On Anonymity with Identity Escrow

Aybek Mukhamedov and Mark D. Ryan

School of Computer Science
University of Birmingham
{A.Mukhamedov,M.D.Ryan}@cs.bham.ac.uk

Abstract. *Anonymity with identity escrow* attempts to allow users of a 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. A protocol for achieving anonymity with identity escrow has been presented by Marshall and Molina-Jiminez. In this paper, we show that that protocol suffers from some serious flaws. We also identify some other less significant weaknesses of the protocol, and we present an improved protocol which fixes these flaws. Our improved protocol guarantees anonymity even if all but one of the escrow holders are corrupt.

1 Introduction

Users of services such as opinion surveys or media downloads may wish to remain anonymous to the service provider. However, it may be important for service providers to be able to break anonymity in special circumstances – for example, to assist a criminal investigation. *Identity escrow* is designed to permit these two aims. The notion was first introduced by Kilian and Petrank in [2], which was motivated by the ideas from *key escrow encryption systems* (e.g. [3], [5]). It allows an agent A to use services provided by S without revealing her identity to S, while allowing S to obtain it in pre-agreed special circumstances (e.g. in a misuse), thus offering a balance between privacy and monitoring/accountability. To achieve this, A places her identity in an *escrowed certificate* generated with a trusted *issuer*, which she presents to S when requesting its services. The certificate is *verifiable*, viz. S is given a guarantee that the escrowed certificate is valid and that the identity is recoverable (with a help of *escrow agent*, which is the same as issuer in our exposition).

Clearly, this identity escrow system breaks down if issuer is dishonest, and to address this problem Marshall and Molina-Jiminez [4] proposed a protocol for anonymity with identity escrow, where escrowed certificate is generated by a set of issuers (named *identity token providers* in their paper). Neither S nor any identity 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.

In this paper, however, we show that their protocol suffers from serious flaws:

– **Service misuse.** Any token provider T_i can misuse services of S (or let someone else do that), and can implicate any entity (such as A) in such misuse.

– **Identity compromise.** For each escrowed certificate Φ that S receives, there exists a token provider T_i who in coalition with S can recover the identity escrowed in Φ.
Additionally, if S has successfully requested for a certain escrowed certificate to be uncovered, then it can discover the identity of any subsequent user of its services.

We also identify some other less significant weaknesses of the protocol in [4], and present an improved protocol which fixes these flaws. We beleive our protocol guarantees anonymity even if all but one of the token providers are corrupt.

The paper is organised as follows. In the next section, we present preliminaries, including the original protocol. Our analysis follows in section 3, and in section 4 we present our improved version of the protocol. Section 5 contains our conclusions.

2 Preliminaries

2.1 Notation

The following labeling conventions are used throughout the paper:

– S denotes an anonymous service provider.
– $T = \{T_1, T_2, \ldots T_q\}$ is a set of *identity token providers*.
– Φ_i is an identity token issued by T_{a_i}. We also write Φ_A for the identity token obtained by A by using the protocol.
– $E = \{E_1, \ldots, E_k\}$ is a set of *adjudicators*.
– A is a service user. $\widehat{A}$ denotes the receiver A of a message, when the identity of A is not known to the message sender.
– K_A is A's public key. $\{m\}_K$ is the message m deterministically encrypted with the public key K.
– $[m]_{K^-}$ is the message m signed with the private key corresponding to the public K. We assume that $[\{m\}_K]_{K^-}$, $\{[m]_{K^-}\}_K$ and m are all distinct (thus, in particular, the PKI algorithm is not simple use of RSA).

2.2 The original protocol

Marshall and Molina-Jiminez's protocol [4] consists of two parts:

– 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;
– and 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.

Signup protocol. In order for A to use the services of S, she must place her identity in escrow with the elements of T and obtain a token. She uses this token to prove to S that she has placed her identity in escrow, and S then provides the service.

