Using Co-operating agents for partial results protection in mobile agent paradigm

George W. Helmy, Magdy A. Ahmed and Mohamed N. El-Derini
Computer and Systems Eng, Dept., Faculty of Eng., Alexandria University, Alexandria, Egypt
Emails: (GeorgeWagdy@yahoo.com, magdy@alex.edu.eg, elderini@ieee.org)

E-commerce can be implemented using mobile agent paradigm which is considered as an extension to distributed system approach, it supports releasing software though network nodes involving its private information to perform its goals. In spite of mobile agent paradigm importance, it is exposed to numerous attacks, which should be resolved; a well-known vulnerability of protecting mobile agent information is the truncation attack. This paper proposes a new protocol (Co-T1) that incorporates co-operating agents approach and protocol T1, whereas Co-T1 protocol depends on a monitoring-agent for generating one-time key pairs, logging movement hops, and satisfying truncation resilience and attack resilience. While a bid-agent is used for migrating through an open network to collect offers according to application requirements in addition to fulfilling remaining security requirements.

التجارة الالكترونية يمكن تفعيلها باستخدام نموذج الوكيل المتنقل، حيث أنه يُعتبر امتداداً لمجال الانظمة الموزعَة ، كما أنه يعمل على إطلاق برنامج خلال الشبكات لتنفيذ أهداف معينة. بالرغم من أهمية هذا النموذج إلا أنه مُعرض للعديد من الاختراقات التى تتطلب التغلب عليها، من امثلتها المهاجمة الحذفية. هذا البحث يقدم بروتوكول جديد (Co-T1) حيث أنه دمج لبروتوكول الوكلاء المتعاوني و بروتوكول تى، حيث أن البروتوكول (Co-T1) يعتمد على وكيل مُراقِب يعمل على إنتاج أزواج من مفاتيح الشفرة ، تسجيل التحركات فى كل خطوات البروتوكول ، تحقيق خاصتى اكتشاف المهاجمة الحذفية و استرجاع الوكيل المتضرر(مُحصّل العروض) . بينما وكيل العروض يتنقل خلال الشبكات مجمعا العروض طبقا لمتطلبات الاستخدام ، كما يعمل على تحقيق باقى متطلبات حماية العروض المحمولة لديه

Keywords: Mobile agent, Cryptographic protocol, Truncation resilience and attack resilience

I. Introduction

A mobile agent is a software program that migrates to visit network nodes for collecting specific data according to owner restrictions and its application requirements. It performs processing during its itinerary to achieve most suitable requirements of its owner.

There are open issues related to the protection of mobile agent against malicious hosts. Protecting a mobile agent is a hard task because of the mobility property, where agents migrate through open networks under full control of hosting-environments; this implies the need for securing agent components, whereas mobile agent consists of several components: agent code, control flow state and data state.

With respect to agent's data-state, researchers have concentrated on detecting attacks that are against mobile agents rather than preventing them. So confidentiality and integrity are considered as important requirements for mobile agent paradigm. A well-known vulnerability of free-roaming mobile agent integrity mechanisms is truncation attack, which is caused by cooperation of multiple malicious hosts to truncate mobile agents' information.

This paper focuses on satisfying data-state security requirements especially truncation resilience, overcoming Satan's agent [1] and issuing attack resilience property.

The rest of this paper is organized as follows: Section II shows security requirements for mobile agent paradigm. Section III reviews the previous work. Section IV describes Co-T1 protocol to resolve truncation resilience and introduces attack resilience issue. Section V expresses security analysis of Co-T1 protocol. Section VI analyzes Co-T1 protocol cost and comparative studies. Finally this paper will formalize our conclusion and future work.

2. Security requirements

Agent data-state security requirements have three categories: primitive security requirements, integrity requirements and extra security requirements.

Suppose a mobile agent migrates from originator host S_0, it will collect a chain of encapsulated offers $O_1,\ldots,O_n$ from different hosts $S_1,\ldots,S_n$ that are selected dynamically when the agent roams through an open network.

