http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSI*net*

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

Information Technology Journal 9 (1): 79-88, 2010
ISSN 1812-5638
© 2010 Asian Network for Scientific Information

An Integrity Batch Report Scheme Based on the Waiting Stack

[1,2]C. Chang, [2]L. Gao, [2]H. Kou, [2]X. Liu, [1]Z. Qin and [2]G. Xu
[1]Institute of Computer Science and Technology, Xi'an Jiaotong University,
710049, Xi'an, Shaanxi, People's Republic of China
[2]Institute of Electronic and Technology, Information Engineering University,
450004, Zhengzhou, Henan, People's Republic of China

Abstract: An integrity batch report protocol based on the waiting stack is proposed, which aims at solving the bottleneck problem of the Trusted Platform Module (TPM) signature of integrity report. In the protocol, integrity attestation requests are placed sequentially in a waiting stack with constant head part and varying length. The TPM processes the integrity batch report one after another and thus the flexibility problem of trusted gateway can be solved efficiently. This study also proposes a data compressing method based on the Merkle tree to reduce the communication complexity in processing integrity batch report. The length of fresh number connection string transferred to the challenger can be compressed from $m\times20$ bytes to $\log_2^{m}\times20$ bytes. The laboratory result indicates that the responding capacity of gateway adopting integrity batch report protocol is prior to adopting TCG integrity report protocol and the processing capacity of gateway can be improved from 1~3.3 to 32~256 times per second.

Key words: Trusted computing, integrity report, batch, Merkle tree

INTRODUCTION

Trusted Network Connect (TNC) is mapped out by Trusted Computing Group (TCG), the purpose of which is ensuring the trusted network connection between the terminal with TPM and the network (Trusted Computing Group, 2009). A mobile trusted network connection based on tri-element peer authentication and evaluation is shown as Fig. 1.

The entitys in TNC involved:

- **Access Requestor (AR):** The AR is the Mobile Equipment (ME) seeking to access a protected network
- **Policy Decision Point (PDP):** The PDP is the entity performing the decision- making regarding the AR's network access request, in light of the access policies
- **Policy Enforcement Point (PEP):** Also, named access gateway. The PEP is a component that controls access to a protected network. The PEP consults a PDP to determine whether this access should be granted
- **Metadata Access Point (MAP):** The MAP is a component to which other TNC components may publish, subscribe and search data which reflects the state of TNC elements and aids in decision making and policy enforcement

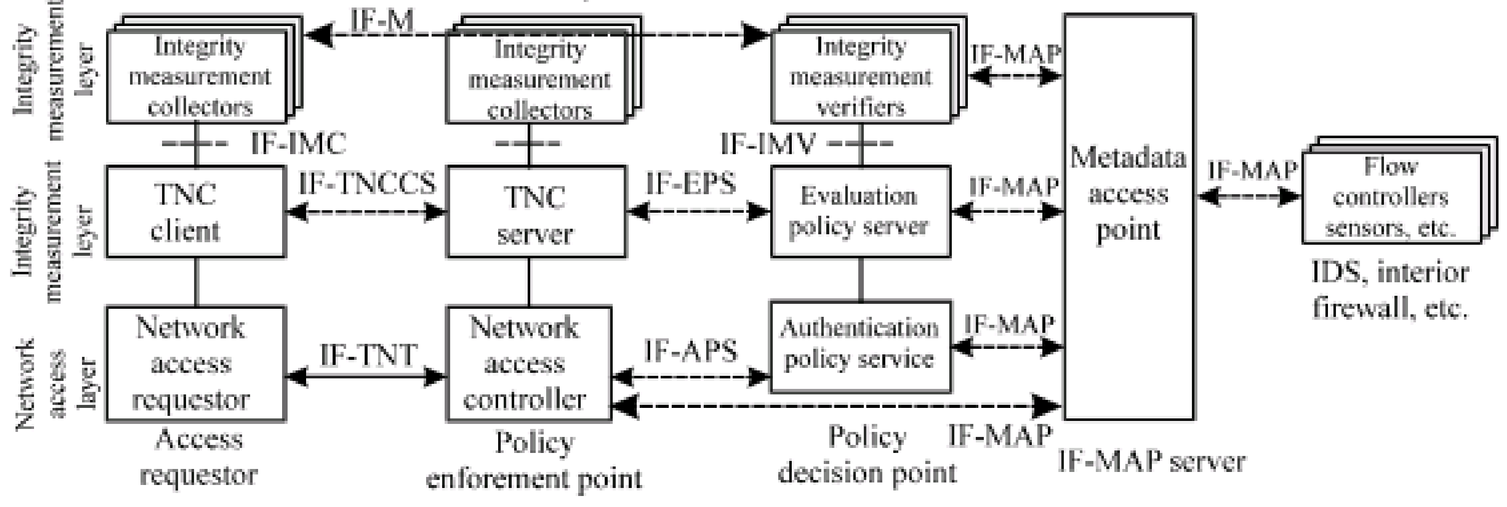

Fig. 1: Trusted network connect based on tri-element peer authentication and evaluation

Corresponding Author: Chaowen Chang, Institute of Computer Science and Technology, Xi'an Jiaotong University, 710049, Xi'an, Shaanxi, People's Republic of China

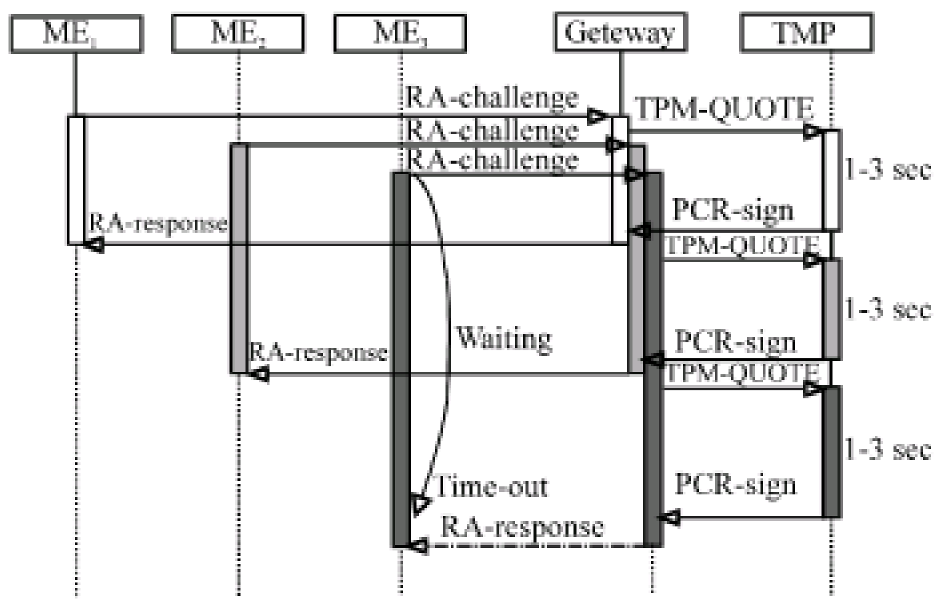

Fig. 2: The attestation request arriving subsequently

- **Flow controllers and sensors:** Flow Controllers are entities that make and enforce decisions about network activities utilizing information from the MAP

