A New Approach for Computation Result Protection in the Mobile Agent Paradigm

Suphithat Songsiri
Dept. of Communication Systems, FernUniversität Hagen
Universität str.11, D-58084 Hagen, Germany
suphithat.songsiri@fernuni-hagen.de

Abstract

One of the primary security challenges of the mobile agent paradigm is that of protecting the result of computation carried out by a mobile agent against an attack by a malicious host. There are various proposals that appeared in the literature. Beside their benefits, a well-known vulnerability of their technique is the collusion attack. The collusion attack mainly considered in this paper is the two colluders truncation attack, which could be engendered by the leakage of a one time private key. This paper demonstrates the prevention of the two colluders truncation attack, the detection of other forms of collusion attacks, and the identification of the malicious host. The proposed protocol incorporates and extends the notion of publicly verified chained signature [2] by using a list of route information and a trusted third party to generate a one time public/private key pair.

1. Introduction

A mobile agent is a program that can migrate from host to host in a heterogeneous network. Due to its benefit over the traditional client-sever paradigm, many applications make use of its mobility. This paper considers the scenario of Free Roaming Mobile Agent, which is a mobile agent that travels in the unfixed itinerary to collect the intermediate data on each visited host. Each collected data can either be the result of some computation done by the mobile agent, based on some local input, or simply some data which has been given by the visited host without any processing by the mobile agent. We particularly concentrate in protecting the integrity of partial results collected by the agent. There are many methods that have been devised to protect the integrity of the data carried by the agent including Partial Result Authentication Code [1], Chained Signature [2], Set Authentication Code [3], Append Only Container [10] and Improved Forward Integrity Protocol [4]. In [1] Yee proposed the Partial Result Authentication Code (PRAC) to ensure the integrity of the collected result. This method provides the agent with a set of secret keys used to calculate MAC (Message Authentication Code) upon the result of each host, using a one-way function to produce the key associated with the current host from an initial secret key given by the originator. PRAC comprises of the result and its MAC. The agent erases the secret key associated with the current host before migration. Yee defines forward integrity which implies that the first visited malicious host cannot modify or forge any PRACs of already visited hosts. Karjoth et al. [2] published a family of protocols that aims to protect the integrity and confidentiality of data collected by a free roaming agent. In their paper, they extended Yee's protocols and defined a set of security properties that has a higher degree of security. They extended public verifiable forward integrity from Yee [1] in the following manner "Anyone can verify the offer o_i by checking whether the chain is valid at encapsulated offer O_i". In other words, every visited host is able to verify the integrity of the encapsulated offer by using one time public/private key pair generated by the previous host. This property is very useful since not only the originator but also the visited host on the agent's itinerary can detect the tampering.

The proposed protocol will demonstrate the prevention of two colluders truncation attack and the identification of the malicious host. The objective of this protocol is to ensure the confidentiality, authenticity, integrity, publicly verification and non repudiability on the collected data by using the trusted third party to issue the one time public/private key pair and forcing the agent to carry the protected list of visited host. The structure of this paper is organized as follows: Section 2 introduces the notations and defines the security properties. Section 3 presents the possible attacks and problem