2.1. Primitive requirements

- *Data confidentiality:* only the agent's originator and data-generating host can recognize encapsulated data.
- *Data authenticity:* offers are associated only to the authorized hosts, e.g. O_i has information ensures that its generating host identity is S_i.
- *Data non-repudiability:* host cannot repudiate its offer once it has been received by originator, whereas non-repudiability implies data authenticity.
- *Origin confidentiality (Forward privacy):* no host can extract identities of offers' creators.

2.2. Integrity requirements

- *Weak forward integrity:* no host can modify the partial results that are involved by previous visited hosts during agent's itinerary.
- *Strong forward integrity:* no host can modify the partial results (in case of collusion or revisiting) that are included by visited hosts during agent's itinerary.
- *Insertion resilience:* no unauthorized host can insert a fake offer without detection.
- *Truncation resilience:* any offer that is attacked to be deleted is detectable.
- *Detecting stemming attack* [2]: is to detect a simultaneous attack of two colluders truncation attack and the insertion of one or more fake offers in place of the truncated results. This attack includes Satan's agents attack [1], where malicious host freezes bid-agent and regenerates new agent throwing it through bid-agent's itinerary to recollect fake offers signed by tricked-hosts.

2.3. Extra security requirements

- *Publicly verifiable forward integrity:* each host can verify that no modifications take place for previous offers.
- *Publicly verifiable data integrity:* each host can verify that no modification, insertion or deletion takes place on encapsulated offers, whereas publicly verifiable data integrity implies publicly verifiable forward integrity.
- *Authorized re-assign offers flexibility:* each host has the ability to modify or re-assign its offer without need to save additional data.
- *Malicious identification:* trusted entity can identify malicious hosts that perform undesired modifications.
- *Attack resilience:* it is similar to fault tolerance, where it is the ability to re-transmit corrupted or attacked mobile agent from check point rather than its originator to continue its life cycle achieving its goal.

3. Previous Work

In this section, different proposals are reviewed, which are achieved towards facing impedances of mobile agent paradigm, especially in mobile agent data-state security.

There are some efforts that are considered as the core technicality of mobile agent data-state security, such as Partial Result Authentication Code (PRAC) [3], Hash Chaining (HC) [4], and Set Authentication Code (SAC) [5].

PRAC protocol [1] provides mobile agent with a set of secret keys used to calculate MAC (Message Authentication Code) using a one-way hash function to produce a key associated with the current host from an initial secret key given by the originator.

HC protocols [4] are considered as extensions to Yee's protocol [3], where they use a chain relation and generation of keys for each host in addition to that every encapsulated offer depends on binding of a nonce with an offer and binding previous offer with next host identity to achieve chain relation through encapsulated offers.

SAC protocol [5] provides authorized re-assign offers flexibility where a host can reassign or modify its offer without need for saving additional data, this is developed by $\Gamma()$ function which is used to achieve offer integrity proof.

There are other proposals, which depend mainly on previous mentioned protocols, such as Modified SAC [6], [2, 7, 8 and 9].

MSAC protocol [5] is considered as a modification to SAC protocol, where it modifies the way of exchanging signing-keys, whereas each host generates its key entirely rather than communicates with th originator.

References [2, 7, 8 and 9] provide other methods trying to satisfy truncation resilience.

Approach T1 [7] provides a good comment about key fabrication categorization.

4. Proposed protocol

The proposed protocol (Co-T1) incorporates and extends co-operating agents protocol [10] and approach T1, whereas it uses a monitoring-agent for generating one-time key pairs and logging movement hops, while a bid-agent is used for migrating through network to collect offers. Co-T1 protocol has the ability to fulfill data-state security requirements especially truncation resilience, overcome Satan's agent [1] and issue attack resilience property.

4.1. Assumptions

There are some basic assumptions that should be demonstrated, these are expressed in the following points:

- Home host is not malicious.
- PKI (Public Key Infrastructure) is supported.
- Public key of certified authority is known to every network node.
- Any host can recognize sender's identity from its public key [9].
- Each visiting host should assign an offer whether it is meaningful or dummy.
- For every authorized reassigning offer, it should consider modified offer as a new one with a new key pair.
- There is a symmetric key that is known for all TSPs, where it is used in the confidential channel of monitoring-agent migration.