The ME seeks to access a protected network which is protected by the access gateway, the gateway decides whether the ME permitted to access the protected network or not, in light of the PDP's decision, the ME decides whether to access the protected network or not also. The decision of the PDP depends on two aspects, the identity authentication and the integrity evaluation of the ME and gateway. In other words, the ME and gateway need to authenticate each other and evaluate the platform integrity of the opposing party.

When the ME and the gateway evaluate platform integrity of each other, they adopt the integrity report protocol in the TCG specification (Trusted Computing Group, 2007). Every ME seeking to access the protected network need validate the gateway's integrity, which needs the integrity report once. When there are large numbers of access request, the integrity reports of gateway will become the system's bottleneck (Stumpf *et al*., 2008), which is shown as Fig. 2.

As is shown in Fig. 2, the mobile entity ME_1 seeks to access the protected network, the first thing needed to do is to check the integrity of gateway, ME_1 sends the remote attestation request to access gateway and the request contains: the set of the Platform Configuration Registers (PCR) sequence numbers (containing one or more sequence numbers), random number (20 Bytes). Access gateway uses the TPM-Quote command to send the PCR sequence numbers and the random numbers to the TPM and the TPM uses AIK to sign, then access gateway returns the PCR value, signature and Stored Measurement Log (SML) as the integrity configure to the ME_1 with the Attestation Identity Key (AIK) certificate. If the aforementioned process doesn't end and at the same time, the access requests of the mobile entities ME_2, ME_3……ME_n arrive early or late, the access gateway's response delay will be longer and longer. There will be a serious situation that without the response arrival of the remote attestation (for example client ME_3 in Fig. 2), the waiting time will be out, which will lead to the failure of the entity's remote attestation.

Table 1: The enforcement time of some TPMs' Quote commands (The Atmel TPMs in T60 and MPC have the different type)

	Infineon	Atmel	Atmel	Broadcom
TPM type	----------------------------(v1.2 TPM)------------------------			
Testing platform	AMD workstation	Lenovo T60	MPC ClientPro	HP dc5750
Average executing time of the quote command	331 msec	382 msec	761 msec	972 msec

Above-mentioned issue is actually about the access gateway's retractility. One of the reasons of the issue is that when designing the TPM, both of the security and low-cost should be considered. In allusion to the TPM's running efficiency, McCune *et al*. (2008) did the contrast testing about the enforcement time of the correlative commands of some TPMs from different manufacturers, the average enforcement time of the TPM-Quote command is shown in Table 1.

The testing shows that the TPM-Quote command can be affected by the AIK signing operation and the running efficiency isn't good. When there are 10 requests arriving at the same time, if only considering the TPM's enforcement time, under the best instance, till the tenth request being dealt with, the time is 3.31 sec and under the worst instance, the time is 9.72 sec (McCune *et al*., 2008). If considering the additional correlative operations and the data traffic, the request waiting time will be longer,which obviously can't be accepted by the moblie trusted access.

RELATED WORK ABOUT THE EFFICIENCY OF INTEGRITY ATTESTATION

The key of the access gateway's retractility in TNC is the efficiency in the public key cryptogram system, the familiar public key cryptogram systems involve RSA pulic key cryptogram arithmetic, ElGamal discrete logarithm public key cryptogram arithmetic and ECC ellipse curve public key cryptogram arithmetic.

Batch RSA arithmetic: Fiat (1997) proposed a batch RSA decoding arithmetic used to improve the RSA's efficiency. The basic idea of Fiat batch RSA arithmetic is:

If e_1, e_2,.......,e_n are n mutual-prime public key and modulus of all of them are N = pq, the n cryptographs earned via encoding the n clear texts m_1,m_2,.....,m_n using the public key e_1, e_2,.......,e_n are c_1, c_2,.......,c_n. Fiat arithmetic can simultaneously decode these cryptographs to earn corresponding clear texts $m_i = c_i^{1/e_i}$ (i = 1,.....,n). The batch process arithmetic (Fiat, 1997) is operated through the full binary tree with n leaves, in which every inner node has two child-nodes and L and R, respectively denote the value of the left and right child-node. The batch arithmetic has three phases to deal with this binary tree: Upward-percolation, Exponentiation-phase and Downward-percolation, then finally obtains every the value of the clear texts.

Shacham and Boneh (2001) improved batch method's performance based on the Chinese Remainder Theorem (CRT), in order to improve the performance of the SSL server. The research results in this literature presented that when the system is using 1024 bits RSA arithmetic, under the instance of n = 4, the server's performance was improved about 2.5 times. Sun *et al*. (2009) proposed Rebalanced RSA arithmetic, which moved the decoding operation load to the encoding side. Qi *et al*. (2005) applied the M/D/1 model to optimizing selecting the queue length in RSA batch arithmetic and gained rather obvious effect.

Resumptively, the batch RSA can improve the public key decoding operation in the server end, but this method can only apply to the RSA arithmetic. But in the public key cryptogram system TPM adopts, adopting RSA, ECC and other non-symmetry cryptogram arithmetic is allowed and the bottleneck of TPM integrity report's efficiency is the issue about the AIK signing speed, thus batch RSA arithmetic can't solve the issue about the integrity report's efficiency well.

Passive attestation based-on trusted third-party: The key idea of the passive attestation based-on trusted third-party is abandoning the way that the client send nonce to make challenge, then TPM respond the challenge and make the TPM signature. The new way is that firstly the Trusted Third Party (TTP) generates the nonce, then initiates the attestation process. Based the idea mentioned earlier, Stumpf *et al*. (2008) proposed two passive attestation methods based-on TTP were presented as followed: Timestamped Hash-Chain Attestation and Tickstamp Attestation.

