http://ansinet.com/itj

ITJ

ISSN 1812-5638

# INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (1): 24-31, 2008
ISSN 1812-5638


# A Robust Framework for Protecting Computation Results of Mobile Agents

Abid Khan, Xiamu Niu and Zhang Yong
Information Security Technique Research Center, Harbin Institute of Technology,
Shenzhen Graduate School, People's Republic of China

**Abstract:** A data integrity solution for mobile agents is presented. The proposed scheme combines digital watermarking and digital signature to achieve the desired security requirements of strong forward integrity, truncation resilience and non-repudiation. The results computed at each hop are first watermarked and then digitally signed. When the agent returns to the home platform we verify the signature first and then extract watermark. Any tampering with the results can be determined in the verification process. We have implemented the proposed scheme with various hashing algorithms like SHA-1, SHA-256 and SHA-512 for a price comparison scenario. For digital signature we have used RSA based signature. We have applied t-test to check the significance of present results. Present experimental results suggest that it can be used in an e-commerce application.

**Key words:** Mobile agent security, data Integrity, watermarking, digital signature, truncation resilience, forward Integrity, non-repudiation

## INTRODUCTION

Mobile agents are software entities consisting of code, data and state that can move from one host to another in a network performing task on behalf of a user or an application. Mobile Agents have been employed in a variety of applications such as information retrieval, workflow management, network management etc. Mobile agents offer many advantages over traditional client server paradigm such as saving network bandwidth, introducing concurrency, adding client specific functionality to servers etc. Despite all these advantages the use of mobile agents is limited mainly because of the security issues associated with them. The security in mobile agents can be divided into two categories: first is the security of the host executing the agent and second is the security of the mobile agent from malicious host. The second type of security can further be divided into two subtypes (1) Security of static code (2) Security of dynamic data state. The security of the dynamic data state is the focus of this study. This paper proposes a method for protecting the computation results of mobile agents. The proposed idea is to initially use reversible watermarking to watermark the results collected by a mobile agent at a remote host. Then compute a cryptographic hash over the watermark results to protect against attacks such as non-repudiation. Any tampering with results can be detected. Digital signature uses a one way function like SHA-1 or SHA-256 or SHA-512. After watermarking the results, these watermarked results are used as input to the hash function. The produced hash is then signed by the private key of the host thus forming a digital signature. The watermarked results, signature and public key of the signer are the proof of the agent execution at a host. This information (watermarked results, signature and public key) is produced per hop on which a mobile agent is executed. This information travels with the agent from server to server until the agent returns to the home platform. When the mobile agent returns to the home platform first of all the hash of the watermarked results is recomputed and then it is verified with the public key of each host. If the stored and produced hash match we say that the results are untampered and can be considered as trusted. In case if there is no match we say that the host has tampered with the results and the host is considered as malicious host.

## RELATED WORK

Mobile agent's execution platform is responsible for providing necessary environment and resources in order to successfully execute an agent. Methods that have been devised to protect the computation results of a mobile agent includes Partial Results Authentication Code (PRAC) (Yee, 1999), Chain Hash Chaining (Karjoth *et al*., 1998), Set Authentication code (Loureiro, 2001) and Ring Signature (Lin *et al*., 2004). Yee proposed the idea of PRAC in which the results of an agent's computation at

**Corresponding Author:** Abid Khan, Information Security Technique Research Center, Harbin Institute of Technology, Shenzhen Graduate School, People's Republic of China

each host is encapsulated using MAC (Message Authentication Code). The result of an agent's execution combined with MAC of the results is called PRAC. This method requires the agent to produce a secret key for each host, using one way function, from the initial secret key given by the originator. This method makes sure to provide forward integrity which means, "none of the results calculated prior to a malicious host can be tampered". In Hash Chaining method the partial results are chained to the identity of the next host in itinerary. This method allows the originator to determine where exactly the chaining is broken if a host behaves maliciously by tampering with the partial results. Although this method provides stronger security it is not flexible enough.

Vigna (1998) gave the idea of cryptographic traces based on the execution tracing and cryptography. It allows the detection of attacks against the code, state and execution flow for mobile agents. These facts can be used to punish the attackers. However this method has some limitation e.g. mobile agent code is executed again only in case of suspicion although there is suspicion detection protocol but its cost is too high.

