Client Oriented Remote Attestation Model in Cloud Environment

Liu Zhenpeng\textsuperscript{1,a}, Wang Xu\textsuperscript{1,b}, Liu Yifan\textsuperscript{2}, Guo Ding\textsuperscript{1}, Zhu Xianchao\textsuperscript{1}

\textsuperscript{1}College of Information and Electrical Engineering, Hebei University, Baoding Hebei 071000, China

\textsuperscript{2}School of Computer Science and Technology, China University of Mining and Technology, Xuzhou Jiangsu 221008, China

\textsuperscript{a}lzp@hbu.edu.cn
\textsuperscript{b}freeman200608@126.com

Abstract

In the field of cloud security, the cloud provider don’t disclose any internal configuration information to protect itself, so the client know nothing about their data stored in the cloud and security status of the node providing services for them, thereby it causes the client’s worry whether to adopt cloud computing services. So that the trust between client and cloud computing provider become one of the biggest obstacles hindering the development of the cloud computing. Based on Direct Anonymous Attestation (DAA) and Dynamic Property Trusted Attestation (DPTA), we propose a client oriented remote attestation (CORA) model in cloud environment, client can select a node in the cloud at corresponding security level according to their own needs and can dynamically verify the node’s security status. At the same time, because the use of anonymous method it will not expose classified information of the node, such as configuration and location information etc. Furthermore we add service life of certificates to update certificates regularly, which enhanced the security of the attestation.

Keywords: cloud security, remote attestation, client oriented

1. Introduction

With the advantages of on-demand self-service, ubiquitous network access, physical location transparency, rapid scalability and pay-for-use, Cloud computing sets off a new wave of information tide and is widely used in many fields of society. For the enterprise users, it can significantly reduce the storage and maintenance cost of information. For the personal users, it will relieve the restraint of personal users’ limited storage and computing resources, if the storage and computing of information is put in the cloud [1]. Although cloud computing has great features, but trust and security problem has become the bottleneck for its development.

Cloud service provider rarely disclose its internal configuration as to ensure its own safety, so client do not know the security situation of its data and the node providing services, which consequently causes the client worry about adopting cloud computing services. Thus trust problem has become one of the biggest obstacles in cloud computing’ development. To solve this problem, trusted computing technology proposed by TCG (Trusted Computing Group) has integrated into cloud computing environment reliably and been the focus in cloud security field [2]. Trusted computing is based on Trusted Platform Module, through the trusted measurement of software and hardware to complete the extension of trust chain and to ensure the system boot credibly. The verifier can judge the trusted state of platform by remote attestation. Based on Direct Anonymous Attestation [3] (DAA) and Dynamic Property Trusted Attestation [4] (DPTA), we propose a client oriented remote attestation (hereafter referred to as CORA) model in cloud environment, client can select a node in the cloud at corresponding security grade.
according to their own needs and can dynamically verify the node’s security status. At the same time, because the use of anonymous method it will not expose classified information of the node, such as configuration and location information etc. Furthermore we add service life of certificates to update certificates regularly, which enhanced the security of the attestation.

2. Related Works

By far, there are two main schemes proposed by TCG, one is Privacy CA (Privacy Certification Authority) scheme in TPM 1.1 standard, another is DAA (Direct Anonymous Attestation) scheme in TPM 1.2 standard. Although DAA has solved the CA bottleneck problem in Privacy CA, but they are all binary attestation scheme, which will easily disclose the system configuration of platform leading to threat to platform security.

To prevent the exposure of platform’s privacy, Poritz et al. [5] propose PBA (Property-Based Attestation) from system architecture level. Sadeghi et al. [6] propose the concrete realization progress of PBA. Chen et al. [7] propose the protocol of PBA. Qin Yu et al. [8] propose a scheme called Component property base remote attestation, component as the granularity, every component’s security attribute can be attested, which has better flexibility and expandability. Yan Jianhong [4] proposes DPTA(Dynamic Property Trusted Attestation) method based on PBA. The verifier analogously compute PCR value and compare with the PCR value in the certificate, so attestor can prove that the attested platform satisfies a certain security attribute. The experiment shows that this method can protect the platform’s privacy and overcome the PBA’s static feature, so it has both real-time and security feature.