Timestamped hash-chain attestation: The way of the passive attestation based-on TTP, divides the time into many timeslice, then every timeslice generates only one nonce and timestamp to every server and the server responds this nonce and finishes the integrity report, in this timeslice, in allusion to all access requests to this server, the server returns the same integrity report, then according to the integrity report, the nonce and timestamp of the TTP, the client validate the platform's integrity. The precondition of the method is that the client need keep time synchronization with the TTP. The TTP just generates a nonce and the corresponding timestamp in the beginning phase, then when every new timeslice is arriving, the TTP anymore generates new nonce, instead the server does HASH operation once for the nonce and the result is just the next timeslice's nonce:

$$Na_{v+1} = (\mathrm{HASH})\,(Na_v)\ (v = 0,1,........,k)$$

After the ending of every timeslice, the server computes a new HASH value as the next timeslice's nonce and finishs the integrity report based-on this nonce. When the client sends the challenge request, the server produces a attestation token (τ) to help the client validate the platform configure's freshness.

$$\tau = \{N_{a_0}, \mathrm{time}_0\}_{K_{TTP}^{-1}}, \mathrm{Cert(AIK)}\{h^v(N_{a_0}), v, \mathrm{PCR}, g^x \bmod p\}_{K_{AIK}^{-1}}$$

Attestation token (τ) contains TTP signature's seed fresh data N_{a0}, timestamp Time_0, AIK certificate and the AIK signature's timeslice v, PCR value, this interval's fresh data $h^v(N_{a0})$ and Diffee-Helman key exchange material is useful to build the security chunnel. Attestation token (τ) contains all information used to validate attestation freshness. Considering of the structure of this token, it can be known that this attestation token is not related to the user challenge's fresh data and the server just needs to produce a attestation token at every timeslice, not to do a time-taking TPM signing operation for every client challenge.

The client needs to validate that the platform configure is trusted and the certificate, signature and the timestamp are valid. Furthermore, it should be validated that the nonce received belongs to the current timeslice, detailed validating formula is shown as follows:

$$\mathrm{time}_0 + v \cdot t \le \mathrm{now} \le \mathrm{time}_0 + (v+1) \cdot t + \xi$$

Here, v denotes the sequence number of the current timeslice, t denotes every timeslice's time length (it is invariable), ξ denotes the range of the time error.

If all of the conditions are confirmed, the client will believe that the server is integrated and trusted.

Tickstamp attestation: The tickstamp attestation uses the tick-counter of a TPM. The TPM can provide the current signature of the tick counter. In the TCG specification, the TPM can use a non-migratable key to bind the platform's configure and only if the platform's configure is integrated and untampered, the non-migratable key can work, which the Tickstamp attestation just uses to complete the platform's integrity attestation. The non-migratable key produced by the TPM signs and releases the certificate with the AIK. The non-migratable key is used to produce the TickStampBlobs (contains the current tick value,tick conversation ID and the data signature) in the period time.In order to validate the platform's configure is trusted to the client, it is essential to build a attestation token (τ) in every timeslice:

$$\tau = \{\text{currentTicks}, g^{s} \bmod p\}_{K_{TS}^{-1}}, \text{Cert}(K_{TS}), \text{Cert}(\text{AIK})$$

The attestation token τ contains the current tick value of the non-migratable key K_{TS} signature, Diffee-Helman key exchange material, non-migratable key and the AIK certificate.

What needs to be demonstrated is that, this token can only assert that at that moment of the current tick value, the platform's configure is integrated. But if to finish this whole integrity attestation, what also needs is to guarantee the synchronization between the tick value and the challenger and the TTP is just imported due to the synchronization requirement. The TTP produces a nonce, then the TPM on the server signs this nonce using the command TPM-TickStampBolb and sends the result to the TTP and then the TTP validates the signature and produces a synchronization token containing the world time, then returns the token to the server who adds this information to above attestation token. If the challenger keeps synchronization with TTP through the synchronization protocol, the synchronization between the tick counter and the challenger could be realized and the platform's remote integrity attestation could be fulfilled.

On the basis of mentioned-above, passive attestation based-on the TTP can provide effective method to solve the efficiency about the integrity attestation. But both Timestamped Hash-Chain Attestation and Tickstamp Attestation need to import the TTP and to keep the time synchronization between the mobile entity and the TTP, which not only increases the cost, but also is not suitable for the access scene in the moblie trusted network with the worse circuitry quality.

AN INTEGRITY BATCH REPORT PROTOCOL BASED-ON WAITING STACK

In this study, we propose a integrity batch report protocol based on waiting stack, which expands the TCG integrity protocol. In this protocol,the request about the integrity attestation is placed in order in a waiting stack with variational length and a fixed head, after the TPM has dealt with one set of the integrity report, it can begin to deal with the waiting stack's integrity requests one time together and so recurrence, which can effectively solve the retractility about the access gateway in the trusted network.

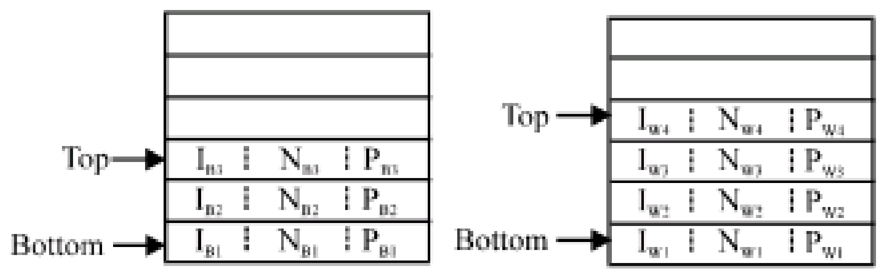
Top
Bottom
I_{B3} N_{B3} P_{B3}
I_{B2} N_{B2} P_{B2}
I_{B1} N_{B1} P_{B1}
Top
Bottom
I_{W4} N_{W4} P_{W4}
I_{W3} N_{W3} P_{W3}
I_{W2} N_{W2} P_{W2}
I_{W1} N_{W1} P_{W1}

Fig. 3: (a) Batch/waiting stack and (b) waiting/batch stack

The structure of the stack: The stacks this protocol involves can be classified into two types, which is shown in Fig. 3.

As is shown in Fig. 3, the waiting stack and batch stack are both first in and last out. Every request has the same data structure. Tri-element tuple I_{BK}, N_{BK}, P_{BK} (K = 1, 2,…,n) respectively denote the entity ID of the batch stack's attestation request with the sequence number k, the nonce and the specific PCR number. Tri-element tuple I_{wk}, N_{wk}, P_{wk} (K = 1, 2,…,n) respectively denote the entity ID of the waiting stack's attestation request with the sequence number k, the nonce and the specific PCR number.