The protocol works as follows. A chooses a sequence $T_{a_1}, T_{a_2}, \ldots, T_{a_p}$ of elements of T (possibly with duplications).

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 $\Phi_1 = [\, \{K_A\}_{K_{T_{a_1}}} \,]_{K_{T_{a_1}}^-}$

`ITKReq` means "identity token request". Next, A anonymises the token by getting $T_{a_2}, \ldots, T_{a_p}$ to encrypt and sign it:

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

Before signing the token, T_{a+1} verifies that it has been signed by another token provider. `ITKSig` indicates a signature request. $*$ indicates repeated application. The dashed arrow indicates that a message is sent anonymously, i.e. the receiver can not trace back the identity of the sender.

3) $A \dashrightarrow S : \{ \texttt{ServReq}, K_{\widehat{A}}, \Phi_A \}_{K_S}$

4) $S \longrightarrow \widehat{A} : \{ n, E \}_{K_{\widehat{A}}}$

5) $A \dashrightarrow S : \{ H \}_{K_S}$, where $H \subseteq E$ of cardinality n

6) $S \longrightarrow \widehat{A} : \{ \texttt{an\_id} \}_{K_{\widehat{A}}}$

$K_{\widehat{A}}$ is a new public key created by A for use with the service. Obviously, only A has the corresponding private key. In step 4, S invites A to choose a set $H \subseteq E$ of n adjudicators, who will vote on whether A's identity should be revealed in the case of a complaint. In step 6, S sends an anonymous identifier for A to use when using the services S offers. In their paper [4], the authors stipulate that S will divide the identity token into several parts and distribute them to adjudicators, using Rabin's information dispersal [7].

The authors assume that the identity token is uniquely tied to an entity, and adjudicators are trusted to provide a fair adjudication for complaints.

Complaint resolution protocol. When S receives a complaint Ψ, the following protocol is executed, without A:

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

Message 1 is sent to each $E_i \in H$.

2) $E_i \longrightarrow S : \{ [\, V_i \,]_{K_{E_i}^-} \}_{K_S}$

The vote V_i consists of a ballot (a decision by the adjudicator on the complaint Ψ, e.g yes/no) together with the complaint Ψ. If the votes are positive in the majority, S presents the tuple of signed votes V to T_{a_p}, the last token provider in the sequence chosen by A:

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

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

The last two steps are repeated several times, tracing backwards through the sequence $T_{a_1}, \ldots, T_{a_p}$ chosen by A, before finally obtaining K_A.

3 Analysis

The protocol is subject to the following serious vulnerabilities:

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 full token for A.

It is evident that such vulnerabilities are possible due to putting full trust on token providers and generating identity token Φ_A that is not tied to an anonymous key $K_{\widehat{A}}$, i.e. whoever gets hold of the token can use it with a key of his own.

In addition, the authors of the protocol do not spell out assumptions they make on the anonymous channels. Thus, one could also claim that because a message anonymously sent by A at step 1a does not include "reply instructions", e.g. a temporary public key $K_{\widehat{A}}$, *any dishonest party C that can eavesdrop on A's outgoing messages of the protocol can acquire a valid identity token Φ_A: C* intercepts/copies messages, which are to be sent anonymously by A, and, then, replays them anonymously to all T_{a_i}s in order to receive Φ_A which can be used to request services from S. Clearly, similar, but a weaker statement can be said of a dishonest entity that can eavesdrop on any of T_{a_i}s' connections.

Note that in any case, a dishonest T_{a_i} or whoever misused the services of S with Φ_A, can not be shown to have cheated.

Identity Compromise (1). *Suppose A has identified the 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 that A has submitted to S, viz. Φ_A:*

Suppose T_{a_1} is dishonest, i.e. it reveals to S identity tickets it issues. Then it takes at most n^{k-1} number of operations (`ITKSig` requests and encryptions), where n is the total number of identity token providers and k is the length of A's requests chain, for the coalition to find out Φ_A - a straightforward brute-force search.