Roth (1999) proposed the idea of transferring commitments to other cooperating agents. This agent can do tasks like storing, gathering and verifying the information. The underlying principle is the generalization of the trusted third party. These cooperating agents share secrets and decisions and have a disjoint itinerary. Each cooperating agent record the itinerary of other cooperating agent. This makes collusion attacks difficult but not impossible. However this technique is only effective if this requirement can be realistically met.

Hohl (2000) proposed the idea of reference state. A reference state is a state that is produced by a non-attacking host or reference host. In this model the execution on one host is checked unconditionally and immediately on the next host, regardless of whether this host is trusted or untrusted.

Villate *et al.* (2000) proposed the notion of data lockers. It is a service provided for mobile users to keep their data in secure and safe locations.

Rivest *et al.* (2001) gave the idea of Ring Signature, in which no prior setup process and no group manager are necessary. It is a special form of generalized group signature.

Loureiro (2001) proposed an original cryptographic technique called Set Authentication Code. In this technique each host exchanges a secret key with the agent owner. This key is used to calculate MAC on its results. When the mobile agent returns to the home platform this integrity proof can be verified.

Roth (2001) pointed out some flaws in some of the proposed protocols of Karnik and Tripathi (1999), Corradi *et al.* (1999) and Karjoth *et al.* (1998). According to Roth these protocols failed because they were unable to bind the collected data by agent with its static code. He proposed fixing these protocols by binding confidential data and acquired data via constructing an agent kernel and ciphertexts. This allows authorized hosts to detect whether a ciphertext brought by an agent actually belongs to the agent.

McDonald *et al.* (2004) gave the idea of using multi-agent architecture. They used different classes of agents like task agent, data collection agent and data computation agent. The task agents are responsible for the completion of job the user wants to complete. Computation agents perform the desired computation in single hop or multi-hop fashion. Data collection agents responsible for data state collection.

## PROPOSED IDEA

Our proposed scheme firstly watermark the computation results of the mobile agent at a host i using the transformation in Eq. 1-4. The details of watermark embedding process are described in the Table 1. We represent these watermark results by $R_w$. For the watermarking purpose we used Tian's expansion algorithm (Tian, 2001). Tian's expansion algorithm is defined for 8-bit pixels images. Since we are dealing with data collected by mobile agents, we will be using that algorithm for 128 ASCII values. For that we have modified the limits to 0→127 to avoid overflow and underflow.

$$l = \lfloor x + y \rfloor /2 \quad (1)$$

$$h = x - y \quad (2)$$

$$x' = l + \lfloor h' + 1 \rfloor /2 \quad (3)$$

$$y' = x' - h' \quad (4)$$

The watermark embedding process can be seen in example in Table 1 below. After watermarking the results we use RSA based digital signature to digitally sign these watermarked results. We make an assumption here that since these results are collected at this host there is no point of hiding them from the current host in other words the results collected by a mobile agent at a host $H_i$ the host $H_i$ is considered as trusted to a certain extent but not completely e.g., for producing digital signature and not to the point after the agent has obtained them etc. But not

for the results collected at host $H_{i-1}$ or $H_{i+1}$. So we can compute the hash of these results at host $H_{i-1}$. Then the host $H_i$ produces a pair of public and private key. The private key is used to sign the hash to produce the signature. The public key is given to the mobile agent to be used later at the origin platform for signature verification. After embedding the watermark in the results, signing the results and giving the public key to the mobile agent, the mobile agent can visit the next host $H_{i+1}$ in its itinerary. When the mobile agent returns to the origin platform we recompute the signature using the public key of each host and compare with the signature previously produced at the remote host. If the signature matches with the previously stored signature we say the results are not tampered so we can extract the watermark using the transformation in Eq. 5-8:

$$l' = \lfloor x' + y \rfloor / 2 \quad (5)$$

$$h' = x - y' \quad (6)$$

$$x = l + \lfloor h + 1 \rfloor / 2 \quad (7)$$

$$y = x - h' \quad (8)$$

The watermark extraction process can be shown in Table 2.