Table 1. Notations

Π	An agent's code.	$Z_{S_i S_{i+1}}$	The shared secret computed by the entities
TTP	Trusted third party	$K_{i,i+1}$	Session key calculated using key derivation function.
Ψ_i	Protected list of already visited host at S_i.	G	A subgroup of $\mathbb{Z}_p^*$
S_0	ID of the originator.	p	A large prime.
S_i	ID of server *i*, $1 \le i \le n$	q	A prime with $q \mid p-1$.
o_0	Dummy offer of originator.	g	A generator of G.
o_i	An offer (a partial result) from S_i.	OST_i	Offline status of TTP generated by S_i
O_i	An encapsulated offer	$MAC_k(m)$	Message Authentication Code generated with key K.
$O_0,...,O_n$	The chain of encapsulated offers from S_0 to S_n.	$SIG_{\upsilon_i}(m)$	Signature of S_i on a message *m*.
r_i	A nonce generated by S_i.	$ENC_{\upsilon_i}(m)$	Message *m* encrypted with the encryption key associated with S_i.
T_{S_i}	Timestamp chosen by S_i.	t_i, t_{i+1}	Ephemeral public key: $t_i = g^{\alpha_{S_i}}$, $t_{i+1} = g^{\alpha_{S_{i+1}}}$
H(m)	Hash function of message *m*.	α_i, α_{i+1}	Random integers, same size as the order of G, chosen by S_i and S_{i+1}.
$(\upsilon_i, \overline{\upsilon_i})$	A public/private key pair of server S_i	$ACK_{i+1}^{T,F}$	Acknowledgement generated by S_{i+1} after checking TTP' offline status.
$(y_i, \overline{y_i})$	A one-time key pair to be used by S_i. key pair is generated by S_{i-1}.	C_{T1}, C_{T2}	The cost according to the protocol T1,T2
$(\mu_i, \overline{\mu_i})$	A one-time key pair to be used by S_{i+1}, S_i. The key pair is generated by TTP	$S_0 \rightarrow S_1 : m$	S_0 sending a message m to S_1.
$(\sigma_i, \overline{\sigma_i})$	A one-time key pair to be used by S_{i+1} and S_i. When TTP is off-line.		

statements. Section 4 demonstrates the proposed protocol. Section 5 describes security and cost analysis. Section 6 comprises the summary of this protocol.

2. Notations and Security Requirements

The notations used in this paper are summarised in table 1. We assume that every host in the mobile agent environment knows the public key of the originator (υ_0) and the TTP (υ_{TTP}). There is no further public key infrastructure in this situation. Given a signature, anyone can extract *m,* if one possesses the associated public key. A chain of encapsulated offers is an ordered sequence of encapsulated offers such that each entry of the chain depends on the previous and/or next host. This dependency is specified by a *chain relation.*

2.1 Security Requirements

In this section, we list some of the basic security requirements behind the design of our scheme. We extend the set of security properties defined in [2] as follows:

- Data confidentiality*:* Only S_0 can extract the offer o_i.
- Non-repudiability*:* S_i cannot deny submitting o_i after S_0 receives o_i.
- Forward Privacy*:* Only S_0 can extract the visited host's identity.
- Strong forward integrity*:* at S_m, where $v < m$, none of O_v can be modified.
- Publicly Verifiable Forward Integrity*:* Anyone can verify the validity of the offer o_i by checking chain at O_i.
- Insertion resilience*:* No offer can be inserted at i unless explicitly allowed i.e. one host can insert only one offer.
- Truncation resilience*:* the chain can only be truncated at S_i if S_i colludes with the attacker.
- Malicious host identification: S_0 can identify the malicious host after the invalidity of the chain of encapsulated offers has been reported.

3. Attacks and Problem Statements

Assume that after visiting *m* undetermined hosts where $m \le n$, a mobile agent is seized by a malicious host. This host, possibly the host S_{m+1}, obtains a chain of encapsulated offers $O_0,\ldots,O_m$, Some host excluding S_m may collude with the malicious host to attack

the chain of encapsulated offers. Collusive hosts perform the following actions:

- Deletion: The malicious host deletes previously encapsulated offers generated by its predecessors.
- Modification: The malicious host alters the collected offer.
- Insertion: The malicious host performs an illegal insertion of an offer into the chain of encapsulated offers.

Please note that, in this paper both the malicious hosts and the collusive hosts/partners are able to perform deletion, truncation, modification, as well as revelation of secret information e.g. one time public/private key.