However, if we allow the coalition to eavesdrop on messages of other token providers T_{a_i}, then the number of operations they need to perform goes down to at most $n(k-1)$. This is done as follows:

- The coalition starts noting in a set M all the messages that other token providers as well as S receive from the moment T_{a_1} sends Φ_1.
- Upon reception of Φ_is check if any is in M, else wait until one of them is.
- Repeat the above steps until Φ_A is found.

So, it is now possible to find the corresponding token for the chosen identity within polynomial time in k modulo the costs of eavesdropping.

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.

Other Weaknesses. The protocol also has the following undesirable properties/glitches:

- Any third party can find out who misused the services of an anonymous service provider S.
- A dishonest service provider S can adjust the set H to include "convenient" adjudicators, when requesting to reveal the identity of an anonymous user in the complaint resolution protocol.
- Obviously, the last message in the complaint resolution protocol needs to be authenticated.

4 Improved Protocol

In order to present the protocol concisely, we omit details about A's choice of the adjudicators $H \subseteq E$. We use just one adjudicator, which we note E.

4.1 The Protocol

It also consists of two parts - *signup* and *complaint resolution* - that have the same purpose as the previous ones, but with a different structure.

Signup.

A chooses a sequence $T_{a_1}, T_{a_2}, \ldots, T_{a_n}$ of elements of T (possibly with duplications). In contrast with the previous protocol, we distinguish two temporary keys for A. A creates a temporary service public key $K_{[A]}$ which she will use to identify herself to S. Additionally, she creates a public key $K_{\hat{A}}$ which she will use in anonymous communication with token providers, to indicate who the reply needs to be sent to. 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.

$$1)\quad A \quad \longmapsto \quad T_{a_1} \quad : \quad \{\ \texttt{InitITKReq}, K_{[A]},\ K_{\hat{A}}\ \}_{K_{T_{a_1}}}$$

$$2)\quad T_{a_1} \quad \longrightarrow\!\!| \quad A \quad : \quad \{\ \Phi_1\ \}_{K_{\hat{A}}},$$
$$\text{where } \Phi_1 = [\ \{\ \texttt{InitITKReq}, K_{[A]}\ \}_{K_{T_{a_1}}}\ ]_{K^-_{T_{a_1}}}$$

By including the service key $K_{[A]}$ in the message of step 1, we will later have this key associated with A's identity token in order avoid the Service Misuse attack, whereby anyone who acquires her token can use it to obtain a service on behalf of A from S. Note that, in contrast with the previous protocol, A has not revealed her identity to T_{a_1}.

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

$$* \begin{cases} 1a) \quad A \quad \longmapsto \quad T_{a_{i+1}} \quad : \quad \{\ \texttt{ITKReq},\ \Phi_i, \texttt{NT}_{a_{i+1}}, K_{\hat{A}}\ \}_{K_{T_{a_{i+1}}}} \\ \qquad\qquad \text{where for } i > 1 \quad \Phi_i = [\ \{\ \Phi_{i-1}, \texttt{NT}_{a_i}, K_{\hat{A}}\ \}_{K_{T_{a_i}}}]_{K^-_{T_{a_i}}} \\ \\ 2a) \quad T_{a_{i+1}} \quad \longrightarrow\!\!| \quad A \quad : \quad \{\ [\ \{\ \Phi_i, \texttt{NT}_{a_{i+1}}, K_{\hat{A}}\ \}_{K_{T_{a_{i+1}}}}\ ]_{K^-_{T_{a_{i+1}}}}\ \}_{K_{\hat{A}}} \end{cases}$$