Under the cloud environment, Xin et al. [9] put the property-based attestation mechanism into cloud environment, based on the character in the cloud, proposed property-based remote attestation oriented to cloud computing. Client can verify node attributes effectively without disclosing the node’s information. Xu [10] et al. propose real-time remote attestation based on IaaS cloud, which uses virtualization and trusted computing to verify whether the behavior of software is trusted in real time. Zhang Yan et al. [11] propose a scheme based on virtual machine identity attestation, under the original authentication and trust relationship, concealing the authoritative identity information in the process of identity attestation, avoided disclosing the organization structure and location at el. information and supporting cloud environment’s transparent structure and location-independent feature.

3. The Overall Structure of CORA Model

The overall structure is as shown in Figure 1. Our model is based on Eucalyptus architecture. The Eucalyptus architecture mainly include 3 roles: common cloud node, called Node Controller (NC), accept responsibility for booting, sleeping, closing and migrating VM. Cluster control node, called Cluster Controller (CC), simultaneously connect server between intranet and extranet and manage several NCs. Cloud control node, called Cloud Controller (CLC), which is the entrance for client and the global decision analysis component, accept responsibility for managing the request of client and administrator, making high level VM dispatch decision, disposing service level agreement and maintaining enduring metadata of system and client.
In this model we add a trusted third party called Certificate Issuer (CI). CI is responsible for issuing DAA certificate and security level attribute (hereafter referred to as SLA) certificate to NC and communicating with CLC in security. In addition CI maintain a security level attribute configuration database, NC use its trusted platform module to measure its configuration, then CI map the measurement data sent by NC into corresponding security level and issue SLA certificate to NC. In CLC, after NC get the security level attribute certificate, CLC verifies the certificate and whether the node is in trusted state, then create a security level list of all NC. The list will be updated at fixed period. When client submit applications, CLC select a NC in the list according to the security level requested by client, then client verifies NC’s SLA certificate to decide whether to adopt the service provided by the NC.

4. The Protocol of CORA Model

This model is based on DAA and DPTA, we add service life of certificates on the basis of DPTA, the setting of service life of certificates is according to the different cloud platform’s character, so that we can update security level list in the service life of certificates. Security parameters used in the protocol is shown in the TABLE I.
Table 1. Security Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Length</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(l_n)</td>
<td>2048</td>
<td>The length of RSA key</td>
</tr>
<tr>
<td>(l_f)</td>
<td>104</td>
<td>The length of secret message (f_0, f_1)</td>
</tr>
<tr>
<td>(l_{\delta})</td>
<td>80</td>
<td>The length of security parameter in Zero-knowledge proof</td>
</tr>
<tr>
<td>(l_v)</td>
<td>160</td>
<td>The length of security level attribute configuration (sv)</td>
</tr>
<tr>
<td>(l_{cv})</td>
<td>160</td>
<td>The length of security configuration (cv)</td>
</tr>
<tr>
<td>(l_H)</td>
<td>160</td>
<td>The length of hash function’s output in Fiat-Shamir heuristic method</td>
</tr>
<tr>
<td>(l_e)</td>
<td>368</td>
<td>The length of index (e) in the certificate</td>
</tr>
<tr>
<td>(l_i)</td>
<td>120</td>
<td>The length of interval for selecting (e)</td>
</tr>
<tr>
<td>(l_r)</td>
<td>2536</td>
<td>The length of random number (v) in the certificate</td>
</tr>
<tr>
<td>(l_p)</td>
<td>80</td>
<td>Safety parameter needed in reducing trusted attestation</td>
</tr>
<tr>
<td>(l_q)</td>
<td>1632</td>
<td>The length of coefficient (p) in Pedersen Commitment</td>
</tr>
<tr>
<td>(l_{\text{exdate}})</td>
<td>160</td>
<td>The length of service life of certificate (\text{exdate})</td>
</tr>
</tbody>
</table>

\(l_{\text{exdate}}\) is the length of the added parameter service life of certificates (represented by \(\text{exdate}\)).