Table 1
Model notations

Π_{Agent}	An agent's code.
TTP/TSP	Trusted third party/ Trusted service provider
A_m	Monitoring-agent.
A_b	Bid-agent.
S_0	ID of the originator.
S_i	ID of server i, $1 \le i \le n$
O_0	Dummy offer of originator.
o_i	An offer (a partial result) that is generated by Si.
O_i	An encapsulated offer.
$O_0,\dots,O_i$	The chain of encapsulated offers from S_0 to S_i.
r_i	A nonce that is generated by S_i.
tn_{si}	Temporary nonce that is generated by S_i.
h_i	Hash function of (O_{i-1}, μ_{i+1})
AI_i	Additional information can be used for satisfying attack resilience.
$NHCI_{i+1}$	Next Host Cooperation Information that is created for negotiation of TSP_{i+1} and S_{i+1}.
$\sum_{k=1 \text{ to } i} (m)$	Collection of $m_1, m_2, \dots, m_i$.
$A \rightarrow B : m$	A sends message m to B.
$A_{b/m}$[content]	(bid/monitoring) Agent includes content
U_i	Unbroken chained-offers length at iteration i.

Table 2
Cryptographic notations

$(\upsilon_i, \underline{\upsilon_i})$	A public/private key pair of server S_i.
$(\mu_i, \underline{\mu_i})$	A one-time public/private key pair to be used by S_{i-1} and S_i, where key pair is generated by A_m.
$Enc_{\upsilon i}(m)$	A message m encrypted with the public key υ_i of S_i.
$Sig_{\underline{\upsilon i}}(m)$	A signature of S_i on message m with its private key $\underline{\upsilon_i}$.
$Ver(O_i, \mu_i)$	A verification function which verifies O_i using μ_i.
$H(m)$	A one-way, collision-free hash function applied on m.
{m}	Message m is sent through confidential channel using a symmetric key that is known for all TSPs.

4.2. Notations

This sub-section describes some notations concerning the mobile agent paradigm, these notations are illustrated in table 1 and table 2.

4.3. Description of Co-T1 protocol

This sub-section describes Co-T1 protocol including initialization step, first visiting host, and generalizes it for host S_i. Firstly, home host generates bid-agent (A_b) and monitoring-agent (A_m), each of which is directed to different destination and has its private information for coordinating with each other.

Bid-agent (A_b) migrates through hosts, which have bids that should be collected according to agent's owner restrictions, while monitoring-agent (A_m) migrates through TSPs for logging bid-agent's itinerary. Bid-agent decides next migration according to environment data, private information, and data collected during its itinerary, while monitoring-agent decides its migration to the nearest TSP according to next migration of bid-agent.

Bid-agent begins with home host S_0, collects offers through S_i's, coordinates with monitoring-agent and returns to home host S_{n+1}, while monitoring-agent migrates from S_0, goes through selected TSPs and ends at home host.

Initialization - At home host:

$S_0 \rightarrow CA : [\ Sig_{\underline{\upsilon 0}}(S_1, TSP_1, tn_{S0})]$.
$CA \rightarrow S_0 : [\ Sig_{\underline{CA}}(\upsilon_1, \upsilon_{TSP1}, tn_{S0})]$.
S_0 : generates r_1, $\underline{\mu_1}$, μ_1.
$S_0 : h_0 = H(\Pi_{Ab}, \mu_1)$.
$S_0 : O_0 = Sig_{\underline{\upsilon 0}}\ [h_0, \mu_1]$.
$S_0 \rightarrow S_1 : A_b\ [O_0\ ,\ Sig_{\underline{\upsilon 0}}\ (Enc_{\upsilon 1}\ (r_1), TSP_1, \upsilon_{TSP1})]$.
$S_0 \rightarrow TSP_1 : A_m\ [Sig_{\underline{\upsilon 0}}\ (Enc_{\upsilon TSP1}(r_1\ ,\ \underline{\mu_1}\ , H(\Pi_{Am})), \upsilon_1, S_1, \mu_1\ )]$

The originator S_0 specifies the first iteration (S_1,TSP_1), correspondingly, it communicates with Certified Authority (CA) to get S_1 and TSP_1 public keys, where this communication needs only two messages rather than three [7], whereas CA does not care for ensuring host availability or key arrival confirmation. S_0 creates dummy offer and releases the agents to their respective destinations.