**Mobile agent watermarking:** Mobile Agent Watermarking is a technique for protecting the computation results of mobile agents by using digital watermarking. The idea of using watermarking for mobile agents protection was initially given by Esparza *et al.* (2003), but there is no experimental evidence of their work. Esparza gave no explanation about how the watermarks are actually embedded and retrieved and what is the effectiveness of this technique. We believe that Tian's expansion algorithm can be successfully implemented for the purpose of watermarking the computation results of mobile agent. There are two main reasons for selecting Tian's algorithm. First of all it is simple and easy to understand and it can be fully implemented in software thus making it ideal for mobile agents.

**Security model:** In order to test the proposed idea we have implemented a simple Ticket Booking Agent. The primary goal is to make sure that the partial results collected by mobile agent are in contact and any modification or tampering can be detected by the originator. The owner of the agent wants to fly say from Beijing to Hong Kong on next weekend. The owner of the agent wants to buy the cheapest ticket. The owner sends his mobile agent to various airline agencies which are providing the ticket service. The owner sends his agent with his desired preferences to various airlines servers to

Table 1: Watermark embedding process

| x | y | *l* | h | $(h)_2$ | $(h')_2$ | h' | x' | y' | $(x')_2$ | $(y')_2$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 90 | 72 | 81 | 18 | 10010 | 100010 | 34 | 98 | 64 | 1100010 | 1000000 |
| 57 | 55 | 56 | 2 | 10 | 100 | 4 | 58 | 54 | 111010 | 110110 |
| 51 | 49 | 50 | 2 | 10 | 100 | 4 | 52 | 48 | 110100 | 110000 |
| 52 | 48 | 50 | 4 | 100 | 1000 | 8 | 54 | 46 | 110110 | 101110 |
| 49 | 45 | 47 | 4 | 100 | 1000 | 8 | 51 | 43 | 110011 | 101011 |
| 54 | 45 | 49 | 9 | 1001 | 10001 | 17 | 58 | 41 | 111010 | 101001 |
| 50 | 48 | 49 | 2 | 10 | 100 | 4 | 51 | 47 | 110011 | 101111 |
| 54 | 48 | 51 | 6 | 110 | 1010 | 10 | 59 | 49 | 111011 | 110001 |
| 49 | 48 | 48 | 1 | 1 | 10 | 2 | 49 | 47 | 110001 | 101111 |
| 58 | 48 | 53 | 10 | 1010 | 10010 | 18 | 62 | 44 | 111110 | 101100 |
| 56 | 53 | 54 | 3 | 11 | 101 | 5 | 57 | 52 | 111001 | 110100 |
| 52 | 48 | 50 | 4 | 100 | 1000 | 8 | 54 | 46 | 110110 | 101110 |

Table 2: Watermark extraction process

| x' | y' | *l'* | h' | $(h)'_2$ | $(h)_2$ | h | x | y |
|---|---|---|---|---|---|---|---|---|
| 98 | 64 | 81 | 34 | 100010 | 10010 | 18 | 90 | 72 |
| 58 | 54 | 56 | 4 | 100 | 10 | 2 | 57 | 55 |
| 52 | 48 | 50 | 4 | 100 | 10 | 2 | 51 | 49 |
| 54 | 46 | 50 | 8 | 1000 | 100 | 4 | 52 | 48 |
| 51 | 43 | 47 | 8 | 1000 | 100 | 4 | 49 | 45 |
| 58 | 41 | 49 | 17 | 10001 | 1001 | 9 | 54 | 45 |
| 51 | 47 | 49 | 4 | 100 | 10 | 2 | 50 | 48 |
| 59 | 49 | 51 | 10 | 1010 | 110 | 6 | 54 | 48 |
| 49 | 47 | 48 | 2 | 10 | 1 | 1 | 49 | 48 |
| 62 | 44 | 53 | 18 | 10010 | 1010 | 10 | 58 | 48 |
| 57 | 52 | 54 | 5 | 101 | 11 | 3 | 56 | 53 |
| 54 | 46 | 50 | 8 | 1000 | 100 | 4 | 52 | 48 |

check the availability of the ticket, the price as well as other relevant information e.g. departure time, arrival time, flight is direct or number of stops etc. The agent queries the servers and return to the originator with the results of its computation. In order to get the best suitable price of tickets for its owner the agent must keep the prices/data collected at all the hosts. When a mobile agent collects price from one host say $S_i$ and move to the next host $S_{i+1}$, the price of the former host $S_i$ must be kept secret from the later sever otherwise the later host may take advantage e.g. offering a false offer or modifying previously stored offers. The host $S_{i+1}$ may offer a price that is not the actual price offered by it or in worst case it can modify the price collected prior to visiting $s_{i+1}$ in order to get unfair advantage. Mobile agent must be protected against such attacks.