Each time A sends a message to T_{a_i} containing $\Phi_{i-1}, \texttt{NT}_{a_i}, K_{\hat{A}}$ and receives back from it a message with $\Phi_i = [\ X\ ]_{K^-_{T_{a_i}}}$, she checks that $X = \{\ \Phi_{i-1}, \texttt{NT}_{a_i}, K_{\hat{A}}\ \}_{K_{T_{a_i}}}$, by reconstructing the encryption. By the end of the sequence of messages (1a, 2a) ($n - 2$ times), A has obtained the token Φ_{n-1}:

$$[\{\,[\{\,\ldots \\ [\{\,[\{\texttt{InitITKReq}, K_{[A]}\}_{K_{T_{a_1}}}]_{K^-_{T_{a_1}}}, \texttt{NT}_{a_2}, K_{\hat{A}}\}_{K_{T_{a_2}}}]_{K^-_{T_{a_2}}} \\ \ldots\}_{K_{T_{a_{n-2}}}}]_{K^-_{T_{a_{n-2}}}}, \texttt{NT}_{a_{n-1}}\}_{K_{T_{a_{n-1}}}}]_{K^-_{T_{a_{n-1}}}}$$

which serves as a disguise of her key $K_{[A]}$. Note the nonces $\texttt{NT}_{a_i}$ in the above steps, generated by A. They are necessary in order to preclude the Identity Compromise (1) attack (see Analysis section).

We assume that all agents receiving a signed message verify the signature. Thus, on receiving message (1a), T_{a+1} verifies that the token has been signed by another token provider before he signs it and sends it on.

Next, A reveals her identity to T_{a_n}, by signing the token Φ_{n-1}. Steps 3, 4, 3a, 4a reverse this sequence of encryptions, and at the same time they build up the identity token $\widetilde{\Phi}_{n-1}$.

$$3)\quad A \quad \longmapsto \quad T_{a_n} \quad : \quad \{\ [\ \texttt{ITKSig},\ \Phi_{n-1},\ A\ ]_{K_A^-}\ \}_{K_{T_{a_n}}}$$

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

After step 3a, before sending out a response, token provider $T_{a_{n-i}}$ checks that the key $K_{\widehat{A}}$ supplied in the `ITKReq` request matches the one embedded in Φ_{n-i} (cf. step 2a above). The same rule applies to T_{a_1} at step 5. (Both token providers also check that $\widetilde{\Phi}$ contained in the `ITKReq` request was signed by some token provider.)

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

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

$$5)\quad A \quad \longmapsto \quad T_{a_1} \quad : \quad \{\ \texttt{ITKSig}, \widetilde{\Phi}_{n-1}, \texttt{NT}'_{a_1}, K_{\widehat{A}}\ \}_{K_{T_{a_1}}}$$

$$6)\quad T_{a_1} \quad \longrightarrow\!| \quad A \quad : \quad \{\ [\ \{\ \widetilde{\Phi}_{n-1}, \texttt{NT}'_{a_1}\ \}_{K_{T_{a_1}}}, K_{[A]}\ ]_{K_{T_{a_1}}^-}\ \}_{K_{\widehat{A}}}$$

Upon reaching T_{a_1} we have the identity token for A:

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

The token $\widetilde{\Phi}_A$ associates the $K_{[A]}$ with A, and therefore can only be used by an entity which knows the private key corresponding to $K_{[A]}$.

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

A presents the token to S and initiates the service. S checks that the key $K_{[A]}$ that A provided is contained inside the token which is signed by one of the providers.

Complaint Resolution

We assume that a complaint $\Psi_{K_{[A]}}$ is uniquely associated with A's service key $K_{[A]}$. It must be verifiable by an adjudicator E and not forgeable by S. If the adjudicator agrees with the complaint he signs it and then sends it back to S.

$$1)\quad S \longrightarrow E \;:\; \{\ \mathtt{AdjReq}, \Psi_{K_{[A]}}, S\ \}_{K_E}$$

$$2)\quad E \longrightarrow S \;:\; \{\ [\Psi_{K_{[A]}}]_{K_E^-}\ \}_{K_S}$$

$$3)\quad S \longrightarrow T_{a_1} \;:\; \{\ \mathtt{Reveal}, \widetilde{\Phi}_A, \widetilde{\Psi}, S\ \}_{K_{T_{a_1}}} \quad \text{where } \widetilde{\Psi} = [\Psi_{K_{[A]}}]_{K_E^-}$$

$$4)\quad T_{a_1} \longrightarrow S \;:\; \{\ \widetilde{\Phi}_{n-1}, \mathtt{NT}'_{a_1}, \widetilde{\Psi}\ \}_{K_S}$$

