International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:5, 2008

A Collusion-Resistant Distributed Signature Delegation Based on Anonymous Mobile Agent

Omaima Bamasak

***Abstract*—**This paper presents a novel method that allows an agent host to delegate its signing power to an anonymous mobile agent in such away that the mobile agent does not reveal any information about its host's identity and, at the same time, can be authenticated by the service host, hence, ensuring fairness of service provision. The solution introduces a verification server to verify the signature generated by the mobile agent in such a way that even if colluding with the service host, both parties will not get more information than what they already have. The solution incorporates three methods: Agent Signature Key Generation method, Agent Signature Generation method, Agent Signature Verification method. The most notable feature of the solution is that, in addition to allowing secure and anonymous signature delegation, it enables tracking of malicious mobile agents when a service host is attacked. The security properties of the proposed solution are analyzed, and the solution is compared with the most related work.

***Keywords*—**Anonymous signature delegation, collusion resistance, e-commerce fairness, mobile agent security.

I. Introduction

THE extensive connectivity of the Internet and the strong architecture of the World Wide Web (WWW) have changed the market conventions and created numerous opportunities for conducting business on the Internet (i.e. e-commerce). Inline with the growth of e-commerce, there have been interesting developments in the area of mobile agent, or software entities that can autonomously perform a given task in open, dynamic and heterogeneous environments. Integrating mobile agents into e-commerce applications (e.g. online shopping and auctioning) to automatically perform e-commerce tasks makes the Internet reaches its full potential as an electronic marketplace.

However, prior to fully enjoy the advantages brought by the mobile agents, the risks and vulnerability they may introduce are worth consideration. Various mobile agents designed by different programmers/developers can work, interact, and also attack at anytime from anywhere in the web, where the transactions can be performed instantly. This has made security an issue that must be considered in any agent-based e-commerce environment.

Manuscript received April 27, 2008. This work was supported by King Abdulaziz University, Jeddah, Saudi Arabia, under Grant 427/515.

Omaima Bamasak is with the Department of Computer Science, College of Computing and Information Technology, King Abdulaziz University, P.O. Box 48288, Jeddah, 21551, Saudi Arabia (phone: 00966-2-6400000 ext 26446; fax:00966-2-6401688; e-mail: obamasek@ kau.edu.sa).

One of the most security sensitive tasks faced by a mobile agent in performing an e-commerce transaction is to sign a digital signature on behalf of its owner (i.e. agent host) autonomously on a service host. The service host may not be trustworthy; for example, it may attempt to steal the signature key and forge signatures for its own benefit. On the other hand, service hosts, openly providing an execution environment for different kinds of mobile agents, increase the possibility that they may be attacked by malicious agents. In addition, the exposure of the identities of agents and/or agent hosts may lead to unfairness in service provision. For example, in an online auction activity, a service host may favor a particular mobile agent (if its identity is known) and grant it a higher priority in service provision over other mobile agents.

To overcome some of these security problems, a Trusted Third Party (*TTP*) based approach has been widely used in which a TTP is used to assist the execution and completion of an e-transaction (e.g. digital signature protocols) or to resolve any disputes incurred during the transaction process [1], [3] – [15]. During the execution of a digital signature protocol, the *TTP*, by facilitating a fair exchange of the signatures between the two signing parties and by preserving the evidence of the transaction, could provide a protection for both the mobile agent and the service host from attacks launched by its counterpart. As a mediator, the *TTP* may have access to the signatures, the signed document, or any related evidence. Therefore, any collusion between the *TTP* and one of the signing parties will result in undesirable consequences in which the other party will be left in a disadvantage position.

A number of researches [16], [18] have tackled the issue of fairness provision, which is defined in [16] as "the equal treatment of authenticated mobile agents by service hosts". However, the approach presented in [16] requires that the mobile agent gets a signed permission from a service host on the services it offers prior to the actual migration for executing its tasks. In addition of being considered as extra communication overhead, this results in the service host's ability to link the mobile agent's identity with its permission and thus, violating the agent anonymity. This approach also extensively use public/private keys for encryption and digital signatures. The work presented in [18] does not provide a mean to track down and penalize a misbehaving anonymous mobile agent.