At Host 1:

$TSP_1 \rightarrow CA : [\ Sig_{\underline{TSP1}}(S_0, tn1_{TSP1})]$.
$CA \rightarrow TSP_1 : [\ Sig_{\underline{CA}}(\upsilon_0, tn1_{TSP1})]$.
$S_1 \rightarrow CA : [\ Sig_{\underline{\upsilon 1}}(S_0, tn1_{S1})]$.
$CA \rightarrow S_1 : [\ Sig_{\underline{CA}}(\upsilon_0, tn1_{S1})]$.
$S_1 : Ver(O_0, \upsilon_0)$, extracts (μ_1, h_0).
$S_1 \rightarrow TSP_1 : Sig_{\underline{\upsilon 1}}\ [Enc_{TSP1}(r_1\ ,\ S_2, U_1)]$.
$TSP_1 \rightarrow CA : [\ Sig_{\underline{\upsilon TSP1}}(S_2\ ,\ TSP_2, tn2_{TSP1})]$.
$CA \rightarrow TSP_1 : [\ Sig_{\underline{CA}}(\upsilon_2, \upsilon_{TSP2}, tn2_{TSP1})]$.
TSP_1 : generates r_2, $\underline{\mu_2}$, μ_2.
$TSP_1 : NHCI_2 = Enc_{\upsilon 2}(r_2\ ,\ S_2, TSP_2)$.
$TSP_1 \rightarrow S_0 : Sig_{\underline{\upsilon TSP1}}[\ Enc_{\upsilon 0}(\ TSP_2, r_2, \upsilon_2, S_2, \mu_2, \underline{\mu_2}), \upsilon_{TSP2}]$.
$TSP_1 \rightarrow S_1 : Sig_{\underline{\upsilon TSP1}}\ [Enc_{\upsilon 1}(r_1\ ,\ \underline{\mu_1}), NHCI_2, \mu_2]$.
$S_1 \rightarrow TSP_1 : Sig_{\underline{\mu_1}}\ [\ Enc_{\upsilon TSP1}(r_1, AI_1\ )]$.
$S_1 : h_1 = H(O_0, \mu_2)$.
$S_1 : O_1 = Sig_{\underline{\mu_1}}\ [Enc_{\upsilon 0}(o_1, r_1), h_1, \mu_2]$.
$TSP_1 \rightarrow TSP_2 : A_m[\{S_0, \upsilon_0, H(\Pi_{Am})\}, \sum_{k=1 to 2} \{(r_k, \upsilon_k, S_k, \mu_k, \underline{\mu_k})\}]$.
$S_1 \rightarrow S_2 : A_b\ [(O_0, O_1), Sig_{\underline{\mu 1}}(NHCI_2)]$.

Each hosting-environment verifies the immutable-part of visiting agent. A_b sends next migration information to its A_m that replies with a key pair, A_b creates its encapsulated offer, and migrates to next host with a chain of encapsulated offers and $NHCI_2$ after encrypting it with $\underline{\mu_1}$.

At Host i (iteration i):