However, as pointed out by Roth in [7, 9], the protocols [1], [2] lack in mechanism to bind the dynamic data of an agent to its static data (i.e. its code and initial parameters). The absence of binding leads to the interleaving attack presented in his paper. Cheng and Wei [3] addressed the well known threat of the *two colluder truncation attack* in which a host with an agent in hand colludes with a previously visited host to discard all encapsulated offers between the two hosts. Recently, in [4] Yao et al. considered the collusion attack. They implied that the security of the KAG protocol [2] relies on the assumption that the predecessor does not leak the secret (the one time public/private key) used by its successor. This attack takes place when, for instance, a host S_{m+1} sends a copy of the one time private key of its successor S_{m+2} to its collusive partner S_k where $m+2<k$, then when the mobile agent arrives at host S_k, S_k truncates all of the encapsulated offers after $m+2^{th}$ We deduce the problem from the previous works [1], [2], [3], [4] and [10], [12] that the vulnerability of all mentioned protocols concerning the collusion attack is the leakage of the one time private key. Generally, we can categorize the technique of key fabrication into four methods namely:

Pre-creation by the originator: In [1] the originator appends the list of the keys used by each host to the mobile agent.

Using key seed: Park et al [12] proposed the algorithm OKGS to ensure the data integrity by using DES to encrypt the result generated by each host. The idea of this method is that the originator S_0 produces the key seed C_{k0} for its successor S_1. S_1 will then incorporate its secret information and the acquired key seed to produce a one time key used by DES for data encryption. S_1 will in turn produce a key seed for its successor based on its one time key. Their assumption is that the secret information from each host (which is encrypted by using the originator's public key) is appended to the mobile agent. Then, nobody except the originator can generate the key to decrypt all the ciphers. This technique is unfortunately still vulnerable to the collusion attack.

Key fabrication by predecessor: Karjoth et al. [2] use this technique to achieve the security property "Publicly verifiable" mentioned in section 2. This technique too is not unexposed to the collusion attack.

Split knowledge key generation: Yao et al. [4] suggested the idea of the cooperation between the predecessor and the successor to produce a one time key used by the successor. This technique is mainly based on the idea from [2] by composing the one time private key generated by its predecessor and its long-term private key. The authors mentioned that this method cannot efficiently defend against the truncation attack. It has been concluded that in order to prevent the two colluders truncation attack and to detect other forms of collusion attacks on the chain of the encapsulated offers, it is essential to have an efficient and secure algorithm to establish the one time public/private key pair for each visited host.

4. The Protocol

The goal of this paper is to design an effective protocol, which prevents the two colluders truncation attack [3] as well as detects other forms of collusion attacks. The security of the process used to generate the computation result at each visited host will not be discussed in this paper. This is based on the assumption that each host can be trusted to produce its own computed data. Assume that S_i has the mobile agent in hand. After having completed its computation for the mobile agent, S_i has two tasks to perform, namely: to sign the computation result and send the mobile agent to S_{i+1}. In order to complete its tasks, S_i interacts with the TTP. All hosts trust the TTP to execute the protocol faithfully and not to engage in any other activity that will deliberately compromise their security. Moreover, the TTP is trusted to generate a new one time public/private key pair for signing and verifying an offer and maintain the list of keys which have been issued for all hosts in an itinerary. The procedure is briefly described as follows: TTP issues the one time public/private key pair and maintains the key list. S_i obtains the one time private key $\overline{\mu_i}$ for signing its offer under the constraint that only S_{i+1} will be its successor. Afterward, S_{i+1} receives the one time public key μ_i for the verification of the validity of S_i's signature on its offer. We separate our protocol into two situations: TTP is always online (T1) and TTP is offline for some period of time (T2).

4.1 Protocol T1

Protocol T1 is usable only under the assumption that TTP is always online to issue the one time public/private key pair for each host. Each host in the mobile agent's itinerary knows the public key of the originator and the TTP. Both originator and TTP know each other's public keys in advanced. TTP, however, has no knowledge about each visited host's public key. The protocol commences with S_0 contacting TTP to perform mutual authentication as well as to obtain a one time private key for signing its offer. We call the latter key transportation. As defined by Boyd in [13], key transportation is a protocol in which one of the users in the protocol generates the key and this key is then transferred to all protocol users

4.1.1 Mutual Authentification

Mutual Authentication serves as an indication of mobile agent dispatching described as follows:

$$S_0 \rightarrow TTP : ENC_{\upsilon_{TTP}}(r_0, \upsilon_0, S_0) \quad (1.1)$$

$$TTP \rightarrow S_0 : ENC_{\upsilon_0}(SiG_{\upsilon_{TTP}}(r_0, r_{TTP}, S_o, TTP)) \quad (1.2)$$

$$S_0 \rightarrow TTP : ENC_{\upsilon_{TTP}}(SIG_{\upsilon_0}(r_{TTP}, r_o, TTP)) \quad (1.3)$$

The mutual authentication protocol is adapted from the ISO/IEC 9798-3 three pass mutual authentication [14]. From equations (1.1) to (1.3) the protocol assures TTP that S_0 is active, and that S_0 is aware of TTP as its peer entity.