To prevent collusion, and to ensure fairness, in which all the mobile agents are treated equally by the service host,

mobile agents should be anonymous during the course of a transaction execution. Therefore, how to achieve identity authentication and how to ensure that the agents behaviors are accountable, while, at the same time, preserving identity anonymity of mobile agent, are an open research issue. The scope of this paper is to address the above mentioned issues by investigating and designing effective mechanisms that provides a secure and fair mobile agent-based signature delegation environment.

The solution proposed in this paper is about splitting the duties of the *TTP*, (e.g. partial signature generation and signature verification as presented in [2]), to be undertaken by two separate entities (*TTP* and Verification Server *VS*). The separation is designed in such a way that even if the *VS* and the service host collude, they will not get more information than each party already have. The solution also incorporates blind signature scheme proposed by Chaum [17] to achieve agents' identity anonymity, and hence, facilitate the fairness property mentioned above.

The reminder of the paper is organized as follows, Section II lists the security requirements that our solution aims at satisfying. Section III presents the principles and philosophy on which our solution is based. Section IV outlines the notion used in the solution description and the assumptions on which the solution is designed. Section V gives detailed coverage of our solution. Section VI provides security analysis of our work. Finally, Section VII outlines our conclusions.

II. Security Requirements

The following lists security requirements a secure and fair mobile agent-based signature delegation solution is aimed at satisfying.

S1) Verifiability of the signature: Validity of the signature generated by the mobile agent on a document *M* can be verified using public parameters.

S2) Unforgeability of the signature: It is difficult for any other entities than the agent's host and the agent itself to generate a valid signature on the specified document.

S3) Non-repudiation of signature origin: It is difficult for an original signer (i.e. the agent host) to falsely deny that it has delegated the signing power to the agent.

S4) Non-repudiation of signature receipt: It is difficult for a signature recipient (i.e. the service host) to falsely deny that it has received the signature, if this signature is taken as the proof of a deal conducted by the mobile agent and the recipient.

S5) Collusion-resistance: it should be difficult for the *VS* and the service host, if collude together, to get any advantage over a mobile agent and the agent host.

S6) Unlinkability: Deciding whether two different valid signatures were computed by the same mobile agent is computationally hard.

S7) Anonymity: The real identity of a mobile agent should not be revealed to any party other than the agent host itself.

S8) Fairness of service provision: The service host should only process requests made from authenticated mobile agents and on the first-come-first-serve basis.

S9) Agent Host and Mobile agent Accountability: Any misbehavior by a mobile agent should be detectable and its host will be accounted for.

III. Design Principles

The design of our solution is based upon the following hypothesis, i.e. if the service provider, i.e. *TTP*/verification service and service host, can not link a request for the service to the identity of the service requestor (i.e. a mobile agent or agent host), then it would be more difficult, or less likely, for the service provider to collude with any of the service requestors to gain unfair advantages over other service requestors.

To realize the above mentioned hypothesis, a number of design principles, i.e. measures, have been taken into account in the solution design. They are listed in the following:

- **Measure 1**. The mobile agent signature key is generated in such a way that it does not reveal any information about the agent host or the mobile agent identities. It is also one-time, i.e. a key is used to generated only one signature. This supports the unlinkability of signatures property.
- **Measure 2**. The blind signature scheme proposed by Chaum [17] is used in our solution to allow for the *TTP* to blindly certify the mobile agent signature key without having knowledge neither of the key nor of the mobile agent's identity. Thus, supporting anonymity of the mobile agent. However, the service host needs to authenticate the arriving mobile agents so as to provide them with the services they request. To solve this dilemma, i.e. making the mobile agent anonymous and, at the same time, can be authenticated, the *TTP* (the party that is trusted by all other parties of the solution) certifies, i.e. signs, the agent's signature key. Thus, the service host will use this signature as a mean to authenticate the mobile agent.
- **Measure 3.** The signature generated by the mobile agent can only be verified by the verification server through the use of a commitment generated by the agent host, rather than the agent host's public key corresponding to the signature key used to generate a conventional signature. By doing so, we deliberately deprive the service host from this signature verification capability in order to achieve non-repudiation of service requests and provisions.
- **Measure 4.** A penalty system is applied on a misbehaving mobile agent and its host. After each transaction is completed, the service host assigns a feedback flag to the transaction and sends it to the *TTP*. The values given are dependent on the outcome of the transaction, i.e. signature generation process, performed by the mobile agent. For example, if the transaction

outcome is positive, the flag value will be 'Success'; if the outcome is negative due to the signature not passing the verification process, then the flag value will be 'Attack'; and if the outcome is negative due to any other reasons, then the flag value will be 'Failure'. The *TTP*, upon receiving the outcome value, updates the status of the corresponding *AH*. That is, if the *TTP* receives 'Attack' as an outcome flag, it will increment agent host's associated counter of malicious incidence. When this counter reaches a certain threshold specified by the *TTP*, i.e. five incidences, this agent host will be blacklisted and the *TTP* will refuse to provide it with any service in the future. This measure will deter the agent host from sending mobile agents to service hosts for malicious purposes.