Moreover: \(l_e, l_i\) in DAA satisfy:

\[
l_e > l_{\delta} + l_p + \max(l_f + 4, l_f + 2), l_e > l_n + l_H + \max(l_f + l_r + 3, l_{\delta} + 2)
\]

\(l_e, l_i\) in security level attribute certificate satisfy:

\[
l_e \geq \max(l_v, l_{cv} + 2), l_i \geq l_n + \max(l_v, l_{cv}) + l_{\delta}, \text{ and } l_q > l_{cv} + l_H + l_{\delta} + 2. l_q\) represents the length of subgroup order \(Q\) of \(Z_p^*\) and \(P, Q\) satisfy: \(2^{l_q} > Q > 2^{l_q-1}, 2^{l_p} > Q > 2^{l_p-1}\).

Our protocol adopts Pedersen Commitment[12], CL signature scheme[13], Zero-knowledge proof and DAA scheme [3]. Pedersen Commitment is used to conceal and binding NC’s configuration. CL signature scheme group signature policy, it permit efficient protocol to sign knowledge proof and restore secret information of the signature effectively according to discrete logarithm based on knowledge proof. Zero-knowledge proof is a method by which attester can prove to verifier that a given statement is true, without conveying any information apart from the fact that the statement is indeed true. DAA scheme is based on CL signature scheme and Zero-knowledge proof, furthermore it will use Fiat-Shamir heuristic method to transform knowledge proof to non-interactive signature of Knowledge.

4.1 Issuing DAA Certificate and SLA Certificate Protocol

All NCs in the cloud platform already acquire the public key published by CI \(PK_i = (n, g, h, S, Z, R, R, R, R, \Gamma, \rho)\). We add two parameters \(R, R\) different with DAA’s \(PK_i\) which are used in SLA certificate. Generation method is same with \(R, R\).

i) NC verifies \(PK_i\) in the same way with DAA.
ii) NC makes a request to CI.
iii) CI generates a random number \(n\) and send to NC to prevent replay attack.
iv) NC sends \([U, N, n, \{cv, sml, ML_{NC}\}_{SK_{EK}}, H(PK_{EK})\}_{PK_i}\) to CI. \(cv\) is the value of platform security configuration, \(sml\) is platform storage measurement log, \(ML_{NC}\) is platform integrity measurement list, \(SK_{EK}\) is private key of endorsement key, \(PK_{EK}\) is public key of endorsement key, \(H(PK_{EK})\) is hash value of \(PK_{EK}\). Specific process is as follows:
a) NC’s TPM generates a secret message \( f \) and a random integer \( v \ (v \in \{0,1\}^{e_{|f|}}) \) and splits \( f \) into two messages \( f_0 = \text{LSB}_f(f), f_1 = \text{CAR}_f(f) \). Compute:
\[
\zeta_j = (H_j(1||bsn_j))^{(1/v_{\rho})} \mod \Gamma, U = R_0^{f_0} R_1^f S^v \mod n. N_j = (\zeta_j^{f_0} f_1)^{2/v} \mod \Gamma, bsn_j \text{ is the basename of } CI.
\]

b) TPM executes Quote command and then gets \( ML_{NC}, sml \). After that, encrypt \( ML_{NC}, cv \) and \( sml \) with \( SK_{EK} \), encrypt \( H(PK_{EK}), \{cv, sml, ML_{NC}\}_{SK_{isv}} \) with \( PK_i \).

c) At last, NC sends \( \{U, N, n, \{cv, sml, ML_{NC}\}_{SK_{isv}}, H(PK_{EK})\}_{PK_i} \) to CI.

v) CI judges whether NC is the black list by calculating \( N_j = (\zeta_j^{f_0} f_1)^{2/v} \mod \Gamma. f_0, f_1 \) are leaked secret message.

vi) CI verifies random number \( n \). Then CI decrypt \( \{cv, sml, ML_{NC}\}_{SK_{isv}}, H(PK_{EK})\) with \( SK_i \). Verify whether NC’s TPM is the valid TPM by compare \( H(PK_{EK}) \) to the hash value of EK public key list. Then get \( cv_i, sml, ML_{