusinessWeek 2000; Cranor et al. 1999; Harris 1999]. Service usages (such as usage of mobile phones, Internet, financial payments) are routinely logged, and those logs will allow organisations to build sophisticated profiles of customers and their preferences and associates. Users fear that this information could be abused. But while users may wish for complete privacy and anonymity, the failure of digital cash to achieve widespread adoption shows that society as a whole also requires security and accountability. Digital cash failed because it would allow criminal behaviour to go undetected. An appropriate balance between unrestricted anonymity and totalitarian security needs to be found, and this is likely to be a major theme in security research for some years.

Authors' address: School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK. Email: A.Mukhamedov@cs.bham.ac.uk, M.D.Ryan@cs.bham.ac.uk. Web page: www.cs.bham.ac.uk/~axm, www.markryan.eu
The authors gratefully acknowledge financial support from EPSRC, via the research grant EP/E040829/1 (Verifying anonymity and privacy properties of security protocols).

Identity escrow attempts to provide such a balance for some applications. It allows users to access services anonymously while guaranteeing that service providers can break the anonymity in special circumstances; for example, to assist in a criminal investigation. If Alice wishes to use the service from provider S, she first puts her identity in escrow with an *escrow agent* T, from whom she obtains a token. She presents the token to S as evidence that she has placed her identity in escrow. S allows her to use the service anonymously. In the event of service misuse, S can apply to T to obtain the identity of the user corresponding to the token.

The term 'identity escrow' was first introduced by Kilian and Petrank in [Kilian and Petrank 1998], which was motivated by the ideas from *key escrow encryption systems* (e.g. [Leighton 1994; Micali 1993]). The idea can also be traced in *group signature schemes* and *anonymous credential systems*, which are mechanisms which can be used to offer identity escrow [Camenisch and Lysyanskaya 2001; 2004; Camenisch and Shoup 2003; Kiayias and Yung 2004]. Clearly, the systems break down if a single escrow agent (known as *group manager* or *issuer*) holds the escrowed identity and he is dishonest – he can reveal Alice's identity even if the agreed conditions for doing so have not been met. To address this problem the escrow agent can be implemented as a *set* of agents called *token providers* [Kilian and Petrank 1998]. Neither S nor any token provider are supposed to know the identity behind an escrowed certificate, but if it is proved necessary, all token providers can cooperate in order to reveal it.

We propose a new identity escrow scheme that offers several advantages over the escrow schemes mentioned above. In our scheme, the escrowed identity is distributed among several token providers chosen by the user and the user's list (except for the last token provider in the list) is not revealed when he presents his token to the service provider. Moreover, our scheme uses standard cryptographic primitives, which are better known than zero-knowledge proofs and more widely implemented in APIs. Having said that, there are some features of our protocol which may make it unattractive in some cases. Although the cryptographic primitives are standard, they are not computationally cheap. The distribution of trust in a large number of token providers is great for the user, but service providers may not like it because obtaining a user's identity in the event of misuse requires all the token providers to be available. We expect the protocol to find applications in niche areas where the anonymity requirement is paramount, and misuse is rare.

We model and analyze our protocol in the applied pi-calculus, and show that it satisfies the anonymity property. In an earlier work Marshall and Molina-Jiminez [Marshall and Molina-Jiminez 2003] proposed a protocol that aimed to achieve goals similar to ours – distribute escrowed user identity among several trusted parties using standard cryptographic primitives. However, their scheme puts strong assumptions on escrow agents and suffers from serious weaknesses, which we mention in the paper.

The paper is organised as follows. In the following two sections, we present our protocol together with its formal analysis in the applied pi-calculus. Next, we briefly detail Marshall and Molina-Jiminez's protocol and the problems we have identified with it, and conclude in section 5. Appendix A summarises the applied pi-calculus, for the benefit of readers not familiar with it. Appendix B contains the proofs of lemmas we rely upon for our analysis.

Our protocol together with its preliminary analysis appeared in [Mukhamedov and Ryan 2007] and our analysis of Marshall and Molina-Jiminez's protocol appeared in [Mukhamedov and Ryan 2005].

2. THE PROTOCOL

The protocol is based on the idea that certain agents that we call *token providers* assist the user by helping him construct an onion. The user's identity is in the innermost layer, while the service key with which the user obtains the service is in the outermost layer. In the event of an upheld complaint, one can link the service key and the user's identity by contacting all the token providers that were used. If they are honest, they will only assist in the link if there really is an upheld complaint. Thus, the anonymity of the user is guaranteed even if all but one escrow holders are dishonest.

2.0.1 *Notation.* The following labeling conventions are used throughout the paper:

—S denotes an anonymous service provider.

—$T = \{T_1, T_2, \ldots\}$ is a set of *identity token providers.*

—Φ_i is an identity token issued by T_{a_i}. We write Φ_A for the full identity token obtained by A by using the protocol.

—K_A is A's long-term public key. $K_{[A]}$ is the public part of A's ephemeral key that it uses to access anonymous services provided by S. $K_{[A]}$ is freshly generated by A for each service usage and no other agent knows the correspondence between $K_{[A]}$ and A.

—$\{m\}_K$ is the message m deterministically encrypted with the public key K and $[m]_{K^-}$ is A's universally verifiable signature on m.

Our protocol consists of two parts. First, there is a *sign-up* protocol, which is the main protocol that is executed by A to receive a token from the members of T. The token permits A to use the service from S. Next, there is a *complaint resolution* protocol, which is executed by S upon a misuse of its service, in order to reveal the identity of the offending anonymous user.

2.0.2 *Sign-up.* Alice has a long-term certified public key K_A. She creates a temporary service public key $K_{[A]}$ which she will use to identify herself to S. Then she proceeds to build up an onion Φ_n (actions 1, 2 below). At the end of this process, the onion consists of the service key $K_{[A]}$ in its centre, wrapped with encryptions and signatures by the token providers. This is then paired with Alice's identity A, and by engaging in the protocol with token providers, it is wrapped again by encryptions and signatures of the token providers. The formal definition follows; an illustration in the case $n = 4$ is given in Figure 1.

The onion build-up is detailed as follows. Alice choses a sequence of token providers $T_{a_1}, T_{a_2}, \ldots, T_{a_n}$ from T (possibly with duplications) and creates the following term:

$$1) \quad \Phi_1 = \{ \ \mathtt{InitITKReq}, K_{[A]}, \ \mathtt{K}_{\mathtt{a}_1} \ \}_{K_{T_{a_1}}}$$

It is an encryption of a tag InitITKReq, the public part of A's service key $K_{[A]}$ and the symmetric key $\mathtt{K}_{a_1}$ by T_{a_1}'s public key, where $\mathtt{K}_{a_1}$ is freshly generated by A. The goal of the protocol is to have the key $K_{[A]}$ associated with A's identity token in a way that does not reveal this link even if all but one T_{a_i} are dishonest.

Next A creates further terms from Φ_1:

$$2) \text{ for } i = 2 \text{ to } n-1 \quad : \quad \Phi_i = \{ \ \mathtt{ITKReq}, \Phi_{i-1}, \mathtt{K}_{\mathtt{a}_i} \ \}_{K_{T_{a_i}}}$$

By the end of the sequence of encryptions (2) ($n-2$ times), A will have obtained the token Φ_{n-1}:

$$\{ \ \mathtt{ITKReq}, \{ \ \ldots \ \{ \ \mathtt{ITKReq}, \{\mathtt{InitITKReq}, K_{[A]}, \mathtt{K}_{a_1}\}_{K_{T_{a_1}}}, \mathtt{K}_{a_2}\}_{K_{T_{a_2}}} \ \ldots \ \}_{K_{T_{a_{n-2}}}}, \mathtt{K}_{a_{n-1}}\}_{K_{T_{a_{n-1}}}}$$

The token Φ_{n-1} serves as a disguise of the service key $K_{[A]}$. The symmetric keys $\mathtt{K}_{\mathtt{a}_i}$ generated by A in the above steps are used in order to randomise the ciphertexts and to encrypt messages from token providers in later stages.

Next, A signs the token Φ_{n-1} and sends it to T_{a_n}, and then contacts $T_{a_{n-1}}, T_{a_{n-2}}, \ldots, T_{a_1}$ anonymously as shown in Steps 3, 4, 3a, 4a. They reverse the sequence of encryptions, and at the same time build up the identity token $\widetilde{\Phi}_{n-1}$. The notation $A \longmapsto B$ means that A anonymously sends a message to B. In this case, B does not know A's identity. Similarly, $A \longrightarrow\!| \ B$ means that B receives a message anonymously, from A; A does not know B's identity.

$$3) \quad A \quad \longrightarrow \quad T_{a_n} \quad : \quad \{ \ [\ \mathtt{InitITKSig}, \ \Phi_{n-1}, \ A \ ]_{K_A^-} \ \}_{K_{T_{a_n}}}$$

$$4) \quad T_{a_n} \quad \longrightarrow \quad A \quad : \quad \widetilde{\Phi}_1$$
$$\text{where } \widetilde{\Phi}_1 = [\ \{ \ [\ \mathtt{InitITKSig}, \ \Phi_{n-1}, \ A \ ]_{K_A^-} \ \}_{K_{T_{a_n}}}, \Phi_{n-1} \ ]_{K^-_{T_{a_n}}}$$

For $i = 1$ to $n-2$:

$$* \begin{cases} 3a) \quad A \quad \longmapsto \quad T_{a_{n-i}} \quad : \quad \{ \ \mathtt{ITKSig}, \widetilde{\Phi}_i, \mathtt{K}_{\mathtt{a}_{n-i}} \ \}_{K_{T_{a_{n-i}}}} \\ \quad \text{where for } i > 1 \quad \widetilde{\Phi}_i = [\ \{ \ \widetilde{\Phi}_{i-1}, \mathtt{K}_{\mathtt{a}_{n-i+1}}\}_{K_{T_{a_{n-i+1}}}}, \ \Phi_{n-i} \ ]_{K^-_{T_{a_{n+1-i}}}} \\ \\ 4a) \quad T_{a_{n-i}} \quad \longrightarrow\!| \quad A \quad : \quad \{ \ [\ \{ \ \widetilde{\Phi}_i, \mathtt{K}_{\mathtt{a}_{n-i}} \ \}_{K_{T_{a_{n-i}}}}, \Phi_{n-i-1} \ ]_{K^-_{T_{a_{n-i}}}} \}_{\mathtt{K}_{\mathtt{a}_{n-i}}} \end{cases}$$

After step 3a, before sending out a response, the token provider $T_{a_{n-i}}$ checks that the session key $\mathtt{K}_{\mathtt{a}_{n-i}}$ supplied in the request matches the one embedded in Φ_{n-i} (cf. step 2 above). The same rule applies to T_{a_1} at step 5. In addition, both token providers also check that $\widetilde{\Phi}_i$ contained in the request was signed by a token provider.

$$5) \quad A \quad \longmapsto \quad T_{a_1} \quad : \quad \{ \ \mathtt{ITKSig}, \widetilde{\Phi}_{n-1}, \mathtt{K}_{\mathtt{a}_1}\}_{K_{T_{a_1}}}$$

$$6) \quad T_{a_1} \quad \longrightarrow\!| \quad A \quad : \quad \{ \ \widetilde{\Phi}_A \ \}_{\mathtt{K}_{\mathtt{a}_1}},$$
$$\text{where } \widetilde{\Phi}_A = [\ \{ \ \widetilde{\Phi}_{n-1}, \mathtt{K}_{\mathtt{a}_1} \ \}_{K_{T_{a_1}}}, K_{[A]} \ ]_{K^-_{T_{a_1}}}$$

Upon reaching step 6, A has the following identity token:

$$\widetilde{\Phi}_A = [\{ \ldots\ [\{\widetilde{\Phi}_1,\ \mathtt{K}_{\mathtt{a}_{n-1}}\}_{K_{T_{a_{n-1}}}}, \Phi_{n-2}\ ]_{K^-_{T_{a_{n-1}}}} \ldots \mathtt{K}_{\mathtt{a}_1}\}_{K_{T_{a_1}}}, K_{[A]}\ ]_{K^-_{T_{a_1}}}$$

The token $\widetilde{\Phi}_A$ associates the service key $K_{[A]}$ with A. He presents the token to S when requesting its service.

$$7)\quad A \quad \longmapsto \quad S \quad : \quad \{\ \widetilde{\Phi}_A,\ K_{[A]}\ \}_{K_S}$$

S checks that the token is signed by some token provider, and that the value signed is a pair whose second element is the key $K_{[A]}$. In case of service misuse the token may be delayered to reveal the identity of a user bound to it via the complaint resolution protocol.