IV. Preliminaries

In this section, we outline the notions used in the solution description and the assumptions on which the solution is designed. This is followed by outlining Chaum's blind signature scheme as it is incorporated in the generation and certification of a mobile agent's signature key to facilitate agent anonymity.

A. Notations

- $H(x)$ is a one-way collision-free hash function that takes a variable sized input (x) and produces a fixed-size output (digest). It should have the following properties: (1) for any x, it is easy to compute $H(x)$; (2) given x, it is hard to find x' ($\neq x$) such that $H(x) = H(x')$; and (3) given $H(x)$, it is hard to compute x. SHA-1 [20] is an example of such a one-way hash function.
- Sign($\{d_I, n\}$, M) denotes a signature of party I on an item M (e.g. a hash value of a document) using the RSA signature scheme [19] with a private key $\{d_I, n\}$ of I. RSA is based on two large prime numbers (p and q), which are multiplied together to get the public modulus n. Party I calculates $f(n) = (p-1)(q-1)$ and chooses e_I to be relatively prime to $f(n)$ and less than $f(n)$. Party I then determines d_I such that $d_I \times e_I = 1 \bmod f(n)$ and $d_I < f(n)$. The public key is $\{e_I, n\}$ and the private key is $\{d_I, n\}$. The signature of party I on message M with its private key is expressed as $\text{Sign}(\{d_I, n\}, M) = H(M)^{d_I} \bmod n$.
- Verify($\{e_I, n\}$, S_I, M) denotes the result of the verification of party I's RSA signature $S_I = \text{Sign}(\{d_I, n\}, M)$ on M with I's public key $\{e_I, n\}$. To verify the signature, the receiver first computes the hash value of *M' received*, $H(M')$, and then calculates $T = S_I^{e_I} \bmod n = H(M)^{e_I \times d_I} \bmod n = H(M)$. It then compares $H(M')$ with T and, if the two values are equal, the signature is considered valid, which is expresses as $\text{Verify}(\{e_I, n\}, S_I, M) = \text{true}$, otherwise, $\text{Verify}(\{e_I, n\}, S_I, M) = \text{false}$.
- $E(\{e_I, n\}, M)$ denotes an encryption of item M using the RSA encryption scheme [19] with the public key $\{e_I, n\}$ of party I.
- $D(\{d_I, n\}, M)$ denotes a decryption of item M using the RSA encryption scheme [19] with the private key $\{d_I, n\}$ of party I.
- $A \xrightarrow{E} B$: M denotes that party A sends a message M to party B via an external channel such as a telecommunication network.
- $A \xrightarrow{I} B$: M denotes that party A sends a message M to party B via an internal message passing mechanism. This case applies when both A and B resides at the same host.
- ID_I, $I \in \{AH, SH, MA, TTP, VS\}$, denotes party I's unique identity, where *AH* denotes Agent Host, *SH* service host, *MA* mobile agent, *VS* verification server.