The main objectives of present study are confidentiality and integrity. Confidentiality here means to reveal cleartext only at trusted hosts. Integrity means the agent must be protected such that it can collect new data set at each host they visit but also any tampering with the pre-existing collected data set must be detected by any trusted host.

**Security properties:** Karjoth *et al.* (1998) have defined a set of security properties which are considered as the basic guidelines for the data integrity mechanism. Here we took the liberty of modifying the original text slightly. While defining these properties Karjoth assumed that a malicious host has captured the agent containing a set of encapsulated offers $O_1, O_2, ..., O_m$ where $m \leq n$ and $O_m$ is the last host visited by the agent before being captured.

**Forward integrity:** According to Yee (1999), none of the partial results collected prior to a malicious host can be modified without detection. If a mobile agents visits a number of hosts $S_1, S_2, ..., S_n$ and the first malicious host is $S_m$ where $1 \leq m \leq n-1$ then none of the partial results collected at hosts $S_i (i \leq m)$ can be undetectably modified by a malicious host.

**Strong forward integrity:** If a mobile agent visits a number of hosts $S_1, S_2, ..., S_n$ and the first malicious host it encounters is $S_m$ then none of the encapsulated offer $O_k$ where $k>m$ can be modified.

**Insertion resilience:** Only hosts that are authorized to insert offers can add the offers.

**Non-repudiation:** No hosts can deny the offers that it made to a visiting agent.

**Cryptographic notations:**

| | | |
|---|---|---|
| $S_0$ | = | Originator |
| $S_i (i \geq 1)$ | = | Intermediate hosts |
| $O_i, 1 \leq i \leq n$ | = | Actual offer at host $H_i$ |
| $O_i$ | = | Encapsulated offer |
| $W(o_i)$ | = | Watermarked offer/results |
| $H_i(W(o_i))$ | = | Cryptographic hash produced at host $H_i$ |
| $(K_1)_i$ | = | Private Key produced at host $H_i$ |
| $(K_2)_i$ | = | Public Key produced at host $H_i$ |
| $SIG_{k_1}(H_i(W(O_i)))$ | = | Signature Produced |

**Setup on origin:** $\Omega_i$ Set of encapsulated offers

S0 → S1:

$W_0 = \theta$ {set of offers}

$\Omega_0 = \theta$ {set of encapsulated offers}

**Initial visit on a host:**

$W(O)_i$ = Embed Watermark $(O)_i$

$H_i = h(W(O)_i)$ {Cryptographic hash of watermark offer}

$$O_i = (W(o_i), SIG_{(K_1)_i}(H_i), (K_2)_i)$$

**Protocol:** The mobile agent partial is described in Fig. 1.

$$Si \rightarrow Si+1: \Omega_i = \Omega_{i-1} \cup \{Oi\}$$

**Watermark embedding algorithm:**

(1) $R = C_0, C_1, ...., C_{n-1}, C_n$

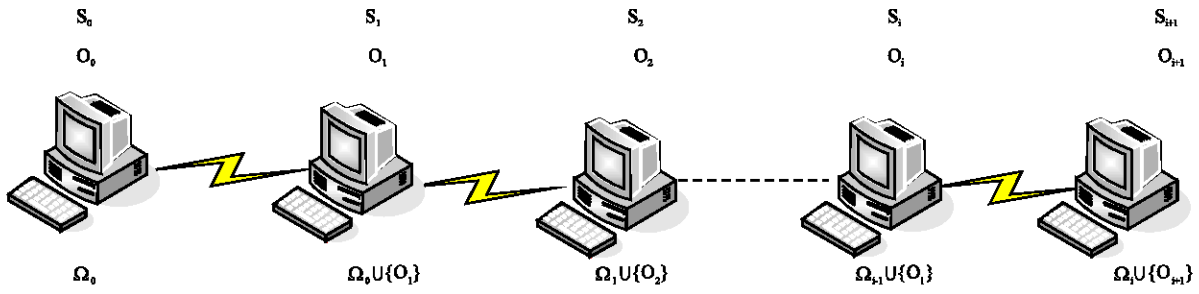


Fig. 1: Mobile agent collecting partial results

(2) $(x, y) = (C_0, C_1)$
(3) $l = \lfloor x + y \rfloor /2$, $h = x - y$
(4) Expand h into its binary representation $h = r_m, r_{m-1}, ..., r_1, r_0$
(5) Embed the watermark bit after MSB bit and right shift all the bits one bit.
$h_w = r_m, b_w, r_{m-2}, r_{m-3}, ..., r_1, r_0$
Where $b_w = \{0, 1\}$
(6) $h' = \text{Interger}(h_w)$
(7) $x' = l + \lfloor h' + 1 \rfloor /2$
(8) $y' = x' - h'$
(9) $R_w = x'_i \parallel y'_i$

In above algorithm $C_0, C_1, ..., C_{n-1}, C_n$ represent the results collected by mobile agent. R is a set consisting of the ASCII values of these results. After this we pick a pair x and y. In Step 3 we compute the values of difference and average for the first pair. The value of difference is expanded to its binary representation and then a watermark bit is embedded after the MSB-bit. The watermarked difference value $h_w$ is then converted to its integer value $h'$. The new values ($x'$ and $y'$) of x and y are computed based on the value of $h'$. At the end we concatenate the pair $x'$ and $y'$. Next we pick the second pair $(C_2, C_3)$ and repeat the steps from 2 to 9. We are done when all the pairs are finished.

**Signature verification algorithm:** The way we create and verify digital signature is stated in the following algorithm lets suppose that $k_1$ Private Key and $K_2$ is the Public Key

(1) Calculate the hash of watermark results using Hash Algorithm (e.g., SHA-1)
i.e., $h = \mathcal{H}(M_w)$
(2) Using RSA Private Key $k_1$ encrypt the hash i.e., $S_1 = E_{K_1}(h)$
(3) For Verification re-compute the hash using same algorithm $h = \mathcal{H}(M_w)$
(4) Encrypt the signature using public key of RSA i.e., $S_2 = E_{K_2}(h)$
(5) If $S_1 = S_2$ verify otherwise reject.

## IMPLEMENTATION AND EXPERIMENTAL RESULTS

We have implemented the proposed idea using IBM Aglets. For cryptographic primitives we have employed JCE/JCA. The provider we have selected is Bouncy Castle Provider (Hook, 2005). Aglet is an open source framework developed by IBM Tokyo Research Laboratory (TRL). Mobile agents in IBM Aglets are called Aglets. Aglets are