$S_i \rightarrow CA : [\ Sig_{\underline{\upsilon i}}(S_0, tn_{si})]$.
$CA \rightarrow S_i : [\ Sig_{\underline{CA}}(\upsilon_0, tn_{si})]$.
$S_i : Ver(O_0, \upsilon_0), Ver(O_1, \mu_1), \ldots, Ver(O_{i-1}, \mu_{i-1})$.
$S_i \rightarrow CA : [\ Sig_{\underline{\upsilon i}}(TSP_i, tn2_{si})]$.
$CA \rightarrow S_i : [\ Sig_{\underline{CA}}(\upsilon_{TSPi}, tn2_{si})]$.
$S_i \rightarrow TSP_i : Sig_{\underline{\upsilon i}}\ [\ Enc_{TSPi}(r_i\ ,\ S_{i+1}, U_i)]$.
$TSP_i \rightarrow CA : [\ Sig_{\underline{TSPi}}(S_{i+1}, TSP_{i+1}, tn_{TSPi})]$.
$CA \rightarrow TSP_i : [\ Sig_{\underline{CA}}(\upsilon_{i+1}, \upsilon_{TSPi+1}, tn_{TSPi})]$.
TSP_i : generates r_{i+1}, $\underline{\mu_{i+1}}$, μ_{i+1}.
$TSP_i : NHCI_{i+1} = Enc_{\upsilon i+1}(r_{i+1}\ ,\ S_{i+1}, TSP_{i+1})$.
$TSP_i \rightarrow S_0 : Sig_{\underline{\upsilon TSPi}}[Enc_{\upsilon 0}(TSP_{i+1}, r_{i+1}, \upsilon_{i+1}, S_{i+1}, \mu_{i+1}, \underline{\mu_{i+1}}), \upsilon_{TSPi+1}]$.
$TSP_i \rightarrow S_i : Sig_{\underline{TSPi}}[Enc_{\upsilon i}(r_i, \underline{\mu_i}), NHCI_{i+1}, \mu_i, \mu_{i+1}]$.
$S_i \rightarrow TSP_i : Sig_{\underline{\mu_i}}\ [\ Enc_{TSPi}(r_i, AI_i\ )\ ]$.
$S_i : h_i = H(O_{i-1}, \mu_{i+1})$.
$S_i : O_i = Sig_{\underline{\mu_i}}\ [Enc_{\upsilon 0}(o_i, r_i), h_i, \mu_{i+1}]$.
$TSP_i \rightarrow TSP_{i+1} : A_m[\{S_0, \upsilon_0, H(\Pi_{Am})\}, \sum_{k=1 to\ i+1} \{(r_k, \upsilon_k, S_k, \mu_k, \underline{\mu_k})\}]$
$S_i \rightarrow S_{i+1} : A_b\ [(\sum_{k=0\ to\ i} O_k), Sig_{\underline{\mu i}}(NHCI_{i+1})]$.

The verification processes at iteration i are performed by co-operating agents, where A_m verifies Π_{Am} and (U_i==i), while A_b performs two sub-processes, first is verifying Π_{Ab} by the existing one in h_0, second is verifying chained-offers integrity, where it compares $H(O_{j-1}, \mu_{j+1})$ with h_j which is existed in O_j. This sub-process is achieved by decrypting O_0 to extract μ_1, where μ_1 is used to decrypt O_1 and extract μ_2 and so on. As integrity chain verification calculates U_i.

Iteration i is repeated from host S_2 to S_n till the two agents return to home host S_{n+1}, where the monitoring-agent will generate next one-time key pair rather than S_0.

At last iteration n, A_m migrates to home host carrying only required information such as $H(\Pi_{Am})$ rather than all iterations data which are gradually sent to home host through key list messages. Received required-information should be decrypted by home host for verifying Π_{Am}.

5. Security analysis

Co-T1 protocol uses one-time private key $\underline{\mu}_i$ as an authenticity tool for the assigned offer, while generated nonce r_i is used as an indicator for A_m and A_b to negotiate. On the other hand, it is useful to hash agent's code for avoiding Satan's agents [1]. These considerations enable most security properties to be satisfied, where Co-T1 protocol should be investigated by security requirements that are presented and discussed in section II.

Data confidentiality: each offer is encrypted by the originator's public key υ_0; therefore S_0 can only decrypt it. During key transportation, each $\underline{\mu}_i$ is encrypted by υ_i, so no one else S_i and A_m know $\underline{\mu}_i$, this is considered as key confidentiality. Similarly, each r_i is encrypted by receiver's public key or home host's public key. Confidential channel between TSPs also enables data confidentiality.

Non-repudiability: A_b sends key confirmation message for ensuring key arrival. Each offer is signed by $\underline{\mu}_i$; therefore S_i cannot deny its offer once the agent carries O_i and returns to the originator S_0.

Strong forward integrity: key leakage problem is resolved by key confidentiality that is applied by the monitoring-agent behaviour.

Insertion resilience: a malicious host tries to insert unauthorized offer, it will need one-time private key corresponding to one-time public key, which is encapsulated in the previous offer, so it will be detectable because of key confidentiality. So insertion resilience is satisfied.