B. Assumptions

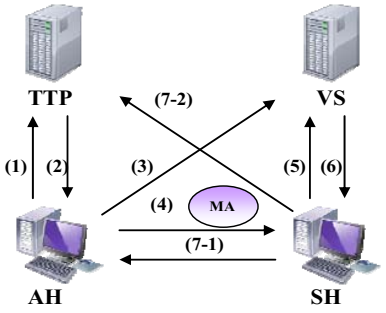
- Every party or host $I \in \{AH, SH, TTP, VS\}$ has a pair of RSA public and private keys $\{e_I, n\}$ and $\{d_I, n\}$, as defined in Section *A*. The public key $\{e_I, n\}$ is certified in the form of a digital certificate *Cert*(I) signed by a certification authority (*CA*), which is trusted by all parties.
- Parties *AH* and *SH* have each *TTP*'s and *VS*'s public key certificate *Cert(I)*. The *TTP* and *VS* also have the public key certificates of each other and of the parties participating in the solution, i.e. *AH* and *SH*. These certificates will play a role in authentication and secure communications between these parties.
- Parties *AH* and *SH* may not have mutual trust. That is, either of them may misbehave in order to gain some advantages over the other party. For example, party *SH* may try to use *MA*'s signature key to sign more that one deal for which *AH* (i.e. the user) will be held responsible. *TTP* and *VS* are introduced in the solution to assist *MA*'s signature verification and to store transaction evidence for dispute resolution. It is assumed that *TTP* and *VS* will not misbehave or collude with each other or with any other party.
- *Req* represents the service required by *AH* which *MA* is delegated to perform on service host *SH*. For example, *Req* typically includes service name, validity period and transaction-specific information (e.g. good type, price, etc).
- Party *AH* (Agent Host) delegates his mobile agent *MA* to perform some tasks and sign a document *M* on a service (remote) host *SH*. Typically, *M* is the service (e.g. offer) presented by *SH* that conforms to *Req*.
- *SH* is assumed to provide mechanism to protect the mobile agents it hosts from being eavesdropped on their contents and execution flows by other agents hosted also by *SH*. *SH* can use existing solutions, e.g. tamper-resistant hardware [21] and time limited blackbox security [22], to provide such mechanisms.

C. Blind Signatures for Untraceable Payments

Chaum proposed a blind signature scheme [17] for untraceable payments based on the RSA public-key cryptographic system [19]. The signature scheme allows a person to get a message signed by another party without revealing any information about the message to the signing party. The scheme works as follows. Assume that Alice has a message M on which she wishes to have Bob's signature, and Alice does not want Bob to learn anything about M during the signing process. Let $\{(e, n)\ (d, n)\}$ be Bob's public and private keys, respectively. The scheme defines the following steps to generate a blind signature on M:

1) Alice generates a random number r such that gcd $(r, n) = 1$, produces a message digest $H(M)$ for message M using a hash function $H()$, and sends $x = r^e \times H(M)\ (mod\ n)$ to Bob. The value of $H(M)$ is "blinded" by the random value r, hence Bob can derive no useful information from it.
2) Bob signs x using his private key and return the signed value $t = x^d\ (mod\ n)$ to Alice.
3) Since $x^d = (r^e \times H(M))^d = r \times H(M)^d\ (mod\ n)$, Alice can obtain Bob's signature S on M by "unblinding" the value t by computing $S = t\ r^{-1}\ (mod\ n)$.

V. Proposed Solution



(1) MA signature key request.
(2) MA signature key delivery
(3) Verification-aiding items
(4) MA dispatch
(5) MA signature verification request
(6) MA signature verification response.
(7-1) MA return
(7-2) Transaction outcome

Fig. 1 Our solution overview

Our collusion-resistant distributed mobile agent-based signature delegation solution is built upon three novel cryptographic methods: the Agent Signature Key Generation Method, the Agent Signature Generation Method, and the Agent Signature Verification Method. As shown in Fig.1, the solution is initiated by the user (represented by an *AH*) who executes, in cooperation with the *TTP*, the Agent Signature Key Generation Method, to generate a certified mobile agent signature key S_{MA} using agent's anonymous ID ($Anony\text{-}ID_{MA}$) and Chaum's blind signature scheme. Using the Agent Signature Generation Method and the signature key, S_{MA}, the mobile agent generates a signature on an offer made by a service host, *SH*. Once the signature is generated, (only) the Verification Server, *SV,* can verify the correctness of the signature by using the Agent Signature Verification Method. These three methods are described in detail below.

A. Agent Signature Key Generation Method

The Agent Signature Key Generation Method is executed by the *AH* with the assistance of the *TTP* to generate its signature key S_{MA}. In addition, as mentioned in Section III (Design principles), we have devised an idea of using a commitment generated by the signature key generator *AH* (instead of using its public key) for the agent signature verification, The following gives the details as how the signature key S_{MA} and the commitment $Comm_{MA}$ are generated by *AH*. The commitment $Comm_{MA}$ will be used by the signature verifier (*VS*) to verify the signature to assure that $Sign_{MA}(Doc)$ has indeed been generated by using the correct signature key S_{MA}, the signature is generated only once using the signature key S_{MA}, and that the signed document meets the user's requirements *Req*. For an *AH* to generate an agent signature key and the corresponding commitment, it performs the following calculations.