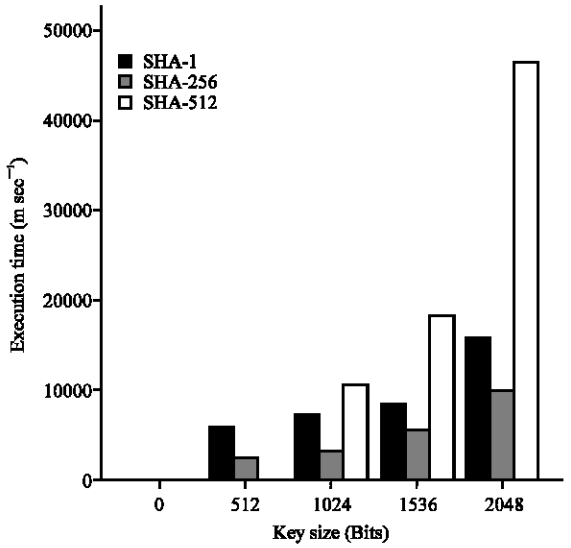


Fig. 2a: Execution time with various hash algorithms

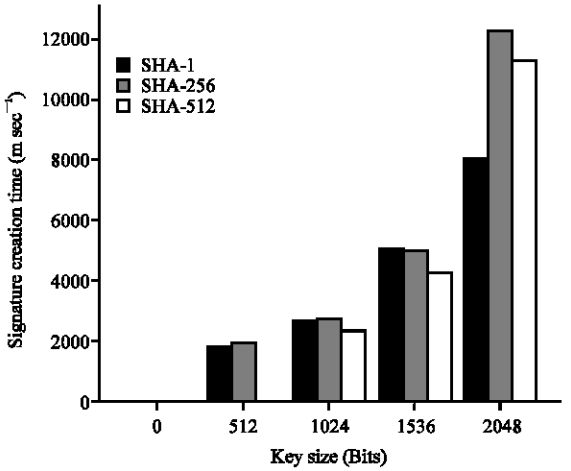


Fig. 2b: RSA signature creation time for various hash algorithms

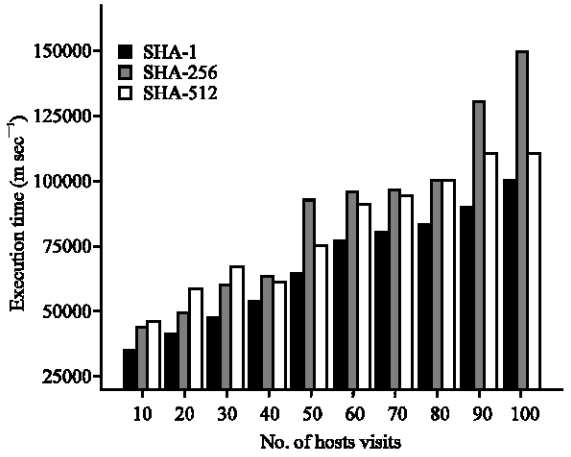


Fig. 2c: Total execution time

fully written in java. The Aglets platform provides many services like mobility, communication between aglets etc. Agent programming with aglets is based on event-driven programming, reacting on when it received a specific message (Lange and Oshima, 1998). We have performed the experiments in a windows based environment with five Pentium IV, 3.01 GHZ Processor, 512MB RAM computer for the remote and home computers (agent's home). We have used RSA based signature algorithm with various types Hash algorithms like SHA-1, SHA-256 and SHA-512. Initially we keep the key size to 512-bits and then gradually increase it for different experimental purposes. We increase key size to 1024-bits, 1536-bits and 2048-bits. We compute the overhead on the remote host by calculating the time to generate digital signature and total execution time (turn around time) with variable key size. Figure 2a shows the execution time for RSA algorithm on remote hosts using variable size key. We can see that as we increase the key size the time to create the signature also increases. But for stronger security we need to have a key size to be resistant enough against attacks. We can see that the execution time for SHA-1 is less than other hash algorithms as we increase the key size. Figure 2b describes the signature creation time for RSA Algorithm with variable key size. RSA with SHA-256 has the largest time to generate the signature. SHA-512 has comparably less time but it requires a key size of at least 1024 bits. RSA with SHA-1 is the fastest one and takes less time as compared to other two hash algorithms to generate signature. Figure 2c presents the total turn around time as the mobile agent visits more hosts this time increases. In case of SHA-1 the execution time is less compared to other algorithms. In case SHA-256 total turn around time increases rapidly. But in case of SHA-512 this time is still less than SHA-256. So we can see that SHA-1 performs better than SHA-256 and SHA-512. But as we know there are some collision detected in SHA-1 so its use is not recommended anymore. We have used this only for experimental purposes just to give a comparison of different hashing algorithm with RSA Digital Signature.

As present experimental shows that this framework can be implemented efficiently with little load on the processing. We also notice that the watermark extraction is independent of the key size. As the Signature verification and watermark extraction is carried out at the agent owner host so we can increase the key size more but that will put an extra burden on the remote hosts for signature creation.

## EVALUATIONS

Significance differences are determined by applying a paired sample t-test. Table 3-5 show significant difference among the three hash algorithms.

$$t = \frac{\bar{d} * \sqrt{n}}{s_d} \tag{9}$$

It follows the t-distribution with the degree of freedom n-1:

Where:

$$\bar{d} = \frac{\sum d_i}{n} \quad \text{and} \quad s_d = \frac{\sum (d_i - \bar{d})^2}{n-1}$$