4.1.2 Key Transportation Protocol and Key List

$$S_o \rightarrow TTP : ENC_{\upsilon_{TTP}}(S_0, S_1, T_{S_0}) \quad (2.1)$$

$$TTP \rightarrow S_0 : ENC_{\upsilon_0}(SIG_{\upsilon_{TTP}}(S_0, S_1, \overline{\mu_0}, T_{S_0}, T_{TTP_{0,1}})) \quad (2.2)$$

$$S_o \rightarrow TTP : ENC_{\upsilon_{TTP}}(SIG_{\mu_0}(S_0, S_1, T_{S_0}, T_{TTP_{0,1}})) \quad (2.3)$$

$T_{TTP_{0,1}}$: The time of one time private key issuing for S_0 when S_1 is its successor generated by TTP

Equation (2.1) is the constraint that only S_1 is the next visited host. Equation (2.2) refers to the key transportation, which guarantees the freshness of a key. Equation (2.3) is the key confirmation where the key $\overline{\mu_0}$ is used to generate a signature. TTP thus knows that S_0 successfully received the key. Consequently, equation (2.4) is initiated by TTP after receiving (2.1) for authentication purpose. The equations (2.5) through (2.7) are the combination of key transportation and key confirmation. Once (2.1) to (2.3) have been done, the originator is ready to sign its offer and dispatch the mobile agent to S_1. Obviously S_1 has an associated public key to check the validity of O_0.

$$TTP \rightarrow S_1 : SIG_{\upsilon_{TTP}}(r_{TTP}, TTP) \quad (2.4)$$

$$S_1 \rightarrow TTP : ENC_{\upsilon_{TTP}}(SIG_{\upsilon_1}(S_1, TTP, r_1, r_{TTP}), \upsilon_1) \quad (2.5)$$

$$TTP \rightarrow S_1 : ENC_{\upsilon_1}(SIG_{\upsilon_{TTP}}(S_1, TTP, r_1, T_{TTP_{1,0}}, \mu_0, S_0)) \quad (2.6)$$

$$S_1 \rightarrow TTP : ENC_{\upsilon_{TTP}}(ENC_{\mu_0}(S_1, TTP, r_1, T_{TTP_{1,0}}, S_0)) \quad (2.7)$$

$T_{TTP_{1,0}}$: The time of one time verification public key issuing for S_1 when S_0 is its predecessor.

Subsequently, TTP generates the key list as follows:

$$KeyList = \left\langle \begin{matrix} \text{Signer}, \text{Next host}, \text{Time of issue}, \text{Key pairs} \\ S_0, \upsilon_o \quad S_1, \upsilon_1 \quad T_{TTP_{1,0}}, T_{TTP_{0,1}} \quad (\mu_0, \overline{\mu_0}) \end{matrix} \right\rangle \quad (2.8)$$

TTP updates or sends the key list periodically to the originator for verification of the chain of signatures in case tampering has been detected.

4.1.3 Offer Encapsulation, List of Visited host and Agent Transmission

$$O_0 = SIG_{\mu_o}(ENC_{\upsilon_0}(o_0, r_0), h_o), h_0 = H(r_o, S_1) \quad (3.1)$$

$$\Psi_o = ENC_{\upsilon_0}(SiG_{\mu_0}(S_0, S_1)) \quad (3.2)$$

$$S_0 \rightarrow S_1 : SiG_{\upsilon_0}(\Pi, T_{S_0}), O_0, \Psi_0 \quad (3.3)$$

Equation (3.1) illustrates that, after S_0 obtains the key issued by TTP, it signs an offer with the given key. This part of the protocol is a modification of the KAG protocol (P4), where the one time public key generated by S_0 has been removed. Then, S_0 produces a list of already visited hosts in form of an encrypted signature containing S_0 and S_1, encrypted. Finally, S_0 sends the mobile agent to its successor.