The roles of the waiting stack and batch stack switch constantly. After the TPM finishs dealing with the integrity reports of all requests in the batch stack, the buttom of the batch stack will point to the top of the stack and the batch stack will be empty, at this time the former stack become an empty waiting stack waiting for receiving the next remote attestation request. At the arrival of a new remote attestation request, the entity ID related to this request, the nonce and the specific PCR number will be pushed into the waiting stack, the buttom pointer of the waiting stack adds 1, so recurrence, the length of the waiting stack will increase step by step.

Similarly, After the TPM finishs dealing with the integrity batch reports of all requests in the batch stack, the pointer of the batch stack will point to the former waiting stack and the former waiting stack becomes the batch stack. At the moment, the platform agent will pop out the request information in the stack into the batch buffer to wait for making the integrity reports.

Integrity batch report and the protocol flow

Integrity batch report: The integrity batch report is in allusion to all of the integrity attestation requests in the batch buffer making one report instead of the traditional doing respectively report for every request, thus it can lighten the burden for the TPM to do signing operation, then increases the access gateway's retractility.

Supposing that there are M validating parties requesting the remote attestation, $I_1, I_2, \ldots, I_m$, respectively denote every attestator whose corresponding PCR sequence number set are $P_1, P_2, \ldots, P_m$ and the random numbers are $N_1, N_2, \ldots, N_m$. The corresponding process for integrity batch report is shown as follows:

Assume P is the combining set of all the PCR sequence number sets, viz., $P = P_1 U P_2 U \ldots U P_m$. N is the hash value earned via all the random numbers doing the iterative hash operation, $N = Hash(N_1 || N_2 || \ldots || N_m)$, (In this paper we adopt SHA-1 as the HASH function, the length of the hash value and random number is the same, 20 Byte).

The P is the PCR sequence number set and N is the batch random number, then transfers the command TPM-Quote, after that, the TPM returns the corresponding PCR value and the signature.

Return the PCR value, signature, SML, AIK certificate and all of the random numbers as the information of the integrity report to all of the validating parties.

Remote attestation protocol based-on the integrity batch report: The remote attestation protocol based-on the integrity batch report is shown in Fig. 4.

The flow of the remote attestation protocol based-on the integrity batch report can be divided into seven steps:

- Multi-challengers can subsequently send the remote attestation requests to the attestator and the requests will be pushed into the waiting stack by the platform agent
- After receiving the TPM has finished dealing with the last batch¡±information send by the TPM state monitor, the platform switches the stack and the request information from the challenger will be pushed into the batch stack, then the original batch stack will become to the waiting stack and it can receive the next request
- The platform agent pops the information in the batch stack into the batch buffer, at the same time makes the batch stack empty
- The platform agent does the batch operation to the information waiting for dealing with in the buffer ($P = P_1 U P_2 U \ldots U P_m$, $N = Hash(N_1 || N_2 || \ldots || N_m)$), then sends the command TPM-Quote to the TPM
- TPM signs them with AIK private key and returns the result to the platform agent
- After obtaining the certificate used to validate TPM platform from the repository, the platform agent put the related PCRs' values, TPM signature, SML, the connection bunch of all entities' request nonce

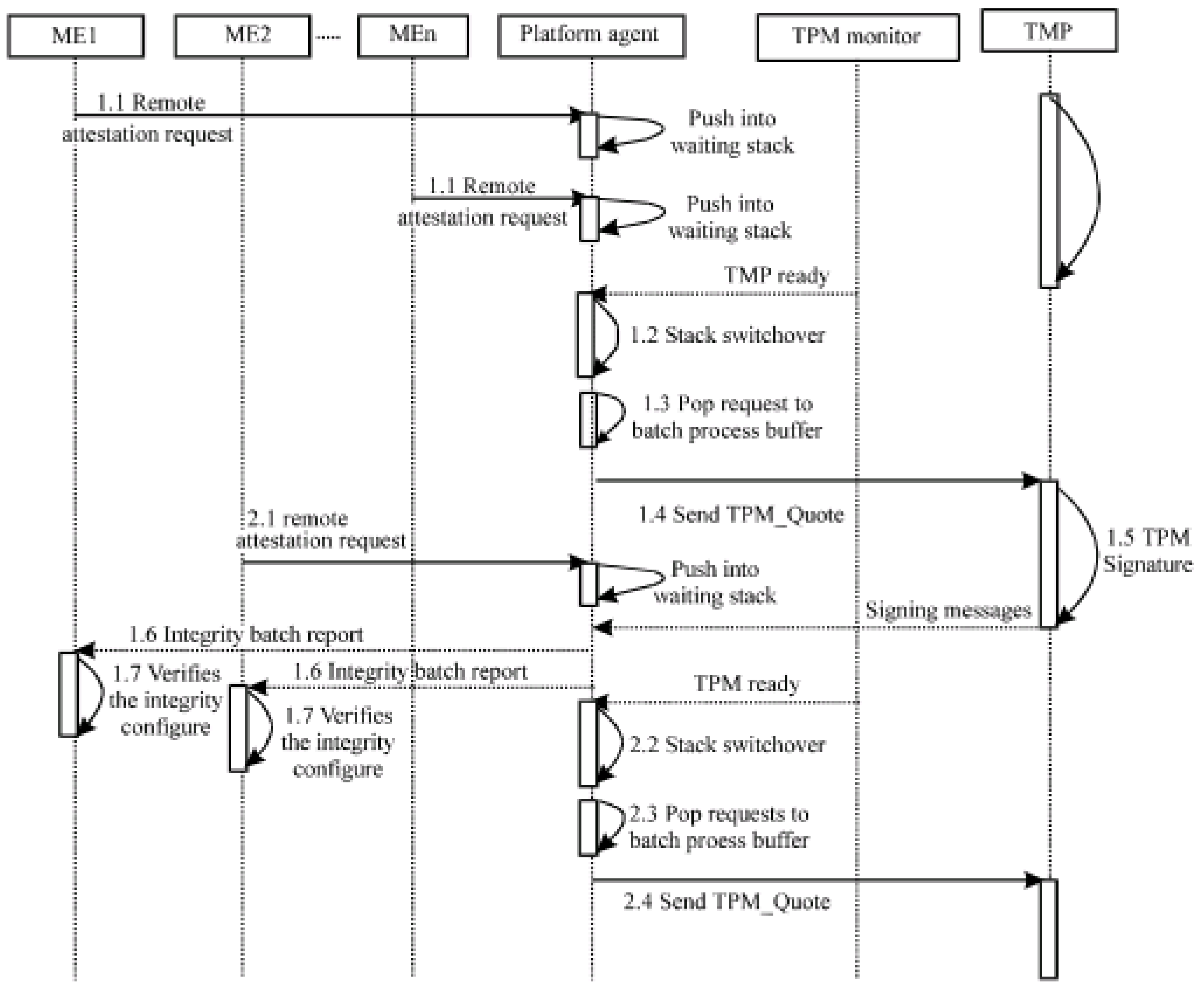
ME1
ME2
MEn
Platform agent
TPM monitor
TMP
1.1 Remote attestation request
Push into waiting stack
1.1 Remote attestation request
Push into waiting stack
TMP ready
1.2 Stack switchover
1.3 Pop request to batch process buffer
1.4 Send TPM_Quote
2.1 remote attestation request
Push into waiting stack
1.5 TPM Signature
Signing messages
1.6 Integrity batch report
1.6 Integrity batch report
1.7 Verifies the integrity configure
1.7 Verifies the integrity configure
TPM ready
2.2 Stack switchover
2.3 Pop requests to batch proess buffer
2.4 Send TPM_Quote