Truncation resilience: truncation resilience is satisfied because malicious host needs next one-time private key to continue chain, e.g. let a malicious host truncates the chain at k, then it needs one-time private key $\underline{\mu}_{k+1}$ to continue chaining, where it is impossible because of key confidentiality.

Detecting stemming attack: corresponding to detecting deletion and insertion besides a property of key confidentiality, then detecting stemming attack is satisfied. $H(\Pi_{Am})$ and h_0 are used to protect TSPs and hosts respectively from Satan's agents [1].

Forward privacy: each encapsulated offer O_i includes μ_{i+1} which is considered as host identity indicator. Therefore the identity of S_i will not be disclosed to others (except A_m) by examining O_i.

Publicly verifiable data integrity: each host S_i verifies the integrity of encapsulated offers chain O_k (k=1...i-1) begins from O_0 to O_{i-1}, extracting $\mu_1,..,\mu_i$. In addition to integrity verification, it also verifies agent code authenticity by verifying h_0 and $H(\Pi_{Am})$ with the code of visiting agent.

Authorized re-assign offers flexibility: Co-T1 protocol assumes that for every authorized reassigning offer, it should consider the modified offer as a new one with a new key pair, which does not imply re-assigning flexibility.

Malicious identification: each host performs verification process so it can detect that previous host is considered as a malicious entity if there is data integrity attack.

Since multiple collusive hosts which do not contact A_m to obtain the authorized keys, data integrity attack can be detected when bid-agent arrives at honest host.

The identity of the malicious host(s) can be extracted by comparing the list of visited hosts (which are collected from encapsulated offers) and the A_m table.

Attack resilience: attack resilience is illustrated by Co-T1 protocol occupying AI_i field into key confirmation message.

There are four cases for showing attack resilience, which are related to $NHCI_{i+1}=[Enc_{\upsilon i+1}(r_{i+1}, S_{i+1}, TSP_{i+1})]$:

1) When the bid-agent resides at its correct host:

a. If $NHCI_{i+1}$ is changed, but it is still signed by $\underline{\mu}_i$, this implies malicious host S_i to be identified.

b. If $Sig_{\underline{\mu}_i}(NHCI_{i+1})$ is changed entirely, where malicious identification is not satisfied, whereas the monitoring-agent can not specify which host modifies it, then the monitoring-agent tries to resolve $NHCI_{i+1}$ continuing its bid-agent life cycle.

2) When the bid-agent is lost, where it is at honest host:

a. If the bid-agent A_b has its monitoring-agent A_m status, they can be reconnected after A_m is timed-out.
b. If the bid-agent has no information about its monitoring-agent, it communicates with home host to get its A_m status for continuing its life cycle.

While the monitoring-agent is not timed-out, it refuses any communication with a bid-agent that has invalid status.

6. Cost analysis and comparative studies

This section presents theoretical estimated cost for the Co-T1 protocol. This is followed by comprehensive comparative studies of this cost with T1 protocol.

6.1. Cost analysis of Co-T1 protocol

To investigate cost analysis, it is required to assume a scenario, where a bid-agent migrates through n nodes beginning from S_0 till it returns again to home S_{n+1}. Table 3 introduces notations required for cost analysis.

There are four variables, which are involved in cost analysis, such as encryption cost, hash cost, message serialization and migration cost (agent serialization cost).

The overall cost is composed of initialization cost, iterative cost and home host offer extraction cost, where each of which is considered as an estimation for the respective actual cost.

Table 3
Cost analysis notations

e	Average encryption cost.
d	Average decryption cost.
h	Average hash cost.
B_{av}	Average block size, which equals encapsulated offer size
msg	Message serialization cost for B_{av}.
mig	Agent migration cost for B_{av}.
Ec	Encryption cost.
Hc	Hash cost.
Mc	Message cost.
MGc	Migration cost.