4.1.4 Agent Migration Protocol at S_i ($1 \le i \le n$):

After receiving the mobile agent, the chain of encapsulated offers and the list of already visited hosts from S_{i-1}, S_i verifies the immutable part to ensure that the code belongs to the originator by checking the signature of the originator on the code. Subsequently if the code has not been altered, it inspects the encapsulated offer proceeding in a decreasing order, otherwise it reports to the originator. S_i uses μ_{i-1} to recover h_{i-1}, μ_{i-2} from O_{i-1}. S_i examines recursively until the first encapsulated offer. If the invalidity in chain signature is identified, S_i sends the mobile agent to the originator immediately. If the agent verification is successful, S_i retrieves the one time private key from TTP and

proceeds stepwise according to the previously defined equations. The generalization of how to produce offer encapsulation, list of visited host and agent transmission is given as follows:

$$O_i = SIG_{\mu_i}(ENG_{\upsilon_0}(o_i, r_i), h_i, S_{i-1}, \mu_{i-1}), h_i = H(O_{i-1}, S_{i+1}) \quad (4.1)$$

$$\Psi_i = ENC_{\upsilon_0}(SiG_{\mu_i}(S_{i-1}, S_i, S_{i+1}), \Psi_{i-1}) \quad (4.2)$$

$$S_i \rightarrow S_{i+1} : SiG_{\upsilon_0}(\Pi, T_{S_0}), \{O_0, O_1, ..., O_i\}, \Psi_i \quad (4.3)$$

4.2 Protocol T2

Protocol T2 is focused mainly on dealing with a situation where TTP is inaccessible for a certain period of time when a host requires the one time private key for signing an offer. Protocol T2 is designed such that a host will be granted a temporary authority to generate the one time key pair on its own. Temporary authority will only be granted to S_i when the absence of TTP has been confirmed by S_{i+1}. Protocol T2 is described as follows: Protocol initialization at S_i: S_i was unable to reach TTP, the following occurs:

4.2.1 Offline Status of TTP Generation (OST_i):

S_i produces the unreachable-status log file, which documents the number of attempts made in contacting TTP by using "ping" command.

4.2.2 Key Agreement and Mutual Authentication.

Assume that there is no key sharing between two hosts and they have no knowledge of each other's public keys. The key agreement protocol is utilized to establish a session key between the two cooperating hosts. The session key is used to send offline status of TTP securely to S_{i+1}. There are some protocols which have been devised in the literature [15], [17], [18] concerning key agreement protocol. Key agreement protocols can be divided into two types. The symmetric protocol [15] describes the two entities a-priori possess common secret information, while the asymmetric protocol describes the two entities sharing only public information that has been authenticated. In our scheme, we employ the asymmetric protocol based on Diffie-Hellman key establishment [16], followed by an exchange of authentication information. Our protocol can be divided into two steps with the following procedures:

- Step 1: equations (5.1) through (5.4), indicate the exchange of algorithms and parameters to establish a session key.
- Step 2: equations (6.1) and (6.2) indicate entity authentication and acknowledgment swapping.

In step 1, S_i offers a list of Diffie-Hellman key exchange parameters that S_i uses for ephemeral key fabrication. Once the ephemeral keys have been exchanged, both entities can derive a shared secret and form a temporary derived key by using key derivation function. MAC is employed as the key derivation function.

$$S_i \rightarrow S_{i+1} : S_i, LIST, r_i, t_i \quad (5.1)$$

$$S_{i+1} \rightarrow S_i : S_{i+1}, t_{i+1}, r_{i+1} \quad (5.2)$$

$$(LIST = (p, q, g, G, K_{i,i+1})), K_{i,i+1} = MAC_{r_i, r_{i+1}}(Z_{S_i S_{i+1}}) \quad (5.3)$$

$$Z_{S_i S_{i+1}} \text{ at } S_i = t_{i+1}^{\alpha_i}, S_{i+1} = t_i^{\alpha_{i+1}} \quad (5.4)$$