Fig. 4: The flow of the remote attestation protocol based-on the integrity batch report

N_1, N_2,…,N_m and the AIK certificate together and send them respectively to every challenger as the integrity report

- The challenger verifies the integrity configure at local, if the verification fails, the challenger can refuses to communicate with it, or wait for that the attestator is at the security station, then communicate again. Detailed verifying method is as follows:

After the challenger I_k receives the attestator's integrity batch report, it firstly validate the validity of the AIK certificate,then according to the signature of the N,PCRs' value with AIK and the storage log SML, it validates the platform's integrity.

In order to keep the freshness of the report, after receiving the attestator's integrity batch report, the challenger I_k analyzes it to obtain N_1,N_2,…,N_m, then recount the hash value N ' = Hash(N_1‖N_2‖…‖N_m), due to the collision-resisting of the HASH function, when N' = N and $N_k \in \{N_1, N_2, \ldots, N_m\}$,the challenger I_k can be sure that the integrity report the attestator returns is produced after N_k is received, thus to ensure the refreshness.

The traffic growth in data communication induced by protocol extending and the solvement: The traffic growth in data communication induced by protocol extending.

As is discussed earlier,the information returned to the challengers in the integrity report protocol of TCG specification includes: the value of PCR, the signature of TPM, SML, the fresh number requested(nonce) and so on. The information returned to the challengers from the platform basing on remote attestatation protocol of integrity batch report includes: the values of PCR, the TPM signature, SML, the fresh number connection string requested by all terminals ($N_1||N_2||\bullet\bullet\bullet||N_m$) and so on.

Obviously, $N_1||N_2||\bullet\bullet\bullet||N_m$ operation results in traffic growth. Supposing the number of a batch processing request as m, the length of fresh number returned to each terminal with the integrity batch report protocol is m×20 Byte (each nonce is 20 Byte) and the number with the original report protocol is 20 Byte.That is to say that the former is ten times larger than the latter.

Supposing the number of the simultaneous attestation request to Secure Access Gateway is 50 $\sec^{-1}$, after adopting integrity batch report protocol, the communication amount of each terminal is increased by 20 B×(50 -1)≈ 1 kb and it is 1 kb×50 = 50 kb for Access Gateway; If the number of concurrent users is 100, the communication traffic of access gateway is increased to 200 kb, that is to say, the burden of the communication speed is increased by 2 Mbps and for the corresponding mobile terminal, the communication burden is increased by 20 kbps. Obviously, it is accepted hardly for the GPRS mobile communication networks which have dozens of kbps communication bandwidth.

The compression scheme based on Merkle tree: Merkle tree (Merkle, 1987) is a special kind of binary tree. The value of the intermediate nodes in Merkle tree (branch nodes) is a one-way function (HASH function) to the value of the children nodes. Because of the simplicity and multi-purpose of Merkle tree, it is widely used in attestation, key consultment, compression storage and so on.

For a complete Merkle tree, suppose it has 2^H leaf nodes. That is, the depth of the Merkle tree is H and it has 2^H-1 branch nodes. So, the value of each branch node can be expressed as:

$$P(n) = HASH(P(n_{left}) \| P((n_{right})) \text{ (HASH function adopts SHA-1)}$$

In the formula above, n_{left} is the left children nodes of the branch node n, denoted as 0 generally and n_{right} is right children nodes of the branch node n, denoted as 1 generally. Figure 5 sketches a Merkle Tree with a depth of 3.

For a complete Merkle Tree with a depth of 3, according to its leaf nodes' values N_1, N_2,…, N_m(m = 2^H), we can gain the value of the branches nodes h_k (k = 1,2, ..., 2^{H-1}) and for the root node of the Merkle tree, there is h_1 = HASH($N_1||N_2||\bullet\bullet\bullet||N_m$).

For each leaf node N_k(k = 1, 2,…, 2^H) given the brother nodes and the value of H-1 corresponding nodes (that is being given the value of H corresponding nodes of the Merkle Tree), the value of root node h_1 can be obtained. As the Fig. 5 showed above, for the leaf node N_3, only given the values of the netlike nodes (N_4, h_{100}, h_{11}), the value of h_1 can be gotten. Similarly, for the leaf node N_6, just needed to be given solid nodes (N_5,h_{111},h_{10}), the value of h_1 can be calculated, too.

Thus, given m random fresh numbers N_1, N_2,…, N_m (m = 2^H), to calculate HASH value of those m numbers which is added in series (HASH ($N_1||N_2||\bullet\bullet\bullet||N_m$)), need to build a Merkle tree. Only given the values of $\log_2^m$(equal to H) Merkle tree's nodes, we can gain the Hash value of those m numbers which is added in series without knowing all the values of N_1, N_2,…, N_m(m = 2^H). This feature is used to solve the problem of the traffic increasement.

The improvement needed for the content of the integrity batch report is to revise the fresh number connection string ($N_1||N_2||\bullet\bullet\bullet||N_m$) to fresh number compressing connection string of Merkle ($M_1||M_2||\bullet\bullet\bullet||M_H$, H = $\log_2^m$).