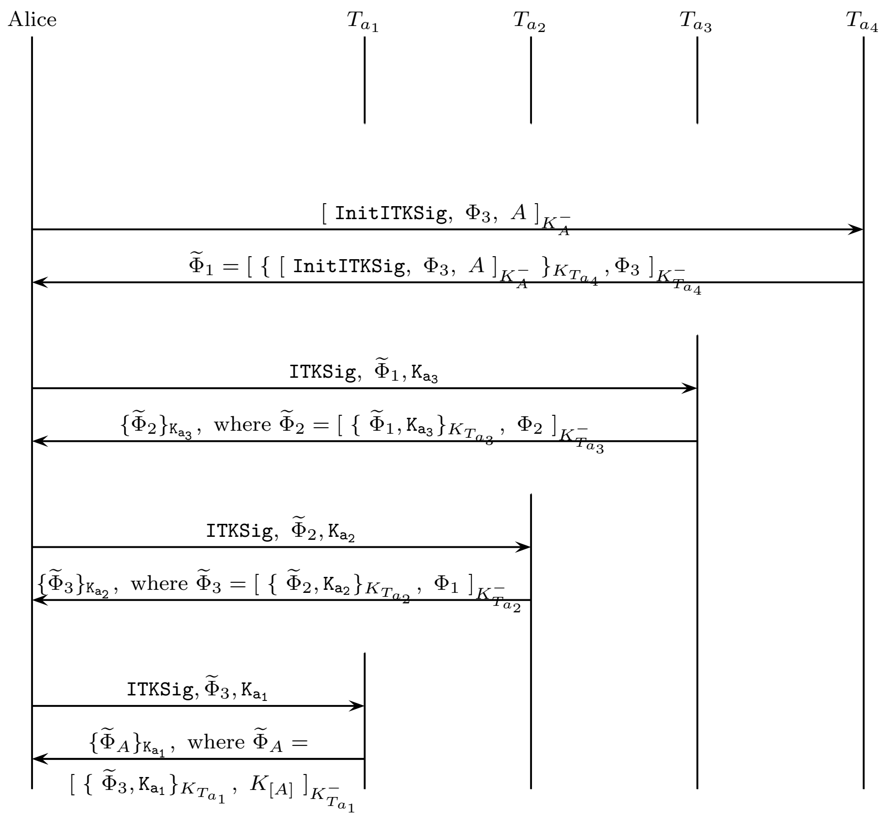
Alice
T_{a_1}
T_{a_2}
T_{a_3}
T_{a_4}
$[\ \mathtt{InitITKSig},\ \Phi_3,\ A\ ]_{K^-_A}$
$\widetilde{\Phi}_1 = [\ \{\ [\ \mathtt{InitITKSig},\ \Phi_3,\ A\ ]_{K^-_A}\ \}_{K_{T_{a_4}}}, \Phi_3\ ]_{K^-_{T_{a_4}}}$
$\mathtt{ITKSig},\ \widetilde{\Phi}_1, \mathtt{K}_{\mathtt{a}_3}$
$\{\widetilde{\Phi}_2\}_{\mathtt{K}_{\mathtt{a}_3}}$, where $\widetilde{\Phi}_2 = [\ \{\ \widetilde{\Phi}_1, \mathtt{K}_{\mathtt{a}_3}\}_{K_{T_{a_3}}},\ \Phi_2\ ]_{K^-_{T_{a_3}}}$
$\mathtt{ITKSig},\ \widetilde{\Phi}_2, \mathtt{K}_{\mathtt{a}_2}$
$\{\widetilde{\Phi}_3\}_{\mathtt{K}_{\mathtt{a}_2}}$, where $\widetilde{\Phi}_3 = [\ \{\ \widetilde{\Phi}_2, \mathtt{K}_{\mathtt{a}_2}\}_{K_{T_{a_2}}},\ \Phi_1\ ]_{K^-_{T_{a_2}}}$
$\mathtt{ITKSig}, \widetilde{\Phi}_3, \mathtt{K}_{\mathtt{a}_1}$
$\{\widetilde{\Phi}_A\}_{\mathtt{K}_{\mathtt{a}_1}}$, where $\widetilde{\Phi}_A =$
$[\ \{\ \widetilde{\Phi}_3, \mathtt{K}_{\mathtt{a}_1}\}_{K_{T_{a_1}}},\ K_{[A]}\ ]_{K^-_{T_{a_1}}}$

Fig. 1. Illustration of the sign-up protocol in the case $n = 4$. Messages from A are encrypted with the public key of the receiver (this encryption is not shown)

2.0.3 *Complaint Resolution.* We assume that some misuse evidence $\widetilde{\Psi}_{K_{[A]}}$ is uniquely associated with A's service key $K_{[A]}$ and the service provider S. It must be verifiable by each token provider (or endorsed by a third party accepted by all token providers) and not forgeable by S.

The protocol is given as follows. As before, an illustration is given for the case $n = 4$ in Figure 2.

$$1)\quad S \longrightarrow T_{a_1} \;:\; \{\, \mathtt{Reveal}, \widetilde{\Phi}_A, \widetilde{\Psi}_{K_{[A]}}, S \,\}_{K_{T_{a_1}}}$$

$$2)\quad T_{a_1} \longrightarrow S \;:\; \{\, [\widetilde{\Phi}_{n-1}, \mathtt{K}_{a_1}, \widetilde{\Psi}_{K_{[A]}}]_{K^-_{T_{a_1}}} \}_{K_S}$$

For $i = 1$ to $n - 2$:

$$* \begin{cases} 3a)\quad S \longrightarrow T_{a_{i+1}} \;:\; \{\, \mathtt{Reveal},\ ((\widetilde{\Phi}_{n-i}, \mathtt{K}_{a_i}), \ldots, (\widetilde{\Phi}_{n-1}, \mathtt{K}_{a_1}), \widetilde{\Phi}_A), \\ \qquad\qquad\qquad\qquad\qquad \widetilde{\Psi}_{K_{[A]}}, S \,\}_{K_{T_{a_{i+1}}}} \\ 4a)\quad T_{a_{i+1}} \longrightarrow S \;:\; \{\, [\widetilde{\Phi}_{n-i-1}, \mathtt{K}_{a_{i+1}}, \widetilde{\Psi}_{K_{[A]}}]_{K^-_{T_{a_{i+1}}}} \}_{K_S} \end{cases}$$

$$5)\quad S \longrightarrow T_{a_n} \;:\; \{\, \mathtt{Reveal},\ ((\widetilde{\Phi}_1, \mathtt{K}_{a_{n-1}}), \ldots, (\widetilde{\Phi}_{n-1}, \mathtt{K}_{\mathtt{a}_1}), \widetilde{\Phi}_A), \widetilde{\Psi}_{K_{[A]}}, S \,\}_{K_{T_{a_n}}}$$

$$6)\quad T_{a_n} \longrightarrow S \;:\; \{\, [[\ \mathtt{ITKSig},\ \Phi_{n-1}, A\ ]_{K^-_A}, \widetilde{\Psi}_{K_{[A]}}]_{K^-_{T_{a_n}}} \}_{K_S}$$

We assume that S has access to the chain of token providers $T_{a_1}, \ldots T_{a_n}$ through the application software, so S knows what signature verification keys should be used.

In message 3a, the tuple of $\widetilde{\Phi}_i$s serves to prevent complaint resolution messages in one session being used in another. Before sending a response each T_{a_i} checks that the sequence he receives is correct, i.e. $\{\widetilde{\Phi}_{n-i}, \mathtt{K}_{\mathtt{a_i}}\}_{K_{T_{a_{i-1}}}}$ equals to the second element in the signed tuple $\widetilde{\Phi}_{n-i+1}$, and $\{\widetilde{\Phi}_{n-i+1}, \mathtt{K}_{\mathtt{a_{i-1}}}\}_{K_{T_{a_{i-2}}}}$ equals to the second element in the signed tuple $\widetilde{\Phi}_{n-i+2}$, etc. , until $\widetilde{\Phi}_A$ is reached. This check ensures that T_{a_i} will not decrypt a token that is not related to $\widetilde{\Phi}_A$.

At the nth iteration S reveals the identity of the user when it receives $[\ \mathtt{ITKSig},\ \Phi_{n-1}, A\ ]_{K^-_A}$ from T_{a_n}. Importantly, in the sequence of unfoldings of $\widetilde{\Phi}_{a_i}$s, S also keeps track of Φ_{a_i}s inside them, using the session keys $\mathtt{K}_{a_i}$, in order to make sure that Φ_{n-1} is formed from the session key she was given in the service request step. That is to avoid rogue token providers disrupting or misleading the identity recovery process.

3. ANALYSIS OF ANONYMITY PROPERTY

We prove that the protocol satisfies anonymity: the identity token produced by the user cannot be linked to its identity, even if all but one of the token providers are dishonest. We start with defining the model of the protocol, then present some general results about the static equivalence on frames involving nonces and encryptions. We conclude the section with the proof of anonymity.

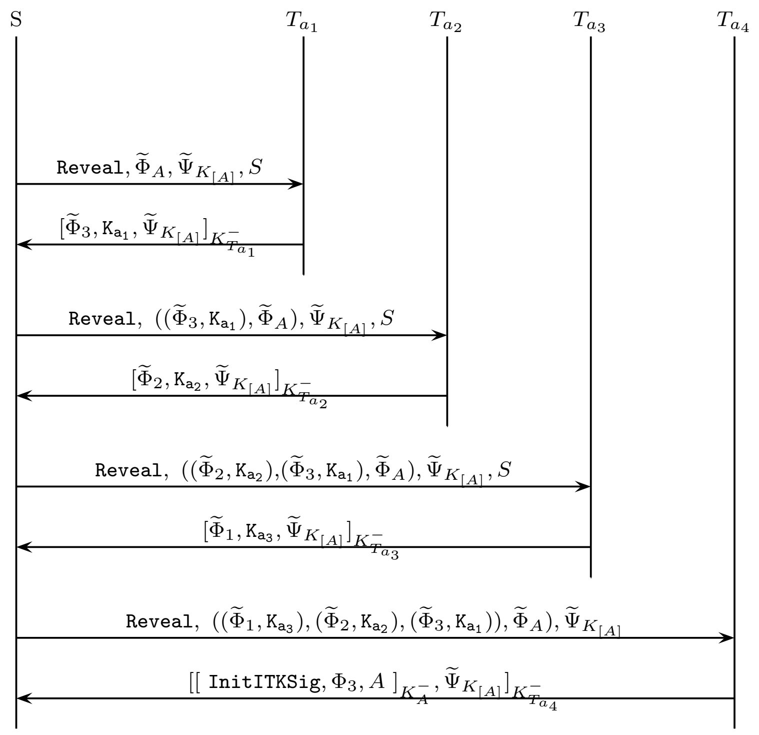
S
T_{a_1}
T_{a_2}
T_{a_3}
T_{a_4}
$\mathtt{Reveal}, \widetilde{\Phi}_A, \widetilde{\Psi}_{K_{[A]}}, S$
$[\widetilde{\Phi}_3, \mathtt{K_{a_1}}, \widetilde{\Psi}_{K_{[A]}}]_{K^-_{T_{a_1}}}$
$\mathtt{Reveal},\ ((\widetilde{\Phi}_3, \mathtt{K_{a_1}}), \widetilde{\Phi}_A), \widetilde{\Psi}_{K_{[A]}}, S$
$[\widetilde{\Phi}_2, \mathtt{K_{a_2}}, \widetilde{\Psi}_{K_{[A]}}]_{K^-_{T_{a_2}}}$
$\mathtt{Reveal},\ ((\widetilde{\Phi}_2, \mathtt{K_{a_2}}),(\widetilde{\Phi}_3, \mathtt{K_{a_1}}), \widetilde{\Phi}_A), \widetilde{\Psi}_{K_{[A]}}, S$
$[\widetilde{\Phi}_1, \mathtt{K_{a_3}}, \widetilde{\Psi}_{K_{[A]}}]_{K^-_{T_{a_3}}}$
$\mathtt{Reveal},\ ((\widetilde{\Phi}_1, \mathtt{K_{a_3}}), (\widetilde{\Phi}_2, \mathtt{K_{a_2}}), (\widetilde{\Phi}_3, \mathtt{K_{a_1}})), \widetilde{\Phi}_A), \widetilde{\Psi}_{K_{[A]}}$
$[[\ \mathtt{InitITKSig}, \Phi_3, A\ ]_{K^-_A}, \widetilde{\Psi}_{K_{[A]}}]_{K^-_{T_{a_4}}}$

Fig. 2. Complaint resolution protocol for the case $n = 4$. All messages are encrypted with the public key of the recepient (to avoid clutter this encryption is not shown).

3.1 Model of the protocol

The protocol is modelled in the applied π-calculus. We do not put restrictions on the number of sessions, or agents, and assume an active adversary (aka Dolev-Yao) that can inject as well as intercept messages from the network. Public channels represent the network and they are the means of interacting with the environment, whereas private channels are used for private communications among processes. Channels by themselves do not reveal sender or recipient of messages, and thus are anonymous. We present the model in the language of the ProVerif tool extended with a `for`-loop construct, which is defined as a macro expansion: `for condition(j)` followed by `body(j)` stands for `body(1), body(2), ..., body(n)` , where $\{1, \ldots, n\}$ is the set of integers that satisfy the `condition(j)`. The scope of the loop is denoted by indetation (as in Python programming language). We also allow conditional expressions inside the for-loop, which are unpacked when the macro expansion takes place: `ifex[e1=e2,e3,e4]` is replaced by `e3`, if `e1` and `e2` are syntactically equal after substituting for the control variable (e.g., `j`) of the for-loop; otherwise it is replaced by `e4`.

```
fun pk/1.         (* gets public key from a private key *)
fun enc/2.        (* public key encryption *)
fun senc/2.       (* symmetric key encryption *)
fun dec/2.        (* and decryption       *)
fun sdec/2.
fun sign/2.       (* universally verifiable signature *)
fun getSigKey/1.  (* retrieves public key of a signer  *)
fun getSigMess/1. (* retrieves a message from a signature *)
fun pair/2.       (* pairs two terms *)
fun fst/1.        (* first term of a pair *)
fun snd/1.        (* second term of a pair *)
fun thd/1.        (* third term of a tuple *)

equation dec(enc(m,pk(sk)),sk) = m.
equation sdec(senc(m,k),k) = m.
equation getSigKey(sign(m,sk)) = pk(sk).
equation getSigMess(sign(m,sk)) = m.
equation fst(pair(x,y))=x.
equation snd(pair(x,y))=y.
equation thd(x) = snd(snd(x)).
```

Fig. 3. Signature and equational theory. The notation "`fun` f/i" means that f is a function of arity i.