Now, we consider the assurance Step 1 provides both entities. From S_i's point of view, it shares a secret (5.4), which is a result of ephemeral key exchange, known only to it and its peer entity who may or may not be S_{i+1}.

$$S_i \rightarrow S_{i+1} :$$
$$SIG_{v_i}(MAC_{K_{i,i+1}}(t_i, t_{i+1}, r_i, r_{i+1}, List, S_i), OST_i), ENC_{K_{i,i+1}}(v_i) \quad (6.1)$$

$$S_{i+1} \rightarrow S_i :$$
$$SIG_{v_{i+1}}(MAC_{K_{i,i+1}}(t_i, t_{i+1}, r_i, r_{i+1}, List, S_{i+1}), ACK_{i+1}^T), ENC_{K_{i,i+1}}(v_{i+1}) \quad (6.2)$$

Subsequently, S_i and S_{i+1} generate MAC over the parameters that they used to produce the session key. S_{i+1} has signed the particular parameters mentioned in Step 1, of which S_i has just created for this individual section. Encrypting its associated public key for the verification of its signature with $K_{i,i+1}, S_{i+1}$ shows S_i that it was the creator of t_{i+1}. This assures S_i that the entity engaged in the key exchange was, indeed S_{i+1}. S_{i+1} obtains an analogous set of assurance from S_i. After verifying the received signature, S_{i+1} proves the genuineness of OST_i by attempting to contact TTP. On one hand, if TTP is indeed inactive, then ACK_{i+1}^T is generated. S_i will then receive an acknowledgment (6.2). This allows S_i to generate the one time key pair. S_{i+1} as a witness to the key generation done by its predecessor is expected to send an acknowledgement about the key generation to TTP once it gets the mobile agent in hand. On the other hand, if TTP is available, S_{i+1} sends ACK_{i+1}^F to inform S_i to try to contact TTP again to retrieve the authorized one time private key. Therefore, the protocol T1 will be applied. There is, however, a possible situation where S_i cannot reach TTP due to reasons like communication channel failure, but S_{i+1} can reach TTP. S_{i+1} will send acknowledgement to S_i. Upon receiving ACK_{i+1}^F S_i retransmits TTP's offline status to S_{i+1}. S_{i+1} will now ask TTP to contact S_i immediately. If TTP is also unable to reach S_i, then

TTP has to provide evidence of communication channel failure to S_{i+1}. Once the evidence arrived, S_{i+1} must cooperate with S_i to execute the appropriate protocol (i.e. T2).

4.2.3 Offer Encapsulation and Agent Dispatching

S_i performs a one time public/private key pair generation. The generated private key pair is used to sign its offer as shown in equation (7.1). Equations (7.2) and (7.3) represent list of visited host and agent dispatching.

$$O_i = SIG_{\overline{\sigma_i}}(ENC_{\upsilon_0}(o_i, r_i, ACK^T_{S_{i+1}}, OST_{S_i}), h_i, S_{i-1}, \mu_{i-1}), \\ ENC_{K_{i,i+1}}(ENC_{\nu_{i+1}}(\sigma_i)) \quad (7.1)$$

$$\Psi_i = ENC_{\upsilon_0}(SiG_{\overline{\sigma_i}}(S_{i-1}, S_i, S_{i+1}), \Psi_{i-1}) \quad (7.2)$$

$$S_i \rightarrow S_{i+1} : SiG_{\overline{\upsilon_0}}(\Pi, T_{S_0}), \{O_0, O_1, ..., O_i\}, \Psi_i \quad (7.3)$$

ACK^T_{i+1} and offline statue in encrypted offer provide assurance to the originator that at the period of time when TTP is inactive, S_i as a claimant and its verifier, S_{i+1} , cooperatively produced the evidence and proof of TTP's inactiveness. ACK^T_{i+1} serves as a confirmation that not only S_i experienced the absence of TTP but also S_{i+1}. After having received the key generation acknowledgement, the TTP instantaneously updates the key list incorporating the information obtained during its inactivity. S_{i+1} must include the offline status generated by the predecessor as well as its own acknowledgment into its offer no matter which protocol, i.e. T1 or T2, it employs. Therefore, the protocol T1 is applied when there is a presence of the TTP otherwise T2.