1) The agent host, *AH,* first generates an anonymous identity ($Anony\text{-}ID_{MA}$) for the mobile agent *MA*.
2) *AH* then generates a random number r, uses r to blind the hash value of the agent's anonymous identity and sends it to the *TTP* after being encrypted with *TTP*'s public key (as Chaum's blind signature algorithm). That is,
$$Z = H(Anony - ID_{MA}) \times r^{e_{TTP}}$$
3) *TTP*, upon the recipient of the request from *AH*, blindly signs *Z,* and sends it back to *AH.*
$$T = Z^{d_{TTP}} = H(Anony - ID_{MA})^{d_{TTP}} \times r^{e_{TTP} \times d_{TTP}} = H(Anony - ID_{MA})^{d_{TTP}} \times r$$
Here, *TTP* has signed *Z* without knowing its contents.
4) *AH* unblinds *T* to reveal *MA*'s signature key S_{MA}:
$$S_{MA} = T / r = (H(Anony - ID_{MA})^{d_{TTP}} \times r) / r = H(Anony - ID_{MA})^{d_{TTP}}$$
S_{MA} is the signature key to be used by *MA* to sign documents on *SH* on behalf of *AH*. It can be seen that S_{MA} represents *TTP*'s signature on *MA*'s anonymous identity.
5) *AH* also constructs a commitment $Comm_{MA}$ containing four items: hashed *MA*'s anonymous ID $H(Anony\text{-}ID_{MA})$, *Req*, and the key's validity period (lifetime), signed with *AH*'s private key (i.e. $Sign(\{d_{AH}, n\}, H(Anony\text{-}ID_{MA}), Req, Lifetime) = H(H(Anony - ID_{MA}), \mathrm{Re}q, lifetime)^{d_{AH}} \bmod n$). When *AH* dispatches *MA*, it sends $Comm_{MA}$ to *VS* for signature verification purpose.
6) For accountability purpose, *AH* generates a *Bond* for each *MA* and pass it to the *MA* before being dispatched to perform its task. The *Bond* is the hash value of the concatenation of a random number and *Anony-IDMA*, that is, $H(rand||Anony\text{-}IDMA)$ and maps to one and only one mobile agent ID (*IDMA*). This *Bond* is sent to the *TTP* to be recorded with other related information (i.e.

AH's identity) in its database to act as the *MA*'s pseudonym and to be used to track down and penalize a misbehaving *MA*.

B. Agent Signature Generation Method

MA, residing at *SH*, generates a signature on document *M* using the method described below:

1) *MA* computes signature on *M* using S_{MA}, i.e. $D = S_{MA}^{H(M)} \bmod n$, where $\text{Sign}_{\text{MA}}(M) = (M, D)$ is *MA*'s signature on *M*.
2) *MA* also submits its *Bond* value to the *SH*. If *MA* attacks an *SH*, the *SH* will send an 'Attack' flag together with the *Bond* value as an outcome of the transaction to the *TTP*. The *TTP* compares the *Bond* with the value it has received earlier from *AH* and stored in its database to fetch the identity of the *AH* who sent this *MA* and penalize accordingly

C. Agent Signature Verification Method

The signature is verified by a Verification Server (*VS*) using the method described below.

1) When *SH* wants to verify the signature $\text{Sign}_{\text{MA}}(M)$ generated by *MA*, it sends *M* signed with its private key (i.e. $\text{Sign}(\{d_{SH}, n\}, M) = H(M)^{d_{SH}} \bmod n$) together with *MA*'s signature on *M* (i.e. $\text{Sign}_{\text{MA}}(M) = (M, D)$) to *VS*.
2) Upon the receipt of the these items, i.e. ($\text{Sign}(\{d_{SH}, n\}, M) || \text{Sign}_{\text{MA}}(M)$), where || denotes concatenation, *VS* performs the following computations:
   - **a.** $T = D^{e_{TTP}} \bmod n = S_{MA}^{e_{TTP} \times d_{TTP} \times H(M)} \bmod n$
     $= H(Anony - ID_{MA})^{H(M)} \bmod n$
   - **b.** Computes the hash of *M* received in $\text{Sign}_{\text{MA}}(M)$, (i.e. $H(M)$), uses this freshly computed hash value together with the hash value of the *MA*'s anonymous ID received earlier from *AH* in the commitment $Comm_{MA}$ (i.e. $H(Anony\text{-}ID_{MA})$) to compute $Y = H(Anony\text{-}ID_{MA})^{\text{H}(M)} \bmod n$.
   - **c.** Check if $T = Y$; if positive, then the signature $\text{Sign}_{\text{MA}}(M)$ is valid.