3.2 Signature and equations

The signature of our model includes function symbols for public key cryptographic operations and universally verifiable signing, as well as other auxiliary constants and functions used in the protocol. The purpose of the functions should be clear from the comments in brackets. The equational theory is generated by the equations shown in Figure 3.

We sometimes write (M, N) instead of $\mathrm{pair}(M, N)$, and (M, N, K) instead of $\mathrm{pair}(M, \mathrm{pair}(N, K))$. We also use some shorthands from ProVerif syntax. We write "let $x = M$ in P" to mean $P\{^{M}/_{x}\}$, and we allow pattern matching. Thus, for example, "let $(x, = N) = M$ in P" means "if $\mathrm{snd}(M) = N$ then $P\{^{\mathrm{fst}(M)}/_{x}\}$.

3.3 The protocol process

The protocol is encoded in the processes as shown in Figure 4 and 5, where `processT` and `processU` denote token provider and user, respectively (defined below).

The fresh name `skTh` represents the private decryption key of the honest token provider. Signing keys are of course different from the private decryption keys. We allow the intruder to access honest token providers signing key; intuitively, the anonymity property relies only on the secrecy of their decryption keys. (Secrecy of the signing keys is important for other properties of the protocol, which we don't analyse.) The dishonest token providers are represented by the attacker that has an encryption key `pk(skTd)` and a signing key `signTd`.

The expression of the form `new n; P` corresponds to the restriction $\nu n.P$ of the applied-π calculus. A construct of the form `(=N,y)=M` pattern matches the left element of a tuple `M` against `N`, but assigns the right element of `M` to `y`.

```
free c.           (* public channel *)
free ITKReq, ITKSig, InitITKReq, InitITKSig.
free skTd,signTd. (* dishonest TP's decryption and signing keys *)
free signTh   (* honest TP's signing key *)

process
  new skTh; (* honest TP's decryption key *)
  out(c,pk(skTh)); new signA;
  let  (pkTh, pkTd) = (pk(skTh),pk(skTd)) in !processT | (out(c,pk(signA));
  let (signU,n,pU) = (signA,nA,pA) in processU)


let processU=
  (* pU is Th's position in U's request chain *)
  for 0<j<n+1 & j<>pU:
     new sesK_j;
     let (upkT_j,signT_j) = (pkTd,signTd) in
  new sesK_pU;
  let (upkT_pU,signT_pU) = (pkTh,signTh) in
  new servK; out(c,pk(servK));

  (* Step 1 *)
  let phi_1=enc((InitITKReq,pk(servK),sesK_1),upkT_1) in
  (* Step 2 *)
  for 1<j<n:
    let phi_j=enc((ITKReq,phi_(j-1),sesK_j),upkT_j) in
  (* Step 3,4 *)
  let commit = sign((InitITKSig,phi_(n-1),pk(signU)),signU) in
  out(c,enc(commit,upkT_n));
  in(c,m3);
  let tphi_1=m3 in
  if getSigKey(tphi_1)=pk(signT_n) then
  let (x1, oldTphi) = getSigMess(tphi_1) in
  if oldTphi = enc((sign((ITKSig,phi_(n-1),pk(signU)),signU)),upkT_n) then
  (* Step 3a,4a *)
  for 0<j<n-1:
     out(c,enc((ITKSig,tphi_j,sesK_(n-j)),upkT_(n-j)));
     in(c,m4);
     let tphi_(j+1)=dec(m4,sesK_(n-j)) in
     if getSigKey(tphi_(j+1))=pk(signT_(n-j)) then
  (* Step 5,6 *)
  out(c,enc((ITKSig,tphi_(n-1),sesK_1),upkT_1));
  in(c,m);
  let token=dec(m,sesK_1) in
  if getSigKey(token)=pk(signT_1) then
  let (x2,key) = getSigMess(token) in
  if key = servK then out(c,token)
```

Fig. 4. The Main and the user processes. We have extended ProVerif syntax with a `for`-loop and conditional expression (used in steps 2 and 3a,4a), as defined in the text.

```
let processT=
  in(c,m);
  let req=dec(m,skTh) in

  let (=ITKSig,d4,k)=req in
  (
   let (x1,oldPhi) = getSigMess(d4) in
   (
   let (=InitITKReq,key,=k) = dec(oldPhi,skTh) in
   out(c,senc(sign((enc((d4,k),pkTh),key),signTh),k))
   else
   let (=ITKReq,oldPhi1,=k) = dec(oldPhi,skTh) in
   out(c,senc(sign((enc((d4,k),pkTh),oldPhi1),signTh),k))
   )
  )

  else let (=InitITKSig,d3,upk)=getSigMess(req) in
  (
   if upk=getSigKey(req) then
   out(c,sign((enc(req,pkTh),d3),signTh))
  ).
```

Fig. 5. The token provider process

3.4 Auxiliary results

In this section we present several results about the static equivalence of frames. The last three lemmas of the section are particularly interesting since two of those establish criteria when a ciphertext (a public key or symmetric key encrypted message) reveals no more information to the attacker than a freshly generated value, and the last one says when a term occuring inside a ciphertext remains hidden from the attacker. We will use the lemmas to simplify analysis of the anonymity property in the next section.

Assumption 1. *Let E be an equational theory. We assume that the relation equality modulo E on terms is closed under substitutions of arbitrary terms for names and variables, and application of contexts.*

Definition 1. *We write $\{^M/_N\}$ for the substitution that replaces all occurrences of the term N by the term M.*

Note that $T_1 =_E T_2$ does not imply that $T_1\{^M/_N\} =_E T_2\{^M/_N\}$.

The first simple lemma shows that exporting nonces does not affect static equivalence on frames. It is a consequence of Lemma 1 of [Abadi and Fournet 2001] using the context $\nu k.(\{^k/_x\} \mid \_)$, although since that lemma is not proved there, we provide our own proof.

Lemma 1. *Let E be an equational theory, φ, φ' be frames, $\tilde{n}, \tilde{n}'$ be sets of names and k a name s.t. $k \notin fn(\varphi, \varphi') \cap (\tilde{n} \cup \tilde{n}')$. If $\nu\tilde{n}.\varphi \approx_s \nu\tilde{n}'.\varphi'$ then $\nu\tilde{n}, k.(\{^k/_x\} \mid \varphi) \approx_s \nu\tilde{n}', k.(\{^k/_x\} \mid \varphi')$, where $x \notin dom(\varphi)$.*

The following lemma establishes sufficient conditions under which parts of frames can be simplified (substituted by fresh names). All further lemmas make use of this result. We say that a frame is in normal form if all the terms occurring in it are in normal form. When we write $M\sigma\tau$, it means $(M\sigma)\tau$.

Lemma 2. *Consider a convergent equational theory E, a closed term L in normal form, names $\tilde{n}, s$ and a frame φ in normal form such that $s \notin fn(\varphi)$. Suppose that:*

—*L does not occur in φ, and $\nu\tilde{n}.\varphi \not\vdash L$.*
—*for any $\tilde{m}, \sigma, M, N$ such that $\nu\tilde{n}.(\{^L/_x\}|\varphi) \equiv \nu\tilde{m}.\sigma$, $(fn(M) \cup fn(N)) \cap \tilde{m} = \emptyset$ and $M\sigma =_E N\sigma$ we have $M\sigma\{^z/_L\} =_E N\sigma\{^z/_L\}$.*

Then: $\nu\tilde{n}.(\{^L/_x\}|\varphi) \approx_s \nu\tilde{n}, s.(\{^s/_x\}|\varphi)$.

All subsequent lemmas are restricted to the equational theory with standard public and session key encryption, decryption and digital signing operations (E_{pk}) defined in Figure 3. We use the notation $\{M\}_k$ and $\texttt{enc}(M, k)$ interchangeably.

Lemma 3. *Consider the equational theory E_{pk}, and let M, N, L and J be terms in normal form s.t. M, N do not contain $\texttt{dec}(x, J)$ and $M\{^{\{L\}_J}/_x\} =_E N\{^{\{L\}_J}/_x\}$. Then:*

$$(M\{^{\{L\}_J}/_x\})\{^z/_{\{L\}_J}\} =_E (N\{^{\{L\}_J}/_x\})\{^z/_{\{L\}_J}\}$$

Lemma 4. *Consider the equational theory E_{pk}, a name k, and a frame $\nu\tilde{n}.\sigma$ in normal form that does not contain $\texttt{dec}(x, k)$, such that $\nu\tilde{n}.\sigma \not\vdash k$. If M is a term such that $\tilde{n} \cap fn(M) = \emptyset$, then $\texttt{dec}(x, k)$ does not occur in $M\sigma\downarrow$.*

Lemma 5. *Consider the equational theory E_{pk}, a frame $\nu\tilde{n}.\sigma$ in normal form, and a name $k \in \tilde{n}$, s.t. k occurs in σ only as a key argument to the encryption function (that is, only in the form $\{\_\}_k$). If M is a term such that $\tilde{n} \cap fn(M) = \emptyset$, then $\texttt{dec}(x, k)$ does not occur in $M\sigma\downarrow$.*

Lemma 6. *Consider the equational theory E_{pk}, a closed term L in normal form, names $\tilde{n}, s$, and a frame $\nu\tilde{n}.\sigma$ in normal form. Suppose $\nu\tilde{n}.\sigma \not\vdash s$ and $\{s, L\}_{pk(k)}$ does not occur in σ. Then $\nu\tilde{n}.\sigma \not\vdash \{s, L\}_{pk(k)}$.*

Lemma 7. *Given a closed term L in normal form, names $\tilde{n}, s$ and a frame $\nu\tilde{n}.\sigma$ in normal form, suppose:*

(1) *$\nu\tilde{n}.\sigma \not\vdash k$, $\nu\tilde{n}.\sigma \not\vdash \{s, L\}_{pk(k)}$ and $m \notin fn(\sigma)$*
(2) *$\{s, L\}_{pk(k)}$ does not occur in σ.*
(3) *$\texttt{dec}(x, k)$ does not occur in σ.*

Then $\nu\tilde{n}, s.(\{^{\{s,L\}_{pk(k)}}/_x\}|\sigma) \approx_s \nu\tilde{n}, m.(\{^m/_x\}|\sigma)$.

Lemma 8. *Given a closed term L in normal form, names $k, s \in \tilde{n}$ and a frame $\nu\tilde{n}.\sigma$ also in normal form, suppose:*

(1) *k occurs in σ only as an encryption key argument, i.e. in the form $\{\_\}_k$;*
(2) *L does not occur in σ and $s \notin fn(\sigma)$.*

Then: $\nu\tilde{n}.(\{^{\{L\}_k}/_x\}|\sigma) \approx_s \nu\tilde{n}, s.(\{^s/_x\}|\sigma)$.

Lemma 9. *Given a frame $\nu\tilde{n}.\sigma$ in normal form and $s, k \in \tilde{n}$, where $\nu\tilde{n}.\sigma \not\vdash k$, suppose for all occurences of s in σ:*

(1) *Either there exists a term L such that $\{L\}_{pk(k)}$ occurs in σ and the occurrence of s is a subterm of L.*

(2) *Or s occurs in σ as an encryption key argument.*

Then $\nu\tilde{n}.\sigma \not\vdash s$.

3.5 Proof of the anonymity property

Notation and set-up:

—Th is the *honest* token provider (meaning that it executes the protocol correctly), and Td is one of the dishonest ones (it does whatever the attacker says). In our threat model we also consider the service provider S to be dishonest. Our aim is to show the identity token produced by the user cannot be linked to its identity, even if all but one of the token providers and the service provider are dishonest and act in collusion.

—We don't model dishonest token providers Td and the service provider, since they form part of the attacker (in the applied pi-calculus the attacker is represented by an active context). Their decryption and signing keys are free names.

—The property is shown to hold even if Th's signing key is public. We model it as a free name that the attacker can use. Intuitively, the anonymity property hinges on the Th's private decryption key sK_{Th} being secret.

—K_{Th} stands for the public encryption key corresponding to the decryption key sK_{Th}, i.e. $pk(sK_{Th}) = K_{Th}$.

—Honest user A is an instantiation of the process `processU` in Figure 4, and correspondingly, Th is an instantiation of the process `processT` in Figure 5.

—$\tilde{n}_A$ is the set of names that include A's restricted values, i.e. signing key, service key, and session keys generated by A during the run of the protocol. $\tilde{n}_A = \{K_A^-, sK_{[A]}\} \cup \{\mathtt{KA}_1, \ldots, \mathtt{KA}_{n_A-1}\}$.

—A's *request chain* is a sequence of token providers that A uses when building a token for anonymous service usage. It is denoted by req_A with length n_A; Th's position in the chain is p_A.

We introduce the following notion of an oracle that is used to model anonymity.

Definition 2. *The oracle $\mathcal{R}_{Th}$ is a process that given an input m outputs a digital signature of Th on m, denoted by $[m]_{K_{Th}^-}$. Namely, we have $\mathcal{R}_{Th} = \mathsf{in}(s_R, m); \mathsf{out}(s_R, [m]_{K_{Th}^-})$, where s_R is a private channel shared between the honest user A and $\mathcal{R}_{Th}$.*

We now define processes A^r and A^l by which the theorem 1 below is stated.