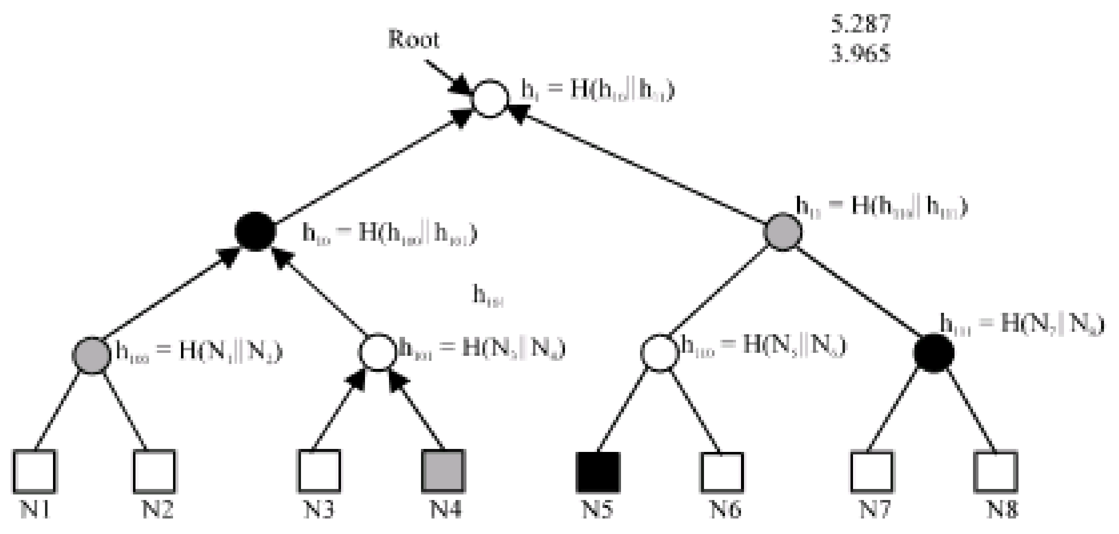

Fig. 5: A Merkle Tree with a depth of 3..

The selection method of $M_1, M_2, \bullet\bullet\bullet M_H$ is as follows:

- Pretreatment. Build a Merkle tree (h_1 is the root of the tree) according to $N_1, N_2, \ldots, N_m$, specify $N_v(v = 1, 2, \ldots, m)$ according to the challenger and let $j = 1$
- Whether N_v is the children node of the current Merkle's root node? If so, then choose N_v 's brother node as M_j and the algorithm ends
- N_v is not the child node of the current Merkle Tree's root node. Choose the left and right children node R_0, R_1 as the root node of the new sub-Tree. Break down the current Merkle Tree to two Merkle Trees: the left tree and the right tree
- Is N_v included in the left tree after being broken down? If so, choose the left tree as the current Merkle Tree, choose R_1 as $M_j$and return to the step 1
- If N_v is not included in the left tree after being broken down, choose the right tree as the current Merkle Tree, choose R_0 as M_j, set $j = j + 1$,and return to the step 1
- The algorithm ends

By compressing the Merkle Tree, we can compress the length of the fresh number connection string tranfered to the challengers from m×20 (Byte) to $\log_2^{m}$×20 (Byte). If m = 1024,the amount of the compressed data is 1024×20-$\log_2^{1024}$ × 20 = 20480 (byte). The amount of the data after being compressed is 10×20 = 200 (byte), which is the same to the amount before compression when m = 10. The effect is obvious after compression.

In order to specify the problem easily, the trees sketched above are complete ones, that is to say that the number of the trees' nodes is integer power of 2. When the condition is not enough, we call it non-ballence Merkle tree as Fig. 6a shows.

The settlement with non-balance Merkle tree is to promote the nodes without brothers to higher level automatically until there is a brother node found. The effect of Merkle Tree after being settled is as Fig. 6b shows.

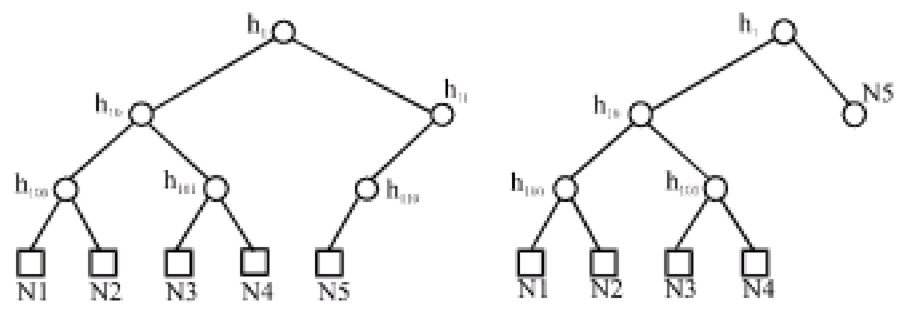

Fig. 6: (a) Non-balance Merkle tree and (b) The settlement with non-balance Merkle tree

PROTOCOL SECURITY ANALYSIS

Man-in-the-Middle attack: There exists Man-in-the-Middle attack in integrity reporting protocol of TCG specification (Goldman *et al*., 2006). This defect exists in integrity batch protocol still. Stumpf *et al*. (2008) had solved this problem by adding Diffee-Helman key exchange protocol and building a secure channel. It is deemed that the defect of Man-in-the-Middle attack requires no special handling, because this defect in Trusted Network Access is non-existent. The concrete reason is that there should be a bidirectional identity authentication between the mobile terminal and the access gateway. Therefore, in the process of the subsequent integrity report, Man-in-the-Middle attack does not exist.

Based on the above considerations, the extended protocol does not increase the protective measures against the Man-in-the-Middle attack, but for the whole access process of trusted network, there will be no attack from Man-in-the-Middle.

Version rollback attack: Integrity batch report protocol is the extension of the TCG integrity report protocol. Supposing that the challengers and the adversaries use integrity report protocol, but the attestators use TCG integrity batch report protocol, adversaries will take advantage of the difference of the versions between the challenger and attestator to do version rollback attack, So, there exists the security drawback in this extended protocol (Stumpf *et al*., 2008).

However, in the specific implementation process, the challenge can find the software configuration of the platform, which is to say the attackers can find the attestator does not use the protocol expected. Therefore, version rollback attack in a real system is impossible to exist.