VI. Security Analysis

In this section, we analyze the security properties of the proposed solution demonstrating that it satisfies all the security requirements stated in Section II.

(S1) Verifiability of the signature:

VS is able to verify the validity of mobile agent's signature $\text{Sign}_{\text{MA}}(M)$ using the information included in commitment $Comm_{MA}$, which is generated and sent by *AH*. It is worth noting that *VS* is able to verify $\text{Sign}_{\text{MA}}(M)$ without accessing *MA*'s anonymous identity $Anony\_ID_{MA}$. This feature supports the anonymity property of *MA* and, in turn, the fairness of service provision (next).

(S2) Unforgeability of the signature:

Since the mobile agent signature key S_{MA} is derived from the agent anonymous ID ($Anony\text{-}ID_{MA}$) that is only known to *AH* and is a one-time, it would be difficult for another party to forge the signature key without knowledge of $Anony\_ID_{MA}$. However, there are two scenarios where the signature might be forged:

- *VS* receives the hash of $Anony\_ID_{MA}$, i.e. $H(Anony\text{-}ID_{MA})$ in $Comm_{MA}$ sent by *AH*. *VS* might try to use this hash to compute S_{MA}. For doing so, *VS* needs also to know the private key of the *TTP*, i.e. e_{TTP}, which only *TTP* has knowledge of. Therefore, it is difficult for *VS* to re-generate S_{MA} and forge *MA*'s signature.
- In *TTP*: the *TTP* certifies the signature key by calculating $T = H(Anony - ID_{MA})^{d_{TTP}} \times r$. For the *TTP* to obtain the signature key $S_{MA} = H(Anony - ID_{MA})^{d_{TTP}}$ from *T*, it has to know the blinding factor *r*, which only *AH* has knowledge of. It is also difficult for the *TTP* to obtain S_{MA} from *T* due to the difficulty of factoring large primes, i.e. factoring *T* to get S_{MA} and *r*.

(S3) Non-repudiation of signature origin:

The Agent Signature Verification Method performed by the *VS* ensures that the signature $\text{Sign}_{\text{MA}}(M)$ is generated by using a signature key that is generated by *AH*. This is because *VS* uses $H(Anony\text{-}ID_{MA})$ received in $Comm_{MA}$, which is signed by *AH*. Therefore, *AH* cannot deny the fact that it has generated the signature key.

(S5) Collusion-resistance:

In order to check if the solution satisfies this requirement, we first have to look at the data items both *VS* and *SH* have or know, shown in Table I.

TABLE I
DATA ITEMS OWNED/KNOWN BY *VS* AND *SH*

VS	SH
$H(Anony\text{-}ID_{MA})$, *Req*, *Lifetime*, $\text{Sign}_{\text{MA}}(M)$, $\text{Sign}(\{d_{SH}, n\}, M)$	*MA*'s contents: (S_{MA}, *itinerary*, *Req*)
$Comm_{MA} = \text{Sign}(\{d_{AH}, n\}, H(Anony\text{-}ID_{MA}), Req)$	*TTP*'s public key
$Sign(\{d_{SH}, n\}, M), Sign_{MA}(M)$	$\text{Sign}(\{d_{SH}, n\}, M) = H(M)^{d_{SH}} \bmod n$
TTP's and *SH*'s public keys	

From Table I, it can be seen that the piece of data owned by *VS* and of an interest to *SH* is $H(Anony\text{-}ID_{MA})$. *SH* might use this information to guess *MA*'s or *AH*'s identities, hence, violate the anonymity properties and, in turn, the fairness of service provision property. This attack is thwarted in our solution as follows. As mobile agent's anonymous identity ($Anony\text{-}ID_{MA}$) is randomly generated and hashed, it does not reveal any information about either *MA*'s or *AH*'s identities. Furthermore, $Anony\text{-}ID_{MA}$ is freshly generated for each transaction, i.e. one-time only, which means *SH* will find it difficult to use this information to link together two different transactions performed by the same agent in a hope of guessing *MA*'s identity. Therefore, it is difficult for the *VS* and the *SH*, if collude together, to get any advantage over the *MA* and the *AH*.