Table 3: t-test on the signature creation time for RSA with SHA-1, SHA-256 and SHA-512

| | | Paired difference | | | | | |
|---|---|---|---|---|---|---|---|
| Paired samples test | | Mean | Std. deviation | Std. error mean | t | df | Sig. (2-tailed) |
| Pair 1 | SHA-256-SHA-512 | -13536.800 | 16603.93718 | 8301.96900 | -1.631 | 3 | 0.20 |
| Pair 2 | SHA-256-SHA-1 | -4020.000 | 1341.06412 | 670.53206 | -5.995 | 3 | 0.00 |
| Pair 3 | SHA-512-SHA-1 | 9516.813 | 15535.72680 | 7767.86300 | 1.225 | 3 | 0.30 |

Table 4: t-test on the execution time for RSA with SHA-1, SHA-256 and SHA-512

| | | Paired difference | | | | | |
|---|---|---|---|---|---|---|---|
| Paired samples test | | Mean | Std. deviation | Std. error mean | t | df | Sig. (2-tailed) |
| Pair 1 | SHA-256-SHA-1 | 1067.52500 | 2094.40069 | 1047.20000 | 1.019 | 3 | 0.383 |
| Pair 2 | SHA-512-SHA-1 | 38.54500 | 2183.56986 | 1091.78500 | 0.035 | 3 | 0.974 |
| Pair 3 | SHA-256-SHA-512 | 1028.98000 | 635.37634 | 317.68817 | 3.239 | 3 | 0.048 |

Table 5: t-test on the total turn around time for RSA with SHA-1, SHA-256 and SHA-512

| | | Paired difference | | | | | |
|---|---|---|---|---|---|---|---|
| Paired samples test | | Mean | Std. deviation | Std. error mean | t | df | Sig. (2-tailed) |
| Pair 1 | SHA-1-SHA-256 | -20880.2 | 14387.072370 | 4549.592 | -4.589 | 9 | 0.001 |
| Pair 2 | SHA-1-SHA-512 | -14100.2 | 4260.799030 | 1347.383 | -10.465 | 9 | 0.000 |
| Pair 3 | SHA-256-SHA-512 | 6780.0 | 14685.711174 | 4644.030 | 1.460 | 9 | 0.178 |

Thus, we obtain the following results as mentioned in Table 3 and 4 with level of significant 0.01 and 0.05. We can see that there is a significant difference in the various hash algorithms when used for RSA-signature. The execution time is critical in our case as we do not want to put a lot of burden on the executing host. But as we can see there is some time consumed by the agent at the remote hosts for key generation and execution. The total turn around time is the total time consumed by the mobile agent from start to end i-e. The time from when it leaves the home platform to the time it comes back to home platform after executing there. As we can see that this time increases gradually as the number of host increases. Our proposed scheme does not suffer from cut and paste attacks as mentioned in (Roth, 2001). It is computationally infeasible to find a collision in the hash function thus providing stronger protection even against the most powerful attackers. We understand that the watermarks can be added, removed or modified. This can be avoided by using hash functions justifying our use of them. The host can not repudiate from the results produced by agent and the owner of the agent can verify the authenticity of the results by re-computing the hash value. The framework is flexible enough as we can implement different types of stronger watermarking algorithms.

## CONCLUSIONS

Mobile agents offer a lot of advantages but their use is limited mainly because of the security problems associated with them. Security in mobile agent system can be divided into two types (1) security of host from malicious mobile agent (2) security of mobile agent from malicious host. The second type of security can further be divided into two subtypes (1) Security of static code (2) Security of dynamic data state. The security of the dynamic data state is still an open problem of research. In this paper we proposed a method of protecting the computation results of the mobile agent. Our proposed scheme uses digital watermarking and digital signature. We first embed a watermark in the results on each host. Then these watermarked results are used as input data to a hash function producing hash value. This hash value is then signed by each host on which it is produced as the mobile agent traverses it forming a digital signature. When the agent returns to the home platform we verify the signature. Any tampering with the results can be readily detected by hash function. We have implemented 3 different types of hash functions SHA-1, SHA-256 and SHA-512 that can work with RSA. We compare the performance of these hash functions so as to determine the load on each host. The experimental results give a measure of the overhead of the framework on executing platform. Unlike other proposed methods like PRAC, set authentication code etc our method does not suffer form the attacks identified like cut and paste and oracle attacks. The reason for that is the computational infeasibility of producing a hash that collides with hash produced at each host. However some questions are still left unanswered for future research, more specifically those regarding the protection of the watermarking algorithm. A realistic solution is still needed to protect the code that watermarks the agent's results from reverse engineering attacks. Those issues are beyond the scope of our research but future research can focus on these problems more deeply and efficiently.

## ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation of China (60671064), the Chinese national 863 high-tech projects (2007 AA01Z458, 2005AA733120), the Science Foundation of Guangdong Province (05109511), the Society Development plan of Guangdong Province (2006B37430001), the Foundation for the Author of National Excellent Doctoral Dissertation of China (FANEDD-200238), the Scientific Research Foundation of Harbin Institute of Technology (HIT.2003.52), the Science of the Foundation for the Excellent Youth of Heilongjiang Province and the Program for New Century Excellent Talents in University (NCET-04-0330). Ph.D scholarship for Mr. Abid Khan is provided by COMSATS Institute of Information Technology (CIIT), Pakistan.

## REFERENCES

Corradi, A., R. Montanari and C. Stefanelli, 1999. Mobile agents protection in the Internet environment. In: COMPSAC, pp: 80-85.

Esparza, O., M. Fernandez, M. Soriano, J.L. Munoz and J. Forne, 2003. Mobile Agent Watermarking and Fingerprinting: Tracing Malicious Hosts. In: Database and Expert System Applications (DEXA), Vol. 2736 of LNCS. Springer-Verlag.

Hohl, F., 2000. A framework to protect mobile agents by using reference states. In: Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS), 10-13 April Taipei, TAIWAN

Hook, D., 2005. Beginning Cryptography with Java. Wrox Press.

Karjoth, G., N. Asokan and C. Gulcu, 1998. Protecting the computation results of free-roaming agents. In: 2nd International Workshop on Mobile Agents, Vol. 1477 of Lecture Notes in Computer Science, Springer-Verlag, pp: 195-207.

Karnik, N.M. and A.R. Tripathi, 1999. Security in the Ajanta mobile agent system. Technical Report TR-5-99, University of Minnesota, Minneapolis, MN 55455, USA.

Lange, D.B. and M. Oshima, 1998. Programming and Deploying Java Mobile Agents with Aglets, Addison Wesley.

Lin, H.C., S.M. Yen and H.S. Chen, 2004. Protection of mobile agent data collection by using ring signature. In: Proceedings of IEEE International Conference on Networking, Sensing and Control Taipei, Taiwan.

Loureiro, S., 2001. Mobile Code Protection. Ph.D Thesis, ENST Paris/Institut Eurecom.

McDonald, J.T., Y. Alec and W.C. Thompson, 2004. Mobile agent data integrity using multi-agent architecture. In: Proceedings of the International Workshop on Security in Parallel and Distributed Systems (PDCS), San Francisco, CA.

Rivest, R.L., S. Adi and T. Yael, 2001. How to Leak a Secret. Advances in Cryptology-ASIACRYPT, Lecture Notes in Computer Science, Vol. 2248. Springer-Verlag, pp: 552-565.

Roth, V., 1999. Mutual Protection of Co-Operating Agents. Secure Internet Programming: Security Issues for Mobile and Distributed Objects, LNCS 1603, New York, USA: Springer-Verlag, pp: 275-285.

Roth, V., 2001. On the Robustness of Some Cryptographic Protocols for Mobile Agent Protection. In: Proceeding of Mobile Agents, Vol. 2240 of Lecture Notes in Computer Science. Springer Verlag.

Tian, J., 2002. Wavelet-based reversible watermarking for authentication. In: Proceedings of Security and Watermarking of Multimedia Contents IV. Electronic Imaging, 4675: 679-690.

Vigna, J., 1998. Cryptographic Traces for Mobile Agents. Mobile Agent and Security, LNCS 1419, Springer, pp: 137-153.

Villate, Y., A. Illarramendi and E. Pitoura, 2000. Data Lockers: Mobile-Agent Based Middleware for the Security and Availability of Roaming Users Data. In: 7th International Conference on Cooperative Information Systems (CoopIS), Eilat, Israel, September 6-8, LNCS 1901, Springer, pp: 275-286.

Yee, B., 1999. A Sanctuary for Mobile Agents. In: Secure Internet Programming, Vitek, J. and C. Jensen (Eds.). Vol. 1603 of Lecture Notes in Computer Science, Springer-Verlag, pp: 261-273.