5. Security and Cost Analysis

In this section, the protocols are analysed to ascertain whether they satisfy the set of security properties described in section 2. The protocols T1 and T2 differ in generation of one time public/private key as well as the parameters contained in the encapsulated offer, however the method of signing an offer and protected list of visited hosts are similar. Therefore, the protocols can be similarly analysed.

Data confidentiality: each host encrypts the offer with the originator's public key so that only the originator can retrieve the protected offer.

Non-repudiability: In T1, each host submits its offer signed with the key given by the TTP, thus making it impossible for S_i to deny the offer it has signed. In T2, each host submits its offer signed with the "self-generated" key, thus prohibiting S_i from denying having signed the offer.

Forward Privacy: is not fulfilled. The reason being every host must append its predecessor's identity to its signature to be used when the host verifies the hash value in every encapsulated offer.

Strong forward integrity: The hash function and digital signature used enforce this property.

Publicly Verifiable Forward Integrity: Any host can retrieve the verification public key.

Insertion resilience: Insertion of any illegal offer anywhere in the chain relation is prevented by the hash function.

Truncation resilience: is fulfilled by protocol T1 because all of the hosts use only the key issued by TTP. We can track which hosts append their offers to the chain of encapsulated offers. Protocol T2 only satisfies this property provided the TTP's offline time is not too long.

Malicious host identification: the protocols T1 and T2 are treated separately. Protocol T1's possible attacks and malicious host identification, for example, are described as follows: *Two colluder truncation attack*: This attack can be eliminated due to the fact that only TTP is entitled to generate the one time public/private key which will be used to sign an offer. Even if a collusive host sends its key to its malicious partner, the key will not be usable to sign an offer. *Signing and verifying an offer without obtaining the authorized keys*: Two or more collusive hosts did not contact TTP to obtain the authorized keys. This attack can be detected when the honest host receives a mobile agent without having been previously contacted by the TTP. It will send the mobile agent back to the originator. The identity of the malicious host(s) can be unmasked by comparing the list of visited hosts and the key list.

Protocol T2's possible attacks and malicious host identification are described as follows: *Two colluders truncation attack*: The protocol T2 detects this attack by checking the list of visited hosts generated by the hosts between the collusive partners. *Spurious one time public/private key deployment*: T2 is capable of detecting a malicious host in case it appended a bogus one time public key on its offer. Since T2 requires all hosts to update the list of generated keys to TTP, the originator only needs to check the list of generated keys in order to identify the malicious host.

5.1 Cost Analysis

In this subsection we investigate the cost introduced by our protocols T1 and T2 under the network utilization aspect. Suppose the cost of sending a single session message is U_n. The total cost of each protocol at each visited host is the multiplication of the cost per message with the total

number of messages (N_i) composed during authentication, key transportation, key confirmation and mobile agent transmission. We derive the cost C_{T1} according to $C_{T1} = N_1 * U_N$, where N_1 = 11. Identically we deduce the cost C_{T2}, where N_2 = 5. Comparing our protocols to the existing protocols [2, 3, 4, 8, 10, 11], ours induce many message transmissions, which results in cost ineffectiveness (i.e. from network utilization point of view). Conversely to these mentioned protocols, our protocols can prevent the problem of two colluders truncation attack and identify the malicious host.

6. Conclusion and Future Work

As shown in section 5, T1 is not only capable of fulfilling most of the stated security requirements, but it is also competent to detect malicious activities as well as identify the malicious hosts. T2 provides a more adaptable scheme as the operation can still take place in the absence of TTP. Apart from that, the protocol is also able to detect the two colluders truncation attack by checking the list of visited hosts generated by the hosts. However, the identification of the real malicious hosts cannot be efficiently capacitated by the protocol. The robustness of the protocols T1 and T2 relies mainly on the authentication protocol, key establishment and key agreement schemes. The two protocols introduce many message passing activities, which inevitably exposes the system to other types of attacks like message replay attack and impersonation attack. Our future research is aimed to avoid the large number of message transmission activities between the hosts and to simulate the cost imposed by our protocol.