S6) Unlinkability:

By looking at the contents of the signature ($\mathrm{Sign}_{MA}(M) = (H(Anony-ID_{MA})^{d_{TTP}\times H(M)} \bmod n,\ M)$, generated by *MA* using the signature key S_{MA}, it can be seen that the signature does not have any information that can be used to link it with other signature generated by the same mobile agent *MA* (in a different transaction) or the same agent host *AH*. This is because the mobile agent signature key S_{MA} is computed using a freshly generated mobile agent anonymous identity (*Anony-ID*$_{MA}$). This key is one-time only, hence, is used to generate one signature on the document for one transaction. This unlinkability feature supports the anonymity property to be discussed next.

S7) Anonymity:

In addition to the anonymity provided by the unlinkability of different signatures to the same *MA* or *AH*, anonymity is also provided through the contents of the mobile agent itself. That is, the data the mobile agent carries while roaming the network, i.e. (*Itinerary*, S_{MA}, *Lifetime*) neither reveals *MA*'s nor *AH*'s identities. Therefore, it will be difficult for *SH*s visited by the agent to obtain any information that leads to the agent's source. One may argue that if the agent does not carry any information regarding it's, or its source's, identity, then how can *SH*s authenticate this agent? The agent is authenticated through the *TTP*'s signature on *MA*'s anonymous identity (S_{MA}). When *SH* verifies this signature, the agent is authenticated by the trust *SH* hold for the *TTP*. This trust stems from the fact that the *TTP* only certifies agents of whom their owners are trustworthy.

S8) Fairness of service provision:

Due to the unlinkability and anonymity properties, the *SH* will not have a mean to distinguish between the authenticated mobile agents as to which to provide its service first. Therefore, the *SH* will serve all the agents on the first-come-first-served basis, hence, fairness of service provision is satisfied.

S9) Agent Host Accountability:

As the mobile agent in our solution is anonymous, i.e. untraceable, one may question about the ability of *SH*s, if attacked by malicious agents, to get hold of them. In our solution, the *TTP*, in collaboration with the *SH*, is able to detect and penalize, i.e. blacklist, an agent host when its agent acts maliciously by using the *Bond* generated by *AH* and sent to both *TTP* and *SH* as described in Section V.

VII. Conclusion

This paper has addressed the mobile agent-based anonymous signature delegation issue by presenting a novel solution that incorporates three methods as its building blocks, namely, Agent Signature Key Generation method, Agent Signature Generation method, Agent Signature Verification method. The protocol enjoys the following features. Firstly, it makes use of Chaum's blind signature scheme to allow for a trusted third party (*TTP*) to blindly certify the mobile agent signature key for mobile agent's anonymity. The mobile agent, while residing at the service host, does not reveal any information about its host's identity, which deprive the service host from favoring a mobile agent on the others, hence, ensuring fairness of service provision. Secondly, the protocol presents a mechanism to track down malicious mobile agents and penalize their hosts accordingly.

Our solution can be applied to many applications, e.g., e-/m-commerce, grid computing, and ubiquitous computing due to the properties of mobile agents.

The future work will include formal verifications of the security properties of the solution using a verification tool, i.e. the Alternating Temporal Logic and the model checker MOCHA [23], [24]. An implementation of the solution will be also conducted using Aglets mobile agent framework [25] and Java libraries and the implemented solution's performance will be evaluated.

References

[1] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures", IEEE Journal on Selected Areas in Communication., vol. 18, pp. 591-610, April 2000.

[2] O. Bamasak, " Delegating Signing Power to Mobile Agents: Algorithms and Protocol Design", PhD Thesis, School of Computer Science, the University of Manchester, January 2006.

[3] F. Bao, R.H. Deng, and W. Mao, "Efficient and practical fair exchange protocols with off-line TTP", in Proc. IEEE Symposium on Security and Privacy, Oakland, CA, May 1998, pp. 77-85.

[4] M. Blum, "How to exchange (secret) keys", ACM Trans. Computer Systems, Vol. 1, no.2, pp. 175-193, 1983.