The security of this protocol: The difference between integrity batch report protocol and the TCG integrity batch report is that the former uses the HASH function of the challenging random number string ($N_1||N_2||\bullet\bullet\bullet||N_m$) from multi-challenge requestors while the latter just uses a random fresh number. The security of the extended protocol depends on the security of Hash function. In this extended protocol,choosing SHA-1 as the HASH value has a good characteristics of collision restraint. It is impossible for enemy attackers to find a collision and to disguise trusted system configuration under current conditions.

It can be said that, under the premise of security of the HASH function, the extended integrity batch protocol has the same security to the TCG integrity report protocol.

PERFORMANCE ANALYSIS OF THE BATCH PROTOCOLANDEXPERIMENTALVERIFICATION

Qualitative analysis: Comparing the integrity batch report protocol proposed in this study with the TCG specification integrity report protocol, some comparison analysis on the processing capacity, the efficiency and the impact on network transmission were down.

The integrity attestation and processing ability

Integrity report protocol with TCG specification: Assume Sign denotes the executing time of TPM_Quote command, Sign generally is 0.3 ~ 1 sec (McCune *et al*., 2008) and the time to complete the integrity report be σ (it is mainly the cost time of the platform agent). Then theoretically, the maximum concurrent number of requests which can be handled in the same time $U_R = 1 / (\text{Sign} + \sigma) \approx 1/\text{Sign}$ (the common hardware supporting platform, s is usually tens of milliseconds of magnitude), U_R is about 1 ~ 3.3 times per second.

Integrity batch report protocol: The main idea of integrity batch report is the one-time signature for all requests in the waiting stack, so its processing capacity and the signature capacity of TPM is weak. Thus, the processing capacity is mainly affected by two aspects: the size of memory resources used as storage in the waiting stack by attestors and the maximum concurrent intercurrent processing capability supported by VPN (Trusted Network Connection is generally based on the secure channel of VPN). Usually, the intercurrent processing capability of VPN gateway is between 32 and 256 times per second, while the memory resources is abundant.Therefore, the processing capability of integrity batch report is the same as the maximum concurrent processing capability of its supporting platform VPN.

The efficiency of remote integrity attestation

Integrity report protocol with TCG specification: The attestation efficiency is closely related to the number of requests in a period. When there is only one request, the responding time of remote attestation is the same as the executing time of TPM_Quote command approximately and the time is about 0.3~1 sec generally (McCune *et al*., 2008). With the increasement of requesting users in the same time, the efficiency is becoming lower and lower. Suppose the average number of requests in time t is Q,thus, the average responding time is:

$$(1\times \text{Sign}+2\times\text{Sign} + \& +Q\times \text{Sign})/Q=(Q+1)\times Q\times\text{Sign}/2Q= (Q+1)\times\text{Sign}/2$$

Seriously, if the maximum tolerant waiting time of challenge attestors is T and the number of arriving requests in time t in the same time satisfies $(Q/T)> (1/\text{Sign})$, there will be no respondency to the latter arriving requests and the efficiency will deteriorate sharply.

Integrity batch report protocol: Suppose the summation of the stack conversion time and platform agent's pretreatment time is denoted as T_P. The attesting efficiency based on integrity batch report is without relation to the number of requests in a time. No matter how many users arriving at the same time (the maximum dose not exceed the limit of safe channel VPN's concurrent user), the average responding time is T_P + Sign.

If take use of the compression based on Merkle, the average responding time will be T_P + Sign + T_m (T_m is the time of building a Merkle Tree and the choosing time of Merkle Tree's nodes aiming at each challenger).

Aiming at the efficiency of Merkle tree, Williams and Sirer (2004) proposed the linear equation in the run time of SHA-1 function on the computer of Intel Pentium 4 CPU 1700 MHz: $T_H = \alpha b + \beta$, when b is the amount of bytes of data blocks, $\alpha = 0.0122348$ μ sec/B and $\beta = 1$ μsec. Basing on this formula, when the height of this Merkle tree is 10 (i.e., the length of the stack is $2^{10} = 1024$), the time of building this Merkle tree is 1.52 msec. Generally, $m < 1024$, that is to say, the time of building a Merkle tree is <1.52 m sec usually. This shows that Tm plays little influence on the responding time.

The impact on the amount of the information transmitted: From the contents of the report we can gain the amount of the information transmitted to the challengers from the TCG specification integrity report protocol.

After the adoption of integrity batch report protocol,the amount of the information transmitted to each challenger is added by (m -1)×20 bytes. After Merkle compression, the amount of the information transmitted to each challenger from the improved integrity batch report protocol is added by ($\log_2^m$ -1)×20 bytes.

Experimental verification: In order to verify the protocol's effect, this paper is based on XINDA JIEAN company's SQY42 Mobile Security Access Gateway (VPN concurrent users number is 128) and homemade TPM Trusted Module (based on ZTE Microelectronics's Z8D64U password chips, the rate of RSA signature is about 3.2 seconds per time). For the TCG specification integrity report protocol and the integrity batch report protocol,we have carried out some analysis and comparative experiments on the concurrent processing capability and the responding time.

The experiment chooses SQY42 Access Gateway as the integrity report testifier and chose ThinkPad X61 as the challenger. It was divided into 9 groups and the duration of each group was 10 sec. In those 9 groups of experiments, the numbers of remote attestation requests from the challengers were respectively 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 and the time of the challenge requests' arriving at the attestor was of Poisson distribution. Each group of experiments was repeated 10 times with 5 groups adopting TCG integrity report protocol and 5 groups adopting integrity batch report protocol. We classified, recorded and averaged t as the measured values. The results are as follows:

Process capacity: The results of experimental Processing capability are shown in Table 2.

Table 2: The results of the request process capacity in the experiment (unit:times)

Integrity report	Request times 16	32	64	128	256	512	1024	2048	4096
TCG									
Success	16	31.6	31.8	31.8	31.4	31.2	31.4	31.6	31.2
Failure		0.4	32.6	96.2	224.2	480.8	992.6	2016.4	4064.8
TCG									
Success	16	32.0	64.0	128.0	256.0	512.0	1024.0	1277.0	1282.0
Failure								771.0	2814.0

Responding time: As can be seen from the Table 2,when the number of requests exceeds 32, if adopting TCG integrity report protocol, it will lead to the result that the majority of the requests will not be responded to, thus, calculating average responding time in this condition is useless. Taking this into consideration,in this comparative experiment,the numbers of the remote attestation are 4, 8, 16, 32, 64, 128, 256, 512, 1024 separately,and other conditions and testing method are as above.

The results of the responding time in this comparative experiment are shown in Table 3.