7. References

[1] B. S. Yee,"*A Sanctuary for Mobile Agents. Secure Internet Programming*",LNCS, Vol. 1603. Springer-Verlag, Berlin Heidelberg (1999) pp.261–273

[2] G. Karjoth, N. Asokan, C. Gülcü,: "Protecting the Computation Results of Free-Roaming Agents". Proceedings of the 2nd International Workshop on Mobile Agents. *LNCS,* Vol. 1477. Springer-Verlag, Berlin Heidelberg New York (1998) 195–207

[3] J. S. Cheng, L. Wei,"*Defenses against the Truncation of Computation Results of Free-Roaming Agents"*, ICICS 2002, Singapore. *LNCS*, Vol. 2513. Springer-Verlag,Berlin Heidelberg (2002) pp.1–12

[4] M. Yao, E. Foo, K. Peng, and E. Dawson, "*An Improved Forward Integrity Protocol for Mobile Agents*." LNCS, Vol. 2908. Springer-Verlag, Berlin Heidelberg (2004) ,pp.272–285

[5] P. Maggi., R. Sisto, "*A Configurable Mobile Agent Data Protection Protocol",* Proceedings of the 2nd International Conference on Autonomous Agents and Multi agent Systems (AAMAS'03), Melbourne, Australia. ACM Press, New York, USA (2003) 851–858

[6] A..Menezes., P. C. van Oorschot, "*Handbook of Applied Cryptography*". CRC Press Inc. (1996)

[7] V. Roth," *Programming Satan's agents"*, In: Fischer, K., Hutter, D.. (eds.): Proceedings of 1st International Workshop on Secure Mobile Multi-Agent Systems (SEMAS 2001). Electronic Notes in Theoretical Computer Science, Vol. 63. Elsevier Science Publishers (2002).

[8] V. Roth, "*On the Robustness of some Cryptographic Protocols for Mobile Agent Protection*", Proceedings Mobile Agents 2001. LNCS, Vol. 2240. Springer-Verlag, Berlin Heidelberg (2001),pp. 1–14

[9] V. Roth, "*Empowering Mobile Software Agents*" Proceedings 6th IEEE Mobile Agents Conference.,LNCS Vol. 2535. Springer-Verlag, (2002),pp. 47–63

[10] N. M. Karnik and A. R. Tripathi, "*Security in the Ajanta Mobile Agent System*". Technical Report TR-5-99, University of Minnesota, Minneapolis, MN 55455, U. S. A., May 1999.

[11]. A. Corradi, R. Montanari, and C. Stefanelli, "*Mobile agents Protection in the Internet Environment*", In The 23rd Annual International Computer Software and Applications Conference (COMPSAC '99), pp. 80–85

[12] J. Y. Park, D. Lee and H. H. Lee, "*Data Protection in Mobile Agents; one-time key based approach*", IEEE ISADS 01, pp.411-418

[13] C. Boyd, A. Mathria, "*Protocols for Authentication and Key Establishment*" Springer-Verlag ISBN 3-540-43107-1

[14]. ISO. " I*nformation technology-Security techniques-Entity authentication mechanisms-part 3: Entity authentication Using a Public key Algorithm ISO/IEC 9798-3".*,2nd Edition , 1998. International standard.

[15]. B.W Simon, A. Menezes, "*Authenticated Diffie-Hellman Key Agreement Protocols*", SAC'98 LNCS 1556, pp. 339-361. 1999

[16]. W. Diffie,Martin E. Hellman, "*New directions in cryptography"*, IEEE transaction on Information Theory, November 1976.

[17]. W. Diffie, Paul C. van Oorschot, M. J. Wiener, "*Authentication and Authenticated Key exchange. Designs, Codes and Cryptography"* March 1992

[18]. H. Orman, "*The OAKLEY Key Determination Protocol*",The Internet Society, November 1998, RFC 2412