[5] C. Boyd and E. Foo, "Off-line fair payment protocols using convertible signature", Advances in Cryptology – in Proc. Asiacrypt' 98, LNCS 1514, Springer-Verlag, 1998, pp. 271-285.

[6] L. Chen, "Efficient fair exchange with verifiable confirmation of signatures", Advances in Cryptology – in Proc. Asiacrypt' 98, LNCS 1514, Springer-Verlag, 1998, pp. 286-299.

[7] R.H.Deng, L. Gong, A. A. Lazar, and W. Wang, "Practical protocol for certified electronic mail", Journal of Network and System Management, vol. 4, no. 3, pp.279-297, 1996.

[8] S. Even, O. Golreich, and A. Lempel, "A randomized protocol for signing contracts", Communications of the ACM, vol. 28, no. 6, pp. 637-647, 1985.

[9] M.K. Franklin and M.K. Reiter, "Verifiable signature sharing", Advances in Cryptology – Proc. Eurocrypt' 95, LNCS 921, 1995, pp. 50-63.

[10] J. A. Garay, M. Jakobsson, and P. MacKenzie, "Abuse-free optimistic contract signing", Advances in Cryptology – Proc. Crypto' 99, LNCS 1666, Springer-Verlag, 1999, pp. 449 – 466.

[11] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications", Advances in Cryptology – Proc. Eurocrypt' 96, LNCS 1070, Springer-Verlag, 1996, pp. 143 – 154.

[12] T. Okamoto and K. Ohta, "How to simultaneously exchange secrets by general assumptions", in Proc. the 2nd ACM Conference on Computer and Communications Security, 1994, pp. 184-192.

[13] C. Wang and C. Yin, "Practical Implementations of a Non-disclosure Fair Contract Signing Protocol", IEICE Trans. on Fundamentals of Electronics, Communications and Computer Science, vol. e89-a, no. 1, pp. 297-309, 2006.

[14] J. Zhou and D. Gollmann, "A fair non-repudiation protocol", in Proc. 1996 IEEE Symposium on Security and Privacy, Oakland, CA, 1996, pp. 55-61.

[15] J. Zhou and D. Gollmann, "An efficient non-repudiation protocol", in Proc. 1997 IEEE Computer Security Foundations Workshop (CSFW 10), 1997, pp. 126 – 132.

[16] M. Lin, C. Chang, Y. Chen, "A fair and secure mobile agent environment based on blind signature and proxy host", Computers & Security, vol. 23, pp. 199-212, Elsevier, 2004.

[17] D. Chaum, “Blind signatures for untraceable payments”, in Proc. CRYPTO’82, Plenum Press, Berlin, 1983, pp. 199-203.
[18] J. Kim, G. Kim, Y. Eom, “Design of the Mobile Agent Anonymity Framework in Ubiquitous Computing Environments”, IEICE Trans. on Information and Systems, Vol. E89-D, No. 12, pp. 2990-2993, December 2006.
[19] RL. Rivest, A. Shamir, LM. Adleman, “A method for obtaining digital signatures and public key cryptosystems”. Communication of ACM, Vol. 21, No. 2, pp. 120-126.
[20] National Institute of Standard and Technology (NIST), “Secure Hash Standard”, Federal Information Processing Standards Publication 180-1.
[21] U. Wilhelm, “Cryptographically Protected Objects”, Technical report, 1997, Ecole Polytechnique Federale de Lausanne, Switzerland.
[22] F. Hohl, “Time Limited Blackbox Security: Protecting Mobile Agents from malicious Hosts”, In Mobile Agents and Security, Lecture Notes in Computer Science, Vol. 1419, 1998, Springer-Verlag, pp. 92-113.
[23] S. Kremer and J. Raskin, "A game-based verification of non-repudiation and fair exchange protocols", in Proc. 12th International Conference on Concurrency Theory (CONCUR 2001), Lecture Notes in Computer Science, Vol. 2154, Springer-Verlag, Berlin, Germany, 2001, pp. 551-566.
[24] S. Kremer and J. Raskin, "Game Analysis of abuse-free contract signing", in Proc. 15th IEEE Computer Security Foundations Workshop, IEEE Computer Society Press, 2002, pp. 206-220.
[25] Aglets Mobile Agent Platform, http://www.trl.ibm.co.jp/aglets