RESULTS AND DISCUSSION

The ME seeks to access a protected network which is protected by the access gateway, They adopt the integrity report protocol in the TCG specification (Trusted Computing Group, 2007) to evaluate platform integrity of each other. When there are large numbers of access request, the integrity reports of gateway will become the system's bottleneck (Stumpf *et al*., 2008) and this issue also shown by Table 1 (McCune *et al*., 2008).

Aiming this issue, some batch RSA arithmetic were proposed (Fiat, 1997; Sun *et al*., 2009; Shacham and Boneh, 2001) proposed, But batch RSA arithmetic can't solve the issue about the integrity report's efficiency well. Stumpf *et al.* (2008) proposed two passive attestation methods based-on TTP to solve this problem, which can solve the efficiency about the integrity attestation effectively. But both Timestamped Hash-Chain Attestation and Tickstamp Attestation (Stumpf *et al*., 2008) need to import the TTP and to keep the time synchronization between the mobile entity and the TTP, which not only increases the cost,but also is not suitable for the access scene in the moblie trusted network with the worse circuitry quality.

Aiming at the bottleneck problem of the TPM signature in integrity report, we propose an integrity batch report protocol based on the waiting stack. By qualitative analysis mentioned above, the responding capacity of gateway adopting integrity batch report protocol is prior to adopting TCG integrity report protocol and the processing capacity of gateway can be improved from 1~3.3 to 32~256 times per second. After Merkle compression, The length of fresh number connection string transferred to the challenger can be compressed from m × 20 bytes to $\log_2^m$ × 20 bytes.

Laboratory result shown that the responding capacity to requests of integrity batch report protocol is prior to TCG integrity report protocol. As can be seen from the Table 2, when request times reached 64 within 10 sec, half requests can not get response from gateway when adopting TCG integrity report protocol, meanwhile the

Table 3: The results of the responding time in this comparative experiment

Batch and protocol	Request times								
	4	8	16	32	64	128	256	512	1024
TCG integrity report	0.35	0.36	0.42	0.6	---	---	---	---	---
Integrity batch report protocol	0.50	0.49	0.52	0.51	0.51	0.52	0.54	0.55	0.57

gateway can response all request well when adopting integrity batch report protocol. Table 2 also shown that when request times is under 1280 within 10 sec, gateway can work well when adopting integrity batch report protocol. It can be got based on Table 2, retractility of the access gateway can be improved adopting integrity batch report protocol.

As can be seen from the Table 3, when there are few requests, the responding time of TCG integrity report protocol is shorter than that of integrity batch report protocol, which is as long as the time of TPM signature (the time of TPM signature in this experiment is 0.32 sec). With the increasement of the number of the requests, when adopting TCG integrity report protocol, the attesting efficiency in the same time is becoming worse and worse. In the period of time(in this experiment, it is 10 sec), when the average arriving number of request exceeds 3.2, the efficiency deteriorates sharply. At same time, The responding time of integrity batch report is about 0.5 sec and the amount of concurrent requests effects it very little.

CONCLUSION

Aiming at the bottleneck problem of the TPM signature in integrity report, we propose an integrity batch report protocol based on the waiting stack. This protocol places integrity attestation requests sequentially in a waiting stack with constant head part and varying length. After dealing with the former integrity reports, then TPM does a batch processing with the integrity reports in waiting stack once only. In this way, the problem of retractility effectly for trusted network access gateway could be resolved. In order to reduce the communication traffic in processing integrity batch reports, a data compressing method based on the Merkle tree is proposed. The length of fresh number connection string transferred to the challenger can be compressed from m ×20 bytes to $\log_2^m$ ×20 bytes. The laboratory result indicates that the responding capacity of gateway adopting integrity batch report protocol is prior to adopting TCG integrity report protocol and the processing capacity of gateway can be improved from 1~3.3 to 32~256 times per second.

ACKNOWLEDGMENTS

This study was supported by the National High Technology Research and Development Program of China (863 Program) under grant No. 2007AA01Z479.

REFERENCES

Fiat, A., 1997. Batch rsa. J. Cryptol., 10: 75-88.

Goldman, K., R. Perez and R. Sailer, 2006. Linking remote attestation to secure tunnel endpoints. Proceedings of the 1st ACM Workshop on Scalable Trusted Computing, Nov. 3, Virginia, USA., pp: 21-24.

McCune, J.M., B. Parno, A. Perrig, M.K. Reiter and A. Seshadri, 2008. How Low Can You Go? Recommendations for Hardware-Supported Minimal TCB Code Execution. ACM Press, Washington, USA., pp: 14-25.

Merkle, R.C., 1987. A Digital Signature Based on a Conventional Encryption Function. In: Advances in Cryptology-CRYPTO 87, Pomerance, C. (Ed.)., LNCS. 293, Springer-Verlag, Berlin, pp: 369-378.

Qi, F., W.J. Jia, F. Bao and Y. Wu, 2005. Batching SSL/TLS Handshake Improved. In: Information and Communications Security, Qing, S. *et al*. (Eds.). Springer, Berlin, Heidelberg, pp: 402–413.

Shacham, H. and D. Boneh, 2001. Improving ssl handshake performance via batching. Proceedings of the 2001 Conference on Topics in Cryptology, (CTC'01), Springer-Verlag, London, UK., pp: 28-43.

Stumpf, F., A. Fuchs and S. Katzenbeisser, 2008. Improving the scalability of platform attestation. Proceedings of the 3rd ACM Workshop on *Scalable* Trusted Computing, Oct. 31, Virginia, USA., pp: 1-10.

Sun, H.M., M.E. Wu, M.J. Hinek, C.T. Yang and V.S. Tseng, 2009. Trading decryption for speeding encryption in Rebalanced-RSA. J. Syst. Software, 82: 1503-1512.

Trusted Computing Group, 2007. TCG specification architecture overview. http://www. trustedcomputing group.org/files/resource_files/AC652DE1-1D09-3519-ADA026A0C05CFAC2/TCG_1_4_Architecture_Over view pdf.

Trusted Computing Group, 2009. TNC architecture for interoperability. http://www. trustedcomputinggroup. org/files/resource_files/51F9691E-1D09-3519-AD1C1E27D285F03B/ TNC_Architecture_v1_4_r4 .pdf.

Williams, D. and E.G. Sirer, 2004. Optimal parameter selection for efficient memory integrity verification using merkle hash trees. Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications, Aug. 30-Sept. 01, IEEE Press, Washington DC. USA., pp: 383-388.