Initialization cost:

Eqs. (1-3 and 4) show the initialization cost for key transportation, creating integrity checksums and agents migration.

$$Ec_{initial} = 7e + 2d. \quad (1)$$

$$Hc_{initial} = 2\ h. \quad (2)$$

$$Mc_{initial} = 2\ msg. \quad (3)$$

$$MGc_{initial} = 4\ mig. \quad (4)$$

Iterative cost:

Eqs. (5-8) show the four variables of iterative cost for n iterations.

$$Ec_{iterative} = [e*19n+d*18n+d*n(n+1)/2-3e-4d]. \quad (5)$$

Eq. (5) shows the iterative encryption cost which includes the following sub-costs for each iteration:
1. S_i sends request to CA for getting public key of S_0, it costs $(2e+2d)$. S_i applies integrity proof, which needs i decryption processes $(i*d)$. A_b decrypts receipt $NHCI_i$, it needs two decryption processes $(2d)$.
2. S_i sends request to CA for getting public key of TSP_i, it costs $(2e+2d)$.
3. A_b needs two encryption processes for sending next host information, which costs $(2e)$. A_m needs two decryption processes for extracting υ_0, $H(\Pi_{Am})$ and current iteration information, it costs $(2d)$. A_m extracts next host information by two decryption processes $(2d)$.
4. A_m requests CA for getting public keys of S_{i+1} and TSP_{i+1}, it costs $(2e+2d)$. A_m creates $NHCI_{i+1}$, it costs (e). A_m sends to home host a key list of next iteration which is decrypted by S_0, it costs $(2e+2d)$.
5. A_m sends one-time key pair to A_b, which costs $(2e)$. A_b extracts its one-time private key by two decryption processes, and sends key confirmation message by two encryption processes, then they cost $(2d+2e)$. A_m decrypts key confirmation message for non-repudiability by two decryption processes $(2d)$. A_b encapsulates its offer which costs two encryption processes $(2e)$.
6. A_b signs $NHCI_{i+1}$ by its one-time private key for data authenticity, which costs (e). A_m

encrypts next iteration information $\{(r_{i+1}, \upsilon_{i+1}, S_{i+1}, \mu_{i+1}, \underline{\mu}_{i+1})\}$ into confidential channel by a symmetric key to be serialized to next TSP, which costs (e).

Since TSP_1 in first iteration needs getting υ_0 from CA which costs ($2e$+$2d$) and needs ($2e$) for confiding home host key and current iteration data, and no need for sub-cost 2 (-$2e$-$2d$), so encryption cost is increased by ($2e$). While for iteration n, no need for sub-costs 4 and 6, but it needs signing required data by last TSP and S_0, so encryption cost is reduced by (-$5e$-$4d$). Eventually iterative encryption cost is reduced by (-$3e$-$4d$).

$$HC_{iterative} = [(n+2)(n+1)/2 - 1 + n] * h. \qquad (6)$$

Eq. (6) shows iterative hash cost, where each S_i applies integrity process which needs applying i iterations of one-way hash function, in additional to creating its offer which costs one extra. While each TSP verifies Π_{Am}.

$$MC_{iterative} = 10n - 1 \ msg. \qquad (7)$$

Eq. (7) shows iterative message cost, where each iteration needs 10 *msg* except iteration *n* which needs 9 *msg*, whereas no need for TSP_n to send next keys information to S_0.

$$MGC_{iterative} = [(n+2)(n+3)/2 - 4 + (n+1)(n+2)/2 - 2] * mig. \qquad (8)$$

Eq. (8) shows iterative migration cost, where each iteration requires serializing encapsulated i+1 offers in addition to next host information which is not required for iteration n. Each iteration also requires serializing its tables costs (i+2)**mig*, with assumption of serializing table entry equals serializing encapsulated offer, where at last iteration A_m carries only required data not the overall table.