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

$$*\begin{cases} 3a)\quad S \longrightarrow T_{a_{i+1}} \;:\; \{\ \mathtt{Reveal}, ((\widetilde{\Phi}_{n-i}, \mathtt{NT}_{a_i}, \mathtt{NT}'_{a_i}), \ldots, (\widetilde{\Phi}_{n-1}, \mathtt{NT}'_{a_1}), \widetilde{\Phi}_n), \widetilde{\Psi}, S\ \}_{K_{T_{a_{i+1}}}} \\ 4a)\quad T_{a_{i+1}} \longrightarrow S \;:\; \{\ \widetilde{\Phi}_{n-i-1}, \mathtt{NT}_{a_{i+1}}, \mathtt{NT}'_{a_{i+1}}, \widetilde{\Psi}\ \}_{K_S} \end{cases}$$

$$5)\quad S \longrightarrow T_{a_n} \;:\; \{\ \mathtt{Reveal},\ ((\widetilde{\Phi}_1, \mathtt{NT}_{a_{n-1}}, \mathtt{NT}'_{a_{n-1}}), \ldots, \widetilde{\Phi}_n), \widetilde{\Psi}\ \}_{K_{T_{a_n}}}$$

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

In message 3a, the tuple of $\widetilde{\Phi}_i$s serves to prevent complaint resolution messages in one session being used in another. Each T_{a_i} checks that the sequence he receives is correct, using the nonces $\mathtt{NT}'_{a_i}$, and that the last element of the sequence is the token that the complaint $\Psi_{K_{[A]}}$ is uniquely associated with.

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 nonces $\mathtt{NT}_{a_i}$, in order to make sure that Φ_{n-1} is formed from the key she was given in the service request step, viz. it is $K_{\widehat{A}}$. If there is a mismatch, she finds out which T_{a_i} cheated, and, furthermore, has evidence to prove that to any other party.

4.2 Properties of the protocol

If the token providers try to generate tokens by themselves, they can be shown to have cheated. Also the token created for A is unusable by any other entity that acquires it.

If at least one of the token providers in the sequence $T_{a_1}, T_{a_2}, \ldots, T_{a_n}$ is honest, then A's identity is not revealed without valid complaint. Thus, the protocol avoids the identity compromise attacks of section 3.

5 Conclusions

The protocol for anonymity with identity escrow in [4] is shown to have some serious flaws. We have presented an improved protocol to achieve the same aim, and in the future work we will verify the protocol using appropriate tools, such as Proverif [1] or Isabelle [6].

References

1. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In S. Schneider, editor, *14th IEEE Computer Security Foundations Workshop*, pages 82–96, Cape Breton, Nova Scotia, Canada, June 2001. IEEE Computer Society Press.
2. J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptology (CRYPTO'98)*, number 1462 in LNCS, pages 169–187. Springer Verlag, 1998.
3. F. Leighton. Failsafe key escrow systems. Technical Memo 483, MIT Laboratory for Computer Science, 1994.
4. L. Marshall and C. Molina-Jiminez. Anonymity with identity escrow. In T. Dimitrakos and F. Martinelli, editors, *Proceedings of the 1st International Workshop on Formal Aspects in Security and Trust*, pages 121–129, Istituto di Informatica e Telematica, Pisa, 2003.
5. S. Micali. Fair public-key cryptosystems. In *Advances in Cryptology (CRYPTO'92)*, number 740 in LNCS. Springer Verlag, 1993.
6. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *J. Computer Security*, 6:85–128, 1998.
7. M. O. Rabin. Efficient dispersal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.