Let A^l be `processU` (in Figure 4) with the name restrictions of lines 13, 15, 17 removed, and with K_A^- in place of `signU`, $sK_{[A]}$ in place of `servK`, $\mathtt{KA}_i$ in place of `sesK_`i, and $\mathtt{KA}_{p_A}$ in place of `sesK_pU`.

Let A^r be `processU` with the same changes as in A^l, and also the following changes: Line 9 is replaced with `new sr; let (signU,n,pU) = (signA,nA,pA)` `in processU | R_Th)`, where `R_Th` is the oracle process of Definition 2 and `sr` is the private channel (noted s_R above) shared only between `R_Th` and A^r. In the for-loop at line 21, line 22 is substituted by `new e;let phi_j = ifex[j=pU,enc((ITKReq, e,sesK_j),upkT_j),`E`] in`, where E is the rhs term of the original let expression on line 22. Line 24 is replaced with `new e;let commit = ifex[pU=n,sign( (InitITKSig,e,pk(signU)),signU),`E`] in`, where E is the rhs term of the original let expression. In the for-loop at line 32, line 35 is replaced with `new e;out(sr, (enc((ITKSig,e,sesK_(n-j)),upkT_(n-j)),phi_(n-j-1)));in(sr,N); let tphi_(j+1) = ifex[j=n-pU,N,`E`] in`, where E is the original rhs expression. Line 40 is substituted by `if pU=n then new e;out(sr,(enc((ITKSig,e,sesK_1),upkT_1),servK)); in(sr,token);`Q `else let token=dec(m,sesK_1) in` Q, where Q is the code on lines 41, 42 and 43.

Intuitively, the process A^r represents the user A constructing an empty service token with the help of the oracle $\mathcal{R}_{Th}$. More precisely, in A^r, in the message to T_{a_1} the user sends its signature on an empty token $\Phi(\varepsilon)_{n_A-1} = \{\texttt{ITKReq}, \ldots \{\texttt{ITKReq}, \varepsilon, \texttt{KA}_{p_A}\}_{K_{Th}} \ldots\}_{K_{T_{a_{n-1}}}}$, where ε is a fresh nonce generated by A. So, $\Phi(\varepsilon)_{n_A-1}$ does not contain A's service key $K_{[A]}$, but a random value inserted at the point of encryption with Th's public key. Afterwards, A^r is the same as A^l up to the point where both receive a reply from Th. Then in A^r the user A discards the reply from Th, creates a fresh nonce ε and with the help of the oracle $\mathcal{R}_{Th}$ constructs the term $\widetilde{\Phi}(\varepsilon)_{n_A-p_A+1} = [\{\texttt{KA}_{p_A}, \varepsilon\}_{K_{Th}}, K_{[A]}]_{K^-_{Th}}$ if Th is the first in its request chain, or $\widetilde{\Phi}(\varepsilon)_{n_A-p_A+1} = [\{\texttt{KA}_{p_A}, \varepsilon\}_{K_{Th}}, \Phi_{p_A-1}]_{K^-_{Th}}$ otherwise. Then A^r continues the protocol as in A^l. All the interaction of A with the oracle is not visible to the attacker.

The resultant service token $\widetilde{\Phi}(\varepsilon)$ constructed by the user in A^r does not have A's identity embedded inside it, as opposed to the service token $\widetilde{\Phi}(A)$ constructed in A^l. So if the attacker cannot tell apart the lhs and the rhs processes depicted in the equivalence below then he cannot link the service token with the user's identity embedded inside it.

Comparison with Mukhamedov and Ryan [2007]. In our earlier work, we used a slightly different formulation of the anonymity property, in which two honest users swap their tokens; the anonymity property then says that this swap is indistinguishable to the attacker. In this paper, we use a simpler formulation for anonymity property that we believe captures the essence of the property more faithfully, saying that the attacker cannot distinguish between a genuine and an oracle-faked token. This version is simpler because it involves one user only; and it is more faithful because the anonymity that a user obtains does not rely on the presence of another user doing something different.

THEOREM 1. *Suppose A is an honest user of the protocol, Th is an honest token provider in A's request chain. Then the protocol guarantees user anonymity; that is,*

$$\nu\, sK_{Th}, K^-_A, K^{'-}_{A'}.\ (\nu\ \tilde{n}_A. A^l \ |\ \nu\ \tilde{n}_{A'}.(A' \mid \mathsf{out}(ch, \widetilde{\Psi}_{K'_{[A']}})) \mid !Th)$$
$$\approx$$
$$\nu\, sK_{Th}, K^-_A, K^{'-}_{A'}.\ (\nu\ \tilde{n}_A, s_R. (A^r \mid \mathcal{R}_{Th}) \mid\ \nu\, \tilde{n}_{A'}.(A' \mid \mathsf{out}(ch, \widetilde{\Psi}_{K'_{[A']}})) \mid !Th)$$

where processes $A^l, A^r, \mathcal{R}_{Th}$ are as defined above. sK_{Th} is Th's private decryption key, K_A^- is A's signing key, s_R is a private channel shared between A^r and oracle $\mathcal{R}_{Th}$, and ch is a public channel. $A' = \nu K_{A'}^-.\mathtt{processU}[\{^{K_{A'}^-}/_{signU}\}, \{^{\_}/_{pU}\}, \{^{\_}/_{n}\}]$ is another service user that has arbitrary values for pU and n, and $\widetilde{\Psi}_{K'_{[A']}}$ is a valid complaint related to the service key $K'_{[A']}$ of A' (if $A' = A$ then we have $K_A^- = K_{A'}^{'-}$). We have $K_{[A]} \in \tilde{n}_A$ and $K'_{[A']} \in \tilde{n}_{A'}$; we assume $K'_{[A']} \neq K_{[A]}$.

Proof. We prove labelled bisimilarity between our processes, since observational equivalence $\approx$ coincides with labelled bisimilarity $\approx_\ell$, and the latter relation is easier to reason about by hand. The definition of $\approx_\ell$ requires that every labelled and internal transitions of a process on one side of the equivalence are matched with those of a process on the other side. Furthermore, all the intermediate processes need to be statically equivalent.

In our case the matching of labelled transitions is straightforward, since we have essentially the same processes on both sides of the equivalence (only the data they manipulate are different): the Out-Atom transition only permits outputting terms by reference, so we shall have the same such labels on both sides of the equivalence; and in case of In rule, the same term M will be input on both sides. The communication of A^r with $\mathcal{R}_{Th}$ over the private channel s_R is not visible to the environment – there is no corresponding labelled transition. Instead, it is captured by the internal reduction Comm. Therefore, we match labelled transitions as follows: for A^l's transitions on the lhs with pick those of A^r on the rhs, (and vice versa) and we match the rest with the transitions of the identical process on the other side of the equivalence. The crux of the theorem is in proving the static equivalence of the lhs and the rhs at each step.

The theorem holds if we show that the static equivalence (1) below holds. That is because frames of all intermediate processes resulting from $\approx_\ell$ transitions are subframes in the equivalence (1) and if it is true then all subframes with equal domains are also statically equivalent:

$$\begin{array}{ll} & \nu\, sK_{Th}, K_A^-, K_{A'}^{'-}.\, (\nu\, \tilde{n}_A.(\phi_A^l \mid \{^{\widetilde{\Phi}(A)^l}/_x\}) \mid \nu\, \tilde{n}_{A'}.(\phi_{A'} \mid \{^{\widetilde{\Psi}_{K'_{[A']}}}/_y\}) \mid \\ & \quad (\|_{i\in N}\, \phi_{Th_i}^l) \mid\mid \{^{K_{Th}}/_z\}) \\ \approx_s & \\ & \nu\, sK_{Th}, K_A^-, K_{A'}^{'-}.\, (\nu\, \tilde{n}_A.(\phi_A^r \mid \{^{\widetilde{\Phi}(\varepsilon)^r}/_x\}) \mid \nu\, \tilde{n}_{A'}.(\phi_{A'} \mid \{^{\widetilde{\Psi}_{K'_{[A']}}}/_y\}) \mid \\ & \quad (\|_{i\in N}\, \phi_{Th_i}^r) \mid \{^{K_{Th}}/_z\}) \end{array} \tag{1}$$

where ϕ_A^l, ϕ_A^r are A^l's and A^r's frames respectively. $\widetilde{\Phi}(A)^l$ is a service token output by A^l and $\widetilde{\Phi}(\varepsilon)^r$ is a service token output by A^r. $\phi_{A'}$ is A''s frame and K_{Th} is the public part of Th's encryption key sK_{Th}. N is a set of integers representing the number of times $!Th$ has been instantiated during the process evolution.

We simplify the equivalence (1) by noting that (1) holds if (2) below holds (by application of the evaluation context $C[\_] = \nu K_A^-, K_{A'}^{'-}.\ \_$). Intuitively, this corresponds to the statement that if the attacker cannot distinguish the outputs in (2), even when having access to A's and A''s signing keys, then he cannot tell part the same processes when he does not have access to those keys.

$$\begin{array}{ll} & \nu\, sK_{Th}.\, (\nu\, \tilde{n}_A.(\phi^l_A \mid \{^{\widetilde{\Phi}(A)^l}/_x\}) \mid \nu\, \tilde{n}_{A'}.(\phi_{A'} \mid \{^{\widetilde{\Psi}_{K'_{[A']}}}/_y\}) \mid \\ & \quad (\|_{i\in N}\, \phi^l_{Th_i}) \mid\mid \{^{K_{Th}}/_z\})) \\ \approx_s & \\ & \nu\, sK_{Th}.\, (\nu\, \tilde{n}_A.(\phi^r_A \mid \{^{\widetilde{\Phi}(\varepsilon)^r}/_x\}) \mid \nu\, \tilde{n}_{A'}.(\phi_{A'} \mid \{^{\widetilde{\Psi}_{K'_{[A']}}}/_y\}) \mid \\ & \quad (\|_{i\in N}\, \phi^r_{Th_i}) \mid \{^{K_{Th}}/_z\})) \end{array} \tag{2}$$

The equivalence (2) can now be simplified by omitting process $\nu\, \tilde{n}_{A'}.(\phi_{A'} \mid \{^{\widetilde{\Psi}_{K'_{[A']}}}/_y\})$, since the private names (including sK_{Th}) of other processes do not occur in it. In other words, if we show that (3) below holds, then (2) will also hold by application of the evaluation context $C[\_] = (\nu\, \tilde{n}_{A'}.(\phi_{A'} \mid \{^{\widetilde{\Psi}_{K'_{[A']}}}/_y\})) \mid \_$.

$$\begin{array}{ll} & \nu\, sK_{Th}.\, (\nu\, \tilde{n}_A.(\phi^l_A \mid \{^{\widetilde{\Phi}(A)^l}/_x\}) \mid (\|_{i\in N}\, \phi^l_{Th_i}) \mid \{^{K_{Th}}/_z\})) \\ \approx_s & \\ & \nu\, sK_{Th}.\, (\nu\, \tilde{n}_A.(\phi^r_A \mid \{^{\widetilde{\Phi}(\varepsilon)^r}/_x\}) \mid (\|_{i\in N}\, \phi^r_{Th_i}) \mid \{^{K_{Th}}/_z\})) \end{array} \tag{3}$$

We now need to consider the specifics of our protocol to show (3) holds. We apply several simplifications to the above equivalence (steps (4)-(7) below) by removing parts of the frames that are already available to the environment and, hence, do not affect the anonymity property.

By inspection of the token provider process (Fig. 5), we note that the frames it generates in response to `ITKSig` requests are of the form $\phi^j_{Th_i} = \{^{\{[\{[M_i,L_i]_{K^-_{T_i}},\mathtt{K_i}\}_{K_{Th}},L'_i]_{K^-_{Th}}\}_{\mathtt{K_i}}}/_{t_i}\}$, where $L'_i = \mathtt{snd}(\mathtt{dec}(L_i, sK_{Th}))$, $K_i = \mathtt{thd}(\mathtt{dec}(L_i, sK_{Th}))$ and $L_i = [\{\mathtt{ITKReq}, L'_i, K_i\}_{K_{Th}}]_{K^-_{Th}}$. Let ψ_{l1} be the left-hand and ψ_{r1} be the right-hand processes of the static equivalence (3), and suppose $C_1(j)[\_]$ is a context such that $C_1(j)[\|_{i\in N}\, \phi^j_{Th_i})] = \psi_{j1}$ for $j \in \{l, r\}$. To prove (3), it is sufficient to show

$$C_1(l)[\, \phi^l_{Th_{p_A}} \mid (\mathop{\Big\|}_{i\in N'} \mathit{dec}\text{-}\phi_{Th_i})] \approx_s C_1(r)[\, \phi^r_{Th_{p_A}} \mid (\mathop{\Big\|}_{i\in N'} \mathit{dec}\text{-}\phi_{Th_i})] \tag{4}$$

where $\phi^j_{Th_{p_A}} = \{^{\{[\{\mathtt{KA}^j_{p_A},i_1\}_{K_{Th}},i_2]_{K^-_{Th}}\}_{\mathtt{KA}^j_{p_A}}}/_{t_2}\}$ for $i_1 = \widetilde{\Phi}(A)^j_{n-p_A-1}$, $i_2 = \Phi(A)^j_{p_A-1}$ if $j = l$, else $i_1 = i_2 = \varepsilon$, and $\mathit{dec}\text{-}\phi_{Th_i} = \{^{\mathtt{dec}(L_i,K_{Th})}/_{t_{(i,1)}}\}$, where L is a token request message that originates from the attacker; $N' = N \setminus \{p_A\}$. To see this, note that (3) is obtained by applying the context $\nu t_{(i,1)}.\_ \mid \{^{\{[\{[M_i,L_i]_{K^-_T},\mathtt{thd}(t_{(i,1)})\}_{x_{t_3}},\mathtt{snd}(t_{(i,1)})]_{K^-_{Th}}\}_{\mathtt{thd}(t_{(i,1)})}}/_{t_i}\}$ to each side of (4) for each $i \in N'$. Intuitively, this step helps us to simplify the equivalence by making explicit the decryption service of the Th that is available to the attacker.

Let ψ_{l2} be the left-hand and ψ_{r2} be the right-hand processes of the static equivalence (4), and for $j \in \{l, r\}$ suppose $C_2(j)$ is a context such that $C_2(j)[\{^{\widetilde{\Phi}(i)^j}/_x\}] = \psi_{j2}$, where $i = A$ if $j = l$, else $i = \varepsilon$. The terms $\widetilde{\Phi}(x)^j$ have application of `sign` (by some token provider T) at their outermost level. To prove (4), it is sufficient to prove

$$\begin{array}{ll} & C_2(l)[\{^{\{\widetilde{\Phi}(A)^l_{n-1},\mathtt{KA}^l_{n-1}\}_{K_T}}/_{x_1}\} \mid \{^{K_{[A]}}/_{x_2}\}] \\ \approx_s & C_2(r)[\{^{\{y,\mathtt{KA}^r_{n-1}\}_{K_T}}/_{x_1}\} \mid \{^{K_{[A]}}/_{x_2}\})] \end{array} \qquad (5)$$

where according to our definition of the process A^r if $K_T = K_{Th}$ then $y = \varepsilon$ else $y = \widetilde{\Phi}(\varepsilon)^r_{n-1}$. We have (5)⇒(4), because (4) is obtained by applying the context $\nu x_1, x_2.(\_ \mid \{^{[x_1,x_2]_{K^-}}/_x\}$, where K^- is T's signing key, to each side of (5).

Intuitively, in (5) we *delayered* the term $\widetilde{\Phi}(i)^j$ by breaking it up using equational rewriting and structural equivalence, and then removing a process context from each side. We recursively repeat such delayering of all non-atomic terms of the frames on both sides of the latter equivalence, except for the terms exported by the honest token provider's frames. Delayering is performed until we reach either (i) atomic terms, or (ii) non-atomic terms which are the result of applying encryption with a restricted name as the key argument (that intuitively represent a message encrypted with the honest token provider's public key, or A's session key). For example, removing one layer from $\widetilde{\Phi}(A)^l_{n-k}$ for $n-2 > k > 0$ results in terms $\widetilde{\Phi}(A)^l_{n-k-1}, \Phi(A)^l_k, \mathtt{KA}^l_{k+1}$ and delayering $\Phi(A)^l_k$ in turn results in terms $\Phi(A)^l_{k-1}, \mathtt{KA}^l_k$. Here is the resulting equivalence, which assumes that the honest token provider Th is at the position p_A of A's request chain req_A of the length n_A:

$$\begin{array}{l} C_3(l)[(\|_{0<i<p_A}\{^{\mathtt{KA}^l_i}/_{a_i}\}) \mid \{^{\Phi(A)^l_{p_A}}/_{a_p}\} \mid \{^{\{\mathtt{KA}^l_{p_A},\widetilde{\Phi}(A)^l_{n-p_A-1}\}_{K_{Th}}}/_{a_q}\}] \\ \approx_s \\ C_3(r)[(\|_{0<i<p_A}\{^{\mathtt{KA}^r_i}/_{a_i}\}) \mid \{^{\{\mathtt{ITKReq},\varepsilon,\mathtt{KA}^r_{p_A}\}_{K_{Th}}}/_{a_p}\} \mid \{^{\{\mathtt{KA}^r_{p_A},\varepsilon\}_{K_{Th}}}/_{a_q}\}] \end{array} \qquad (6)$$

where $C_3[\_]$ is the context $\nu sK_{Th}, \tilde{n}_A.\,(\phi^j_{Th_{p_A}} \mid \_)$. Note that *dec*-$\phi_{Th_i}$ also reduce to some of the frames above.

Remark. In the special case when $p_A = n_A$ (i.e. when Th is the last one in A's request chain) the resulting equivalence is slightly simpler and is dealt with in a similar way as below omitting non-applicable steps.

Next, by Lemma 1 we eliminate all substitutions that export session keys. So, equivalence (6) holds if:

$$\begin{array}{l} \nu\, sK_{Th}, \tilde{n}'_l\,.(\{^{\{\mathtt{ITKReq},\,\mathtt{KA}^l_{p_A},\Phi(A)^l_{p_A-1}\}_{K_{Th}}}/_{a_p}\} \mid \{^{\{\mathtt{KA}^l_{p_A},\widetilde{\Phi}(A)^l_{n-p_A-1}\}_{K_{Th}}}/_{a_q}\} \mid \\ (\phi^l_{Th_{p_A}}) \mid \{^{K_{Th}}/_z\} \mid \{^{K_{[A]}}/_{x_2}\}) \\ \approx_s \\ \nu\, sK_{Th}, \tilde{n}'_r\,.(\{^{\{\mathtt{ITKReq},\,\varepsilon,\mathtt{KA}^r_{p_A}\}_{K_{Th}}}/_{a_p}\} \mid \{^{\{\mathtt{KA}^r_{p_A},\varepsilon\}_{K_{Th}}}/_{a_q}\} \mid \\ (\phi^r_{Th_{p_A}}) \mid \{^{K_{Th}}/_z\}) \mid \{^{K_{[A]}}/_{x_2}\}) \end{array} \qquad (7)$$

We have unfolded $\Phi(A)^l_{p_A} = \{\mathtt{ITKReq}, \mathtt{KA}^l_{p_A}, \Phi(A)^l_{p_A-1}\}_{K_{Th}}$, which appears in (6), to elucidate the difference between the lhs and the rhs frames.

Let ψ_{l_3}, ψ_{r_3} be the lhs and the rhs of the equivalence (7), respectively. We normalize those frames and consider each of the substitutions of the resulting equivalence in turn:

—$\{^{\{\mathtt{KA}^j_{p_A},i\}_{K_{Th}}}/_{a_q}\}$, where $i = \widetilde{\Phi}(A)^l_{n-p_A-1}$ if $j = l$, else $i = \varepsilon$. This substitution resulted from delayering the token Th issues to A. We have $\psi_{j_3} \not\vdash sK_{Th}$ as Th's private decryption key never occurs in messages sent between the agents.

Moreover, since all occurrences of $\mathtt{KA}^j_{p_A}$ in ψ_{j_3} are in ciphertexts, either in the main body that is encrypted with K_{Th} or as an encryption key argument, by Lemma 9 we also have $\psi_{j_3} \not\vdash \mathtt{KA}^j_{p_A}$.
The term $\{\mathtt{KA}^j_{p_A}, i\}_{K_{Th}}$ does not occur in other parts of ψ_{j_3}, so by Lemma 6 we have $\psi_{l_3}, \psi_{r_3} \not\vdash \{\mathtt{KA}^j_{p_A}, i\}_{K_{Th}}$. We also note by examination of Th's protocol that it only performs decryption of messages of the form $\{\mathtt{ITKReq}, Z\}_{K_{Th}}$ for some term Z, which it receives as part of a larger input, and so $\mathtt{dec}(a_q, sK_{Th})$ does not occur in ψ_{j_3}. As a consequence, by Lemma 7 we can replace the substitution in question by a substitution of a fresh name on both sides of the equivalence.

—$\{^{\{\mathtt{ITKReq}, \mathtt{KA}^j_{Th}, i\}_{K_{Th}}}/_{a_p}\}$, where $i = \Phi(A)^l_{p_A-1}$ if $j = l$, else $i = \varepsilon$. The exported term is the token A produces in the construction phase of the protocol. As above we have $\psi_{j_3} \not\vdash \mathtt{KA}^j_{p_A}, sK_{Th}$. Since $\{\mathtt{ITKReq}, \mathtt{KA}^j_{Th}, i\}_{K_{Th}}$ does not occur in other parts of ψ_{j_3}, by Lemma 6 it is not derivable from those frames. The frames ψ_{j_3} are closed and in normal form, so $\mathtt{dec}(a_p, sK_{Th})$ does not occur in them. That in turn implies that by Lemma 7, we can replace the substitution in question by a substitution of a fresh name on both sides of the equivalence.

—$\phi^j_{Th_{p_A}} = \{^{\{[\{\mathtt{KA}^j_{p_A}, i_1\}_{K_{Th}}, i_2]_{K^-_{Th}}\}_{\mathtt{KA}^j_{p_A}}}/_{t_2}\}$, where $i_1 = \widetilde{\Phi}(A)^j_{n-p_A-1}$, $i_2 = \Phi(A)^j_{p_A-1}$ if $j = l$, else $i_1 = i_2 = \varepsilon$. It is a reply from Th to A in the second stage of the protocol. As above, $\psi_{j_3} \not\vdash \mathtt{KA}^j_{p_A}, sK_{Th}$ and after the previous two transformations $\mathtt{KA}^j_{p_A}$ does not occur in other parts of ψ_{j_3}. So by Lemma 8 we can replace the exported term of the substitution in question by fresh names on both sides of the equivalence.

Following the above transformations we note that the lhs of the equivalence is α-equivalent to the rhs. Consequently, $\psi_{l_3} \approx_s \psi_{r_3}$.

□

4. PROTOCOL BY MARSHALL AND MOLINA-JIMINEZ

In an earlier work Marshall and Molina-Jiminez [Marshall and Molina-Jiminez 2003] proposed a protocol that aimed to distribute escrowed user identity among several trusted parties using standard cryptographic primitives. However, their scheme requires strong assumptions on the trusted parties and suffers from weaknesses. In this section we briefly detail Marshall and Molina-Jiminez's (MJ) protocol and the problems we have identified with it.

The MJ protocol consists of two parts: sign-up protocol, where users generate anonymous service tokens, and complaint resolution protocol, which is executed to reveal the identity of the offending anonymous user. Those parts are depicted in Protocols 1 and 2, respectively.

In Protocol 1, E stands for a set of adjudicators $\{E_1, \ldots, E_k\}$. The set is initially chosen by S and then $H \subseteq E$ is chosen by A. The set of adjudicators H decide in case of a complaint from S whether it is valid and A's identity should be revealed. $\mathtt{an\_id}$ is a pseudonym that is used by A when accessing services of S; $K_{[A]}$ is the public part of A's anonymous service key; $K_{\widehat{A}}$ is the public part of A's temporary anonymous communication key that each T_{a_i} $(i > 1)$ use to anonymously send messages to A.

1. $A \longrightarrow T_{a_1}$: $\{\ [\ \texttt{ITKReq}\ ]_{K_A^-}\ \}_{K_{T_{a_1}}}$

2. $T_{a_1} \longrightarrow A$: $\{\ \Phi_1\ \}_{K_A}$, where $\Phi_1 = [\ \{K_A\}_{K_{T_{a_1}}}\ ]_{K^-_{T_{a_1}}}$

$$* \begin{cases} \text{1a} \quad A \quad \longmapsto \quad T_{a_{i+1}} \quad : \quad \{\ \texttt{ITKSig},\ \Phi_i\ \}_{K_{T_{a_{i+1}}}} \\ \qquad\qquad\qquad\qquad \text{where} \quad \Phi_i = [\{\ \Phi_{i-1}\}_{K_{T_{a_i}}}\ ]_{K^-_{T_{a_i}}} \\ \text{2a} \quad T_{a_{i+1}} \quad \longrightarrow\!\!\mid \quad A \quad : \quad \{\ [\ \{\ \Phi_i\}_{K_{T_{a_{i+1}}}}\ ]_{K^-_{T_{a_{i+1}}}}\ \}_{K_{\widehat{A}}} \end{cases}$$

3. $A \longmapsto S$: $\{\ \texttt{ServReq}, K_{[A]}, \Phi_A\ \}_S$

4. $S \longrightarrow\!\!\mid A$: $\{\ n, E\ \}_{K_{[A]}}$

5. $A \longmapsto S$: $\{\ H\ \}_S$

6. $S \longrightarrow\!\!\mid A$: $\{\ \texttt{an\_id}\ \}_{K_{[A]}}$

Protocol 1: MJ sign-up protocol

1. $S \longrightarrow E_i$: $\{\ \texttt{AdjReq}, \Psi\ \}_{K_{E_i}}$

2. $E_i \longrightarrow S$: $\{\ [\ V_i\ ]_{K^-_{E_i}}\ \}_{K_S}$

3. $S \longrightarrow T_{a_i}$: $\{\ \texttt{Reveal}, \Phi_i, \Psi, H, V\ \}_{K_{T_{a_i}}}$

4. $T_{a_i} \longrightarrow S$: $\{\ \Phi_{i-1}\ \}_{K_S}$

Protocol 2: MJ complaint resolution protocol

In Protocol 2, Ψ is a complaint that S submits to adjudicators from the set H (chosen in the previous protocol), and to the set of token providers chosen by A. $V = \{V_1, \ldots, V_n\}$ is the set of 'yes/no' votes by $E_i \in H$ that decide whether Ψ is valid or not. If V is positive in the majority, S presents the tuple of signed votes V to each T_{a_i}, in order to reveal the identity of a user associated with the token of the offending user.

4.1 Analysis of MJ protocol

The protocol is subject to the following vulnerabilities:

4.1.1 *Service Misuse. Any of the identity token providers can misuse services of S (or let someone else to do that) and, furthermore, it can implicate any entity of its choice in such a misuse*:

Suppose T_{a_i} is a dishonest token provider. He can present any intermediate token which he receives during the sign-up protocol to S, and obtain an identifier to use the service. He can misuse the service and in doing so implicate the user who initiated the creation of the intermediate token. Moreover, since the identity token takes the form

$$[\ \{\ \ldots [\ \{K_A\}_{K_{T_{a_1}}}\ ]_{K^-_{T_{a_1}}}\ \ldots \}_{K_{T_{a_p}}}\ ]_{K^-_{T_{a_p}}}$$

and T_{a_1} has access to A's public key K_A, he can create $[\ \{K_A\}_{K_{T_{a_1}}}\ ]_{K^-_{T_{a_1}}}$ and anonymously request the signature services from $T_{a_2}, \ldots, T_{a_p}$ in order to create the A's full identity token Φ_A. (Other token providers cannot easily create Φ_A since they don't directly receive $[\mathtt{ITKReq}]_{K^-_A}$).

Clearly, the MJ protocol requires putting full trust on all token providers that are involved in generating Φ_A, since the token is not unforgeably tied to A's real identity.

4.1.2 *Identity Compromise (1). Suppose A has chosen a sequence of token providers $T_{a_1}, T_{a_2}, \ldots, T_{a_p}$, and T_{a_1} is dishonest. Then the service provider S in a coalition with T_{a_1} can identify the identity token Φ_A that A has generated:*

Suppose T_{a_1} is dishonest and reveals to S identity tickets it issues. Then it takes at most n^{k-1} number of operations (`ITKSig` requests and encryptions) for the coalition to find out Φ_A, where n is the total number of identity token providers and k is the length of A's requests chain - a straightforward brute-force search. If we allow the coalition to eavesdrop locally on messages of other token providers T_{a_i}, then the number of operations they need to perform is polynomial in k, namely $n(k-1)$.

4.1.3 *Identity Compromise (2). Suppose S has successfully processed a complaint about a particular user. Then S can reveal the identity of any subsequent user of the service.*

Once S has successfully processed a complaint, he is in possession of the information Ψ, H, V corresponding to the complaint. He can use this to make `Reveal` requests to any sequence of T_i's corresponding to some other protocol session, and thereby break its anonymity.

The MJ protocol has some other weaknesses too: any third party can find out who misused the services of an anonymous service provider S; a dishonest service provider S can change the set H to include more lenient adjudicators, when requesting to reveal the identity of an anonymous user in the complaint resolution protocol.

5. CONCLUSION

In the paper, we proposed a protocol that addresses the problem of anonymity with identity escrow, which is about balancing the need for anonymity with the need for traceability and accountability on the Internet. Our protocol allows online users to access the services anonymously, while providing the possibility that the service owner can break the anonymity if its service is misused. The main feature of our scheme is it provides user anonymity even if all but one escrow holders are dishonest acting in a coalition and it achieves that property using standard cryptographic primitives.

We analysed the anonymity property of the protocol in the applied pi calculus and in the process established compelling reduction criteria for static equivalences in a public-key equational theory. Lastly, we reviewed a protocol proposed by Marshall and Molina-Jiminez [Marshall and Molina-Jiminez 2003] that aimed to achieve goals similar to ours, and showed that their scheme suffers from serious weaknesses.

REFERENCES

ABADI, M., BLANCHET, B., AND FOURNET, C. 2004. Just fast keying in the pi calculus. In *13th European Symposium on Programming (ESOP'04)*, D. Schmidt, Ed. LNCS, vol. 2986. Springer, Barcelona, Spain, 340–354.

ABADI, M. AND FOURNET, C. 2001. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages*, H. R. Nielson, Ed. ACM, London, UK, 104–115.

BLANCHET, B. 2001. An efficient cryptographic protocol verifier based on prolog rules. In *14th IEEE Computer Security Foundations Workshop*, S. Schneider, Ed. IEEE Computer Society Press, Cape Breton, Nova Scotia, Canada, 82–96.

BLANCHET, B. 2004. Automatic Proof of Strong Secrecy for Security Protocols. In *IEEE Symposium on Security and Privacy*. Oakland, California, 86–100.

BUSINESSWEEK. 2000. Business Week/Harris poll: A growing threat. In *Business Week*.

CAMENISCH, J. AND LYSYANSKAYA, A. 2001. An identity escrow scheme with appointed verifiers. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*. Lecture Notes in Computer Science. Springer-Verlag, London, UK, 388–407.

CAMENISCH, J. AND LYSYANSKAYA, A. 2004. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO '04: Proceedings of the 24th Annual International Cryptology Conference on Advances in Cryptology*. LNCS. Springer-Verlag, California, USA, 56–72.

CAMENISCH, J. AND SHOUP, V. 2003. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, D. Boneh, Ed. Lecture Notes in Computer Science, vol. 2729. Springer, 126–144.

CRANOR, L. F., REAGLE, J., AND ACKERMAN, M. 1999. Beyond concern: Understanding net users' attitudes about online privacy. Tech. rep., AT&T Labs-Research.

FOURNET, C. AND ABADI, M. 2003. Hiding names: Private authentication in the applied pi calculus. In *Software Security – Theories and Systems. Mext-NSF-JSPS International Symposium (ISSS'02)*, M. Okada, B. C. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, Eds. LNCS, vol. 2609. Springer, Tokyo, Japan, 317–338.

HARRIS. 1999. IBM multi-national consumer privacy survey. In *Harris Interactive*.

KIAYIAS, A. AND YUNG, M. 2004. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076. `http://eprint.iacr.org/`.

KILIAN, J. AND PETRANK, E. 1998. Identity escrow. In *Advances in Cryptology (CRYPTO'98)*. Number 1462 in LNCS. Springer Verlag, 169–187.

KREMER, S. AND RYAN, M. D. 2005. Analysis of an electronic voting protocol in the applied pi-calculus. In *Programming Languages and Systems — Proceedings of the 14th European Symposium on Programming (ESOP'05)*, M. Sagiv, Ed. LNCS, vol. 3444. Springer, Edinburgh, U.K., 186–200.

LEIGHTON, T. 1994. Failsafe key escrow systems. Technical Memo 483, MIT Laboratory for Computer Science.

MARSHALL, L. AND MOLINA-JIMINEZ, C. 2003. Anonymity with identity escrow. In *Proceedings of the 1st International Workshop on Formal Aspects in Security and Trust*, T. Dimitrakos and F. Martinelli, Eds. Istituto di Informatica e Telematica, Pisa, 121–129.

MICALI, S. 1993. Fair public-key cryptosystems. In *Advances in Cryptology (CRYPTO'92)*. Number 740 in LNCS. Springer Verlag.

MUKHAMEDOV, A. AND RYAN, M. D. 2005. On anonymity with identity escrow. In *Third international Workshop on Formal Aspects in Security and Trust (FAST'05)*. LNCS. Springer.

MUKHAMEDOV, A. AND RYAN, M. D. 2007. On anonymity with identity escrow. In *3rd Symposium on Trustworthy Global Computing (TGC)*. LNCS. Springer.

A. APPLIED PI-CALCULUS

The applied pi calculus [Abadi and Fournet 2001] is a language for describing concurrent processes and their interactions. It is based on the pi calculus, but is intended to be less pure and therefore more convenient to use. Properties of processes described in the applied pi calculus can be proved by employing manual techniques [Abadi and Fournet 2001], or by automated tools such as ProVerif [Blanchet 2001]. As well as reachability properties which are typical of model checking tools, ProVerif can in some cases prove that processes are observationally equivalent [Blanchet 2004]. This capability is important for anonymity-type properties such as the one we study here. The applied pi calculus has been used to study a variety of security protocols, such as those for private authentication [Fournet and Abadi 2003], for key establishment [Abadi et al. 2004], as well as an electronic voting protocol [Kremer and Ryan 2005].

To describe processes in the applied pi calculus, one starts with a set of *names* (which are used to name communication channels or other constants), a set of *variables*, and a *signature* Σ which consists of the function symbols which will be used to define terms. In the case of security protocols, typical function symbols will include *enc* for encryption, which takes plaintext and a key and returns the corresponding cipher text, and *dec* for decryption, taking ciphertext and a key and returning the plaintext. Terms are defined as names, variables, and function symbols applied to other terms. Terms and function symbols can be sorted, and function symbol application must then respect sorts. An equational theory E is a a signature Σ together with a set of equations which hold on terms constructed from the signature. We denote $=_E$ the smallest equivalence relation that includes the equations in E and is closed under substitutions of arbitrary terms for names and variables, and application of contexts. A typical example of an equational theory useful for cryptographic protocols is

$$dec(enc(x, k), k) = x.$$

For example, given this theory and terms $T_1 = dec(enc(enc(n, k_1), k_2), k_2)$ and $T_2 = enc(n, k_1)$, we have that $T_1 =_E T_2$ (while obviously the syntactic equality does not hold). We write $M == N$ if M is syntactically equivalent to N. By orienting the equations in the equational theory from left to right one can obtain a rewriting system R_E. If R_E is convergent, we say that E is convergent; and in that case, all terms have unique normal forms. $N{\downarrow}$ denotes normal form of N. A frame is in normal form if all the terms occurring in it are in normal form. The standard public key encryption equational theory E_{pk} which includes projections, pairing, digital signing, public and session key encyption, decryption functions is known to be convergent.

In the applied pi calculus, one has (plain) processes and extended processes. Plain processes are built up in a similar way to processes in the pi calculus, except that messages can contain terms (rather than just names). In the grammar described below, M and N are terms, n is a name, x a variable and u is a metavariable, *i.e.* a name or a variable.

$P, Q, R :=$	plain processes
0	null process

$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction
if $M = N$ then P else Q	conditional
$\mathsf{in}(u, x).P$	message input
$\mathsf{out}(u, N).P$	message output

Extended processes add *active substitutions* and restriction on variables:

$A, B, C :=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/_x\}$	active substitution

$\{^M/_x\}$ is the substitution that replaces the variable x with the term M. Active substitutions generalise "let". The process $\nu x.(\{^M/_x\} \mid P)$ corresponds exactly to "let $x = M$ in P". As usual, names and variables have scopes, which are delimited by restrictions and by inputs. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of free and bound variables and free and bound names of A, respectively. We say that an extended process is closed if all its variables are either bound or defined by an active substitution.

Active substitutions are useful because they allow us to map an extended process A to its *frame* $\phi(A)$ by replacing every plain process in A with 0. A frame is an extended process built up from 0 and active substitutions by parallel composition and restriction. The frame $\phi(A)$ can be viewed as an approximation of A that accounts for the static knowledge A exposes to its environment, but not A's dynamic behaviour. Every frame can be written in the form $\varphi = \nu\tilde{n}.\{T_1/x_1, \dots, T_n/x_n\}$. The domain of φ is $dom(\varphi) = \{x_1, \dots, x_n\}$. We say that φ is closed if all the terms T_i are closed, *i.e.* contain no variables.

Definition 3. *We say two terms M, N are equal in the frame φ, and write $(M = N)\varphi$, if $\varphi \equiv \nu\tilde{n}.\sigma$, $M\sigma =_E N\sigma$, and $\tilde{n} \cap fn(M, N) = \emptyset$ for some names $\tilde{n}$ and substitution σ.*

A context $C[\cdot]$ is a process with a hole; an evaluation context is a context whose hole is not under a replication, a conditional, an input, or an output. Structural equivalence, noted $\equiv$, is the smallest equivalence relation on extended processes that is closed under α-conversion on names and variables, by application of evaluation contexts, and satisfying some further basic structural rules, namely, for all names n, extended processes A, B, plain processes P, and variables or names u, v:

—$!P \equiv !P \mid P$; $A \mid 0 \equiv A$;

—associativity and commutativity of $\mid$;

—$\nu n.0 \equiv 0$; $\nu u.\nu w.A \equiv \nu w.\nu u.A$;

—$A \mid \nu u.B \equiv \nu u.(A \mid B)$ where $u \notin fn(A) \cup fv(A)$;

—$\nu x.\{^M/_x\} \equiv 0$; $\{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}$;
—$\{^M/_x\} \equiv \{^N/_x\}$ whenever $M =_E N$.

We can now define what it means for two frames to be statically equivalent.

Definition 4 Static equivalence. *Two closed frames φ_1 and φ_2 are* statically equivalent, *written $\varphi_1 \approx_s \varphi_2$, iff for some names $\tilde{n}$, and substitutions σ_1, σ_2, such that $\varphi_1 \equiv \nu\tilde{n}.\sigma_1$ and $\varphi_2 \equiv \nu\tilde{n}.\sigma_2$, we have that:*

(i) $\mathrm{dom}(\varphi_1) = \mathrm{dom}(\varphi_2)$,
(ii) for all terms M, N with variables included in $\mathrm{dom}(\varphi_i)$ and using no names occurring in $\tilde{n}$, $M\sigma_1 =_E N\sigma_1$ is equivalent to $M\sigma_2 =_E N\sigma_2$.

We also say that two extended processes A and B are statically equivalent if and only if $\phi(A) \approx_s \phi(B)$.

Example 1. *Consider $\varphi_0 = \nu k.\{enc(s_0, k)/x_1, k/x_2\}$ and $\varphi_1 = \nu k.\{enc(s_1, k)/x_1, k/x_2\}$ where s_1, s_2 and k are names. Let E be the theory defined by the axiom $dec(enc(x, k), k) = x$. We have $(dec(x_1, x_2) =_E s_0)\varphi_0$ but not $(dec(x_1, x_2) =_E s_0)\varphi_1$. Therefore, $\varphi_0 \not\approx_s \varphi_1$ although $\nu k.\{enc(s_0, k)/x_1\} \approx_s \nu k.\{enc(s_1, k)/x_1\}$.*

The operational semantics of processes in the applied pi calculus is defined by structural rules defining two relations: *structural equivalence* (described above) and *internal reduction*, noted $\rightarrow$. Internal reduction $\rightarrow$ is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that $\bar{a}\langle x\rangle.P \mid a(x).Q \rightarrow P \mid Q$ and for all terms M, N, K such that N, K are ground and $N \neq_E K$,

$$\begin{array}{r}\text{if } M = M \text{ then } P \text{ else } Q \;\rightarrow\; P\\ \text{if } N = K \text{ then } P \text{ else } Q \;\rightarrow\; Q.\end{array}$$

Definition 5 Deducibility. *Given a frame φ that represents a history of messages output during the execution of the processes, we define the* deduction *relation, denoted by $\varphi \vdash M$. Deducible messages are those that can be derived from φ by application of function symbols and the equational theory E.*

$$\frac{}{\nu\tilde{n}.\,\sigma \vdash x\sigma}[x \in dom(\sigma)] \qquad \frac{}{\nu\tilde{n}.\,\sigma \vdash m}[m \in \mathcal{N} \setminus \tilde{n}]$$

$$\frac{\nu\tilde{n}.\,\sigma \vdash T_1 \quad \ldots \quad \nu\tilde{n}.\,\sigma \vdash T_l}{\nu\tilde{n}.\,\sigma \vdash f(T_1, \ldots, T_l)} \qquad \frac{\nu\tilde{n}.\,\sigma \vdash T \qquad T =_E T'}{\nu\tilde{n}.\,\sigma \vdash T'}$$

Example 2. *n is deducible from the frame $\nu n, k'.(\{^{enc(n,k')}/_x\} \mid \{^{k'}/_y\})$, but not from the frame $\nu n.\{^{f(n)}/_x\}$, where f is a hash function (i.e., there are no equations involving f).*

Proposition A.0.1. *Let $\varphi = \nu\tilde{n}.\sigma$ be a frame and M be a term. $\varphi \vdash M$ if and only if there exists a term T, such that $fn(T) \cap \tilde{n} = \emptyset$ and $T\sigma =_E M$.*

We further extend the operational semantics by a labeled operational semantics enabling us to reason about processes that interact with their environment. Labeled operational semantics defines the relation $\xrightarrow{\alpha}$ where α is either an input or the output of a channel name or a variable of base type. More details can be found in [Abadi and Fournet 2001].

$$\mathsf{in}(c,x).P \xrightarrow{\mathsf{in}(c,M)} P\{^M/_x\} \quad [\text{In}] \qquad \mathsf{out}(c,u).P \xrightarrow{\mathsf{out}(c,u)} P \quad [\text{Out-Atom}]$$

$$\frac{A \xrightarrow{\mathsf{out}(c,u)} A'}{\nu u.A \xrightarrow{\nu u.\mathsf{out}(c,u)} A'}[\text{Open-Atom}] \qquad \frac{A \xrightarrow{\alpha} A'}{\nu u.A \xrightarrow{\alpha} \nu u.A'}[\text{Scope}]$$

$$\frac{A \xrightarrow{\alpha} A'}{A \mid B \xrightarrow{\alpha} A' \mid B}[\text{Par}] \qquad \frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'}[\text{Struct}]$$

where u is a metavariable that ranges over names and variables, and Par rule is restricted to $bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset$.

Definition 6 Labeled bisimilarity ($\approx_\ell$). *Labeled bisimilarity is the largest symmetric relation $\mathcal{R}$ on closed extended processes, such that $A\ \mathcal{R}\ B$ implies:*

(1) $A \approx_s B$;

(2) if $A \to A'$, then $B \to^* B'$ and $A'\ \mathcal{R}\ B'$ for some B';

(3) if $A \xrightarrow{\alpha} A'$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A'\ \mathcal{R}\ B'$ for some B'.

In [Abadi and Fournet 2001], it is shown that labeled bisimilarity coincides with observational equivalence, which is used to formalize many security properties, and in particular anonymity properties. In this work we prefer to use labeled bisimilarity, rather than observational equivalence, because proofs for labeled bisimilarity are easier for us to carry out.

B. PROOFS OF LEMMAS

Lemma 1. *Let E be an equational theory, φ, φ' be frames, $\tilde{n}, \tilde{n}'$ be sets of names and k a name s.t. $k \notin fn(\varphi, \varphi') \cap (\tilde{n} \cup \tilde{n}')$. If $\nu\tilde{n}.\varphi \approx_s \nu\tilde{n}'.\varphi'$ then $\nu\tilde{n}, k.(\{^k/_x\} \mid \varphi) \approx_s \nu\tilde{n}', k.(\{^k/_x\} \mid \varphi')$, where $x \notin dom(\varphi)$.*

Proof. Let $\phi_l \equiv \nu\tilde{m}_l.\sigma_l$ and $\phi_r \equiv \nu\tilde{m}_r.\sigma_r$ be the lhs and the rhs of the proposed equivalence, respectively. Suppose $\nu\tilde{n}.\varphi \equiv \nu\tilde{m}.\sigma$ with $\tilde{m} = \tilde{m}_l \setminus \{k\}$, since by our hypothesis $k \notin fn(\varphi, \varphi') \cap (n \cup n')$, and let $\nu\tilde{n}.\varphi \approx_s \nu\tilde{n}'.\varphi'$. We have $\sigma_l = \sigma \mid \{^k/_x\}$.

Take two terms M, N such that $fn(M, N) \cap \tilde{m}_l = \emptyset$. If $(M = N)\phi_l$ then $M\sigma_l =_E N\sigma_l$, and so $(M\sigma)\{^k/_x\} =_E (N\sigma)\{^k/_x\}$. Since $fv(\sigma) = \emptyset$ and $x \notin dom(\varphi)$, we have $(M\sigma)\{^k/_x\} == (M\{^k/_x\})\sigma$, $(N\sigma)\{^k/_x\} == (N\{^k/_x\})\sigma$, and hence $(M\{^k/_x\})\sigma =_E (N\{^k/_x\})\sigma$. As $k \notin \tilde{m}$ we get $((M\{^k/_x\}) = (N\{^k/_x\}))\nu\tilde{n}.\varphi$.

Now, if $(M = N)\nu\tilde{n}.\varphi$ then $(M\sigma)\{^k/_x\} = (N\sigma)\{^k/_x\}$, and hence $(M = N)\phi_l$, since equational theory is closed under substitution of arbitrary terms for variables. So, $(M = N)\phi_l$ iff $(M = N)\nu\tilde{n}.\varphi$ and similarly we have $(M = N)\phi_r$ iff $((M\{^k/_x\}) = (N\{^k/_x\}))\nu\tilde{n}'.\varphi'$, which imply $(M = N)\phi_l$ iff $(M = N)\phi_r$. Hence, $\phi_l \approx_s \phi_r$. $\Box$

Lemma 2. *Consider a convergent equational theory E, a closed term L in normal form, names $\tilde{n}, s$ and a frame φ in normal form such that $s \notin fn(\varphi)$. Suppose that:*

—*L does not occur in φ, and $\nu\tilde{n}.\varphi \not\vdash L$.*

—*for any $\tilde{m}, \sigma, M, N$ such that $\nu\tilde{n}.(\{^L/_x\}|\varphi) \equiv \nu\tilde{m}.\sigma$, $(fn(M) \cup fn(N)) \cap \tilde{m} = \emptyset$ and $M\sigma =_E N\sigma$ we have $M\sigma\{^z/_L\} =_E N\sigma\{^z/_L\}$.*

Then: $\nu\tilde{n}.(\{^L/_x\}|\varphi) \approx_s \nu\tilde{n}, s.(\{^s/_x\}|\varphi)$.

Proof. Let ϕ_l, ϕ_r denote the right and left parts of the equivalence, respectively. We need to show that for any terms J, K we have $(J = K)\phi_l$ iff $(J = K)\phi_r$. Here, $(J = K)\phi$ means that there exists $\tilde{n}$ and σ such that $\phi \equiv \nu\tilde{n}.\sigma$, $\tilde{n} \cap (fv(J) \cup fv(K)) = \emptyset$, and $J\sigma =_E K\sigma$.

"⇒" direction. Suppose $(J = K)\phi_l$. Then, $\exists\tilde{m}, \sigma$ such that $\phi_l \equiv \nu\tilde{m}.\sigma$, $J\sigma =_E K\sigma$ and $\tilde{m} \cap (fn(J) \cup fn(K)) = \emptyset$. By hypothesis, $(J\sigma)\{^z/_L\} =_E (K\sigma)\{^z/_L\}$, and hence, $((J\sigma)\{^z/_L\})\{^s/_z\} =_E ((K\sigma)\{^z/_L\})\{^s/_z\}$, where $s \notin fn(K) \cup fn(J)$.

Wlog suppose φ is a substitution σ_φ, then $\sigma = \sigma_\varphi \cup \{^L/_x\}$. Consider terms $J\sigma_\varphi$ and $K\sigma_\varphi$. Since L does not occur in φ and $\nu\tilde{n}.\varphi \not\vdash L$, L does not occur in $J\sigma_\varphi$ and $K\sigma_\varphi$. Hence, $((J\sigma)\{^z/_L\})\{^s/_z\} = (((J\sigma_\varphi)\{^L/_x\})\{^z/_L\})\{^s/_z\} = (J\sigma_\varphi)\{^s/_x\}$, and similarly $((K\sigma)\{^z/_L\})\{^s/_z\} = (K\sigma_\varphi)\{^s/_x\}$. We have $\phi_r \equiv \nu\tilde{m}'.\sigma'$, where $\tilde{m}' \cap fn(J, K) = \emptyset$ and $\sigma' = \sigma_\varphi \cup \{^s/_x\}$. Therefore, $(J = K)\phi_r$.

"⇐" direction. Suppose $(J = K)\phi_r$. Then $\exists\tilde{m}, \sigma$ such that $\phi_r \equiv \nu\tilde{m}.\sigma$, $J\sigma =_E K\sigma$ and $\tilde{m} \cap fn(J, K) = \emptyset$. Wlog suppose φ is a substitution σ_φ, then $\sigma = \sigma_\varphi \cup \{^s/_x\}$, where $s \in \tilde{m}$, and $(J\sigma_\varphi)\{^s/_x\} =_E (K\sigma_\varphi)\{^s/_x\}$, and also $J\sigma_\varphi =_E K\sigma_\varphi$ by our assumption 1 that the equational theory is closed under substitution of arbitrary terms for names. So, $(J\sigma_\varphi)\{^L/_x\} =_E (K\sigma_\varphi)\{^L/_x\}$. We have $\phi_l \equiv \nu\tilde{m}'.\sigma'$, where $\tilde{m}' \cap fn(J, K) = \emptyset$ and $\sigma' = \sigma_\varphi\{^L/_x\}$. Therefore, $(J = K)\phi_l$. □

Lemma 3. *Consider the equational theory E_{pk}, and let M, N, L and J be terms in normal form s.t. M, N do not contain $\mathtt{dec}(x, J)$ and $M\{^{\{L\}_J}/_x\} =_E N\{^{\{L\}_J}/_x\}$. Then:*

$$(M\{^{\{L\}_J}/_x\})\{^z/_{\{L\}_J}\} =_E (N\{^{\{L\}_J}/_x\})\{^z/_{\{L\}_J}\}$$

Proof. Since M, L and J are in normal form and $\mathtt{dec}(x, k)$ does not occur in M, the term $M\{^{\{L\}_J}/_x\}$ is in normal form. Similarly, $N\{^{\{L\}_J}/_x\}$ is also in normal form. We have $M\{^{\{L\}_J}/_x\} =_E N\{^{\{L\}_J}/_x\}$ and since normal forms are unique in E_{pk}, we have $M\{^{\{L\}_J}/_x\} == N\{^{\{L\}_J}/_x\}$. Since $==$ is closed under replacement, we can conclude. □

Lemma 4. *Consider the equational theory E_{pk}, a name k, and a frame $\nu\tilde{n}.\sigma$ in normal form that does not contain $\mathtt{dec}(x, k)$, such that $\nu\tilde{n}.\sigma \not\vdash k$. If M is a term such that $\tilde{n} \cap fn(M) = \emptyset$, then $\mathtt{dec}(x, k)$ does not occur in $M\sigma\!\downarrow$.*

Proof. The proof is by induction on M.

Base case: $M == u$. If $u \notin dom(\sigma)$ then there is nothing to do, else $u\sigma == N$, where $\{^N/_u\}$ is a substitution in σ. N is in normal form, since σ is in normal form, and it does not contain $\mathtt{dec}(x, k)$ as $\mathtt{dec}(x, k)$ does not occur in σ.

Induction step. Suppose for some M, N, $\tilde{n} \cap fn(M,N) = \emptyset$, $\mathtt{dec}(x,k)$ does not occur in $M\sigma\downarrow$ and $N\sigma\downarrow$. Then we need to consider the following cases:

—$M' == f(M,N)$, where f is $\mathtt{enc}, \mathtt{pair}$ or $\mathtt{sign}$: we have $M'\sigma\downarrow == f(M\sigma\downarrow, N\sigma\downarrow)$. So, by the IH $\mathtt{dec}(x,k)$ does not occur in $M'\sigma\downarrow$.

—$M' == f(M)$, where f is a unary function in E_{pk}. $M'\sigma\downarrow$ is $f(M\sigma\downarrow)$ or M'', where M'' is a subterm of $M\sigma\downarrow$ or $pk(N)$ for some subterm N of $M\sigma\downarrow$. By our IH $\mathtt{dec}(x,k)$ does not occur in $M'\sigma\downarrow$.

—$M' == \mathtt{dec}(M,N)$. $M'\sigma\downarrow == \mathtt{dec}(M\sigma\downarrow, N\sigma\downarrow)$ or $M'\sigma\downarrow$ is a subterm of $M\sigma\downarrow$. The term $N\sigma\downarrow$ is distinct from k, since $\nu\tilde{n}.\sigma \not\vdash k$. Again, by the IH $\mathtt{dec}(x,k)$ does not occur in $M'\sigma\downarrow$.

□

Lemma 5. *Consider the equational theory E_{pk}, a frame $\nu\tilde{n}.\sigma$ in normal form, and a name $k \in \tilde{n}$, s.t. k occurs in σ only as a key argument to the encryption function (that is, only in the form $\{\_\}_k$). If M is a term such that $\tilde{n} \cap fn(M) = \emptyset$, then $\mathtt{dec}(x,k)$ does not occur in $M\sigma\downarrow$.*

Proof. The proof is by induction on M.

Base case: $M == u$. If $u \notin dom(\sigma)$ then there is nothing to do, else $u\sigma == N$, where $\{^N/_u\}$ is a substitution in σ. N is in normal form, since σ is in normal form, and it does not contain $\mathtt{dec}(x,k)$ by our assumption.

Induction step. Suppose for some M, N, $\tilde{n} \cap fn(M,N) = \emptyset$, $\mathtt{dec}(x,k)$ does not occur in $M\sigma\downarrow$ and $N\sigma\downarrow$. Then we need to consider the following cases:

—$M' == f(M,N)$, where f is $\mathtt{enc}, \mathtt{pair}$ or $\mathtt{sign}$: we have $M'\sigma\downarrow == f(M\sigma\downarrow, N\sigma\downarrow)$. So, by the IH $\mathtt{dec}(x,k)$ does not occur in $M'\sigma\downarrow$.

—$M' == f(M)$, where f is a unary function in E_{pk}. $M'\sigma\downarrow$ is $f(M\sigma\downarrow)$, M'', or $pk(M'')$, where M'' is a subterm of $M\sigma\downarrow$. By our IH $\mathtt{dec}(x,k)$ does not occur in $M'\sigma\downarrow$.

—$M' == \mathtt{dec}(M,N)$. $M'\sigma\downarrow == \mathtt{dec}(M\sigma\downarrow, N\sigma\downarrow)$ or $M'\sigma\downarrow$ is a subterm of $M\sigma\downarrow$. The term $N\sigma\downarrow$ is distinct from k, since $\nu\tilde{n}.\sigma \not\vdash k$, which easily follows from our assumption that k occurs in σ only as an encryption key argument. So, again by the IH $\mathtt{dec}(x,k)$ does not occur in $M'\sigma\downarrow$.

□

Lemma 6. *Consider the equational theory E_{pk}, a closed term L in normal form, names $\tilde{n}, s$, and a frame $\nu\tilde{n}.\sigma$ in normal form. Suppose $\nu\tilde{n}.\sigma \not\vdash s$ and $\{s, L\}_{pk(k)}$ does not occur in σ. Then $\nu\tilde{n}.\sigma \not\vdash \{s, L\}_{pk(k)}$.*

Proof. Let $L' = \{s, L\}_{pk(k)}$. It is known that $\nu\tilde{n}.\sigma \not\vdash L'$ if for any M, s.t. $fn(M) \cap \tilde{n} = \emptyset$, $M\sigma \neq_E L'$. The latter inequality will hold if we show that $M\sigma\downarrow \neq_E L'$. We proceed by induction on M.

Base case: $M == u$. If $u \notin dom(\sigma)$ then there is nothing to do, else $u\sigma == N$, where $\{^N/_u\}$ is a substitution in σ. As σ is in normal form, all subterms of N are also in normal form. We note that L' is in normal form too because L is in normal form by assumption. Furthermore, every subterm K of N is distinct from L', as L' does not occur in σ. As a result, for all such K, $K \neq_E L'$ by the fact that normal forms are unique.

Induction step. Take some M, N such that $\tilde{n} \cap fn(M, N) = \emptyset$. By the facts that $s \in \tilde{n}$ and $s \notin fn(\sigma)$, we see that for all subterms K of $M\sigma\!\downarrow$ and $N\sigma\!\downarrow$, we have $K \neq_E L'$. Then we need to consider the following cases:

—$M' == f(M, N)$, where f is $\mathtt{pair}$ or $\mathtt{sign}$: we have $M'\sigma\!\downarrow == f(M\sigma\!\downarrow, N\sigma\!\downarrow)$. So, by the IH for all subterms K of $M'\sigma\!\downarrow$ we get $K \neq_E L'$.

—$M' == \mathtt{dec}(M, N)$: $M'\sigma\!\downarrow == \mathtt{dec}(M\sigma\!\downarrow, N\sigma\!\downarrow)$ or it is M'', where M'' is a subterm of $M\sigma\!\downarrow$. In any case, by the IH for all subterms K of $M'\sigma\!\downarrow$ we get $K \neq_E L'$.

—$M' == f(M)$, where f is a unary function in E_{pk}. $M'\sigma\!\downarrow$ is $f(M\sigma\!\downarrow)$, M'' or $pk(M'')$, where M'' is a subterm of $M\sigma\!\downarrow$. By our IH for all subterms K of $M'\sigma\!\downarrow$ we get $K \neq_E L'$.

—$M' == \mathtt{enc}(M, N)$. $M'\sigma\!\downarrow == \mathtt{enc}(M\sigma\!\downarrow, N\sigma\!\downarrow)$. The term $M\sigma\!\downarrow \neq_E \{s, L\}$, since otherwise $\mathtt{fst}(M\sigma\!\downarrow) =_E s$ contradicting our assumption $\nu\tilde{n}.\sigma \not\vdash s$. So, again by the IH for all subterms K of $M'\sigma\!\downarrow$ we get $K \neq_E L'$.

□

LEMMA 7. *Given a closed term L in normal form, names $\tilde{n}, s$ and a frame $\nu\tilde{n}.\sigma$ in normal form, suppose:*

(1) $\nu\tilde{n}.\sigma \not\vdash k$, $\nu\tilde{n}.\sigma \not\vdash \{s, L\}_{pk(k)}$ *and* $m \notin fn(\sigma)$

(2) $\{s, L\}_{pk(k)}$ *does not occur in* σ.

(3) $\mathtt{dec}(x, k)$ *does not occur in* σ.

Then $\nu\tilde{n}, s.(\{^{\{s,L\}_{pk(k)}}/_x\}|\sigma) \approx_s \nu\tilde{n}, m.(\{^m/_x\}|\sigma)$.

PROOF. Take terms M, N such that $fn(M, N) \cap \tilde{n} = \emptyset$ and $M\sigma' =_E N\sigma'$, where $\sigma' = \sigma \cup \{^{\{s,L\}_{pk(k)}}/_x\}$. We show that any $M\sigma'\{^z/_{\{s,L\}_{pk(k)}}\} =_E N\sigma'\{^z/_{\{s,L\}_{pk(k)}}\}$.

The substitution σ is in normal form and it does not contain $\mathtt{dec}(x, k)$ by assumption 3. Since $\nu\tilde{n}.\sigma \not\vdash k$ by Lemma 4 we have $M\sigma\!\downarrow$, $N\sigma\!\downarrow$ do not contain $\mathtt{dec}(x, k)$. The term $\{s, L\}_{pk(k)}$ is in normal form, since L is in normal form by assumption. As a result, by Lemma 3 $((M\sigma\!\downarrow)\{^{\{s,L\}_{pk(k)}}/_x\})\{^z/_{\{s,L\}_{pk(k)}}\} =_E ((N\sigma\!\downarrow)\{^{\{s,L\}_{pk(k)}}/_x\})\{^z/_{\{s,L\}_{pk(k)}}\}$, and hence $((M\sigma)\{^{\{s,L\}_{pk(k)}}/_x\})\{^z/_{\{s,L\}_{pk(k)}}\} =_E ((N\sigma)\{^{\{s,L\}_{pk(k)}}/_x\})\{^z/_{\{s,L\}_{pk(k)}}\}$.

By our assumptions $\nu\tilde{n}.\varphi \not\vdash \{s, L\}_{pk(k)}$ and $\{s, L\}_{pk(k)}$ does not occur in σ, so by Lemma 2 it follows that our lemma holds.

□

LEMMA 8. *Given a closed term L in normal form, names $k, s \in \tilde{n}$ and a frame $\nu\tilde{n}.\sigma$ also in normal form, suppose:*

(1) *k occurs in σ only as an encryption key argument, i.e. in the form $\{\_\}_k$;*
(2) *L does not occur in σ and $s \notin fn(\sigma)$.*

Then: $\nu\tilde{n}.(\{^{\{L\}_k}/_x\}|\sigma) \approx_s \nu\tilde{n}, s.(\{^s/_x\}|\sigma)$.

PROOF. Take terms M, N such that $fn(M,N) \cap \tilde{m} = \emptyset$ and $M\sigma' =_E N\sigma'$, where $\sigma' = \sigma\{^{\{L\}_k}/_x\}$. We show that $M\sigma'\{^z/_{\{L\}_k}\} =_E N\sigma'\{^z/_{\{L\}_k}\}$.

The substitution σ is in normal form, where k occurs only as an encryption key argument so by Lemma 5 $M\sigma\downarrow$, $N\sigma\downarrow$ do not contain $\texttt{dec}(\texttt{x},\texttt{k})$. Since L is in normal form using Lemma 3 we conclude $M\sigma'\{^z/_{\{L\}_k}\} =_E N\sigma'\{^z/_{\{L\}_k}\}$.

From our assumptions 1 and 2 it follows that $\nu\tilde{n}.\sigma \not\vdash \{L\}_k$ and $\{L\}_k$ does not occur in σ, so by Lemma 2 our lemma holds. □

LEMMA 9. *Given a frame $\nu\tilde{n}.\sigma$ in normal form and $s, k \in \tilde{n}$, where $\nu\tilde{n}.\sigma \not\vdash k$, suppose for all occurences of s in σ:*

(1) *Either there exists a term L such that $\{L\}_{pk(k)}$ occurs in σ and the occurrence of s is a subterm of L.*
(2) *Or s occurs in σ as an encryption key argument.*

Then $\nu\tilde{n}.\sigma \not\vdash s$.

PROOF. We know that $\nu\tilde{n}.\sigma \not\vdash s$ if for any M, s.t. $fn(M) \cap \tilde{n} = \emptyset$, $M\sigma \neq_E s$. The latter inequality will hold if we show that $M\sigma\downarrow \neq_E s$. We proceed by induction on M.

Base case: $M == u$. If $u \notin dom(\sigma)$ then there is nothing to do, else $u\sigma == N$, where $\{^N/_u\}$ is a substitution in σ. As by assumption σ is in normal form, N is in normal form. As a result, by our hypothesis and the fact that normal forms are unique in this case $M\sigma\downarrow \neq s$.

Induction step. Take a term M' and suppose for all M, N, $\tilde{n} \cap fn(M,N) = \emptyset$, such that M, N are subterms of M' we have $M\sigma\downarrow \neq s$ and $N\sigma\downarrow \neq s$. Then we need to consider the following cases:

—$M' == f(M,N)$, where f can be $\texttt{pair}$, $\texttt{enc}$ or $\texttt{sign}$: we have $M'\sigma\downarrow == f(M\sigma\downarrow, N\sigma\downarrow)$. So, $M'\sigma\downarrow \neq s$.

—$M' == \texttt{dec}(M,N)$: $M'\sigma\downarrow == \texttt{dec}(M\sigma\downarrow, N\sigma\downarrow)$ or it is M'', where M'' is a subterm of $M\sigma\downarrow$. In the former case we are done, and in the latter we note that $M'\sigma == \texttt{dec}(\texttt{enc}(K,L), N\sigma)$, where $L = pk(N\sigma)$. $N\sigma \neq_E k$ since otherwise that would violate our assumption $\nu\tilde{n}.\sigma \not\vdash k$ and hence $L \neq pk(k)$.
Note $M\sigma == \texttt{enc}(K,L)$. Since σ is in normal form if K is not in normal form then $\exists K'$ s.t. $K == K'\sigma$ and K' is a subterm of M. By the IH $K\downarrow \neq s$, and therefore in this case $M'' \neq s$. If K is in normal form then by $L \neq pk(k)$ and by our assumptions $K \neq s$, and again $M'' \neq s$.

—$M' == f(M)$, where f is a unary function. $M'\sigma\downarrow$ is $f(M\sigma\downarrow)$, $pk(M'')$ or M'', where M'' is a subterm of $M\sigma\downarrow$. In the former two cases we are done. In the

latter $M'\sigma == f(f'(K))$, where f' is a function that is not `enc` (it may have arity more than one).
Note $M\sigma == f'(K)$. Since σ is in normal form if K is not in normal form then $\exists K'$ s.t. $K == K'\sigma$ and K' is a subterm of M. By the IH $K{\downarrow} \neq s$, and therefore in this case $M'' \neq s$. If K is in normal form then by our assumptions $K \neq s$, and again $M'' \neq s$.

□

Received Month Year; revised Month Year, accepted Month Year