<u>*Home host offer extraction cost:*</u>

Home host performs verifying its dummy offer, extracting offers by one-time public keys and $\underline{\upsilon}_0$, and decrypting required data which are sent by last TSP. Where eqs. (9 to 12) show Home host offer extraction cost.

$$EC_{offer\ extraction} = (2n+3) * e. \qquad (9)$$

$$HC_{offer\ extraction} = (n+2) * h. \qquad (10)$$

$$MC_{offer\ extraction} = 0. \qquad (11)$$

$$MGC_{offer\ extraction} = 0. \qquad (12)$$

Then, the overall costs for the four variables of cost analysis after assuming e=d, are summarized by eqs. (13 to 16).

$$EC_{overall} = [0.5n^2 + 39.5n + 5] * e. \qquad (13)$$

$$HC_{overall} = [0.5n^2 + 3.5n + 4] * h. \qquad (14)$$

$$MC_{overall} = [10n + 1] * msg. \qquad (15)$$

$$MGC_{overall} = [n^2 + 2(k+2)n + 2(k+1)] * mig. \qquad (16)$$

Eqs. (13-14 and 16) are considered as $O(n^2)$, while eq. (15) is O(n), with respect to eq. (16), it is calculated after assuming agent-code migration cost, where Size (Π_{Am}) = Size(Π_{Ab}) =k* B_{av}.

6.2. Comparison between Co-T1 and T1 protocols

This sub-section presents a comparison according to cost analysis variables between Co-T1 and T1 protocols, where it is illustrated in table 4.

Table 4 presents four variables of cost analysis, whereas it shows that encryption cost is better in Co-T1 protocol rather than T1 protocol, which uses protected list of already visited host, where it adds additional cost.

Migration cost of Co-T1 protocol is higher than T1 protocol, where the former has additional cost which is monitoring-agent migration. Although migration cost is worse in Co-T1 protocol than T1 protocol, it satisfies attack resilience. Message and hash costs are almost the same for the two protocols.

Table 4
Comparison between Co-T1 and T1 protocols

Variables	Co-T1 Protocol	T1 Protocol
Encryption cost	$0.5n^2+39.5n+5$	$1.5n^2+38.5n+30$
Hash cost	$0.5n^2+3.5n+4$	$0.5n^2+2.5n+2$
Messages	$10n+1$	$8n+6$
Migrations	$n^2+2(k+2)n+2(k+1)$	$0.5n^2+(3.5+k)n+(3+k)$

7. Conclusions and future work

In spite of mobile agents are supported to the electronic commerce. However they are vulnerable.

Co-T1 protocol is not only capable of achieving most of data-state security requirements, but it also able to detect malicious activities in addition to identifying the malicious hosts. Attack resilience issue has been proposed, and is occupied through key confirmation in Co-T1 protocol, on other hand; it is supported when bid-agent is lost. Eavesdropping can exploit the communications of CA.

Reliability issue should be taken into account in future work, where the persistence of coordinating agents. Attack resilience needs more investigation in our future.

Re-assign offers flexibility has not been achieved, so it has a position in future work, where it will be satisfied by using SAC protocol as integrity chain. Co-T1 protocol should not be restricted to TSPs existence.

References

[1] V. Roth, "Programming Satan's Agents", In: K. Fischer, D. Hutter, (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).

[2] 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, pp. 1–12 (2002).

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

[4] 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, pp. 195–207 (1998).

[5] S. Loureiro, "Mobile Code Protection", Ph.D. thesis, Ecole Nationale Sup´erieure des T´el´ecommunications, January (2001).

[6] Vandana Gunupudi, Stephen R. Tate, "Performance Evaluation of Data Integrity Mechanisms for Mobile Agents", Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04), Vol. 2, IEEE Computer Society, April (2004).

[7] S. Songsiri, "A New Approach for Computation Result Protection in the Mobile Agent Paradigm", 10th IEEE Symposium on Computers and Communications (ISCC'05) pp 575-581 (2005).

[8] P. Maggi and 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, pp. 851–858 (2003).

[9] J.Y. Zhou, J. Onieva and J. Lopez "Protecting Free Roaming Agents Against Result-Truncation Attack", In Proceedings of the IEEE Vehicular Technology Conf. (Los Angles, California, Sept. 26-29 (2004).

[10] V. Roth, "Secure Recording of Itinerary through Co-Operating Agents", LNCS, Vol. 1543/1998, Springer Berlin/Heidelberg, pp. 590 (1998).

[11] Satoh, "Building Reusable Mobile Agents for Network Management", Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on Vol. 33, Issue 3, Aug. pp. 350–357 (2003).

Received July 15, 2007
Accepted September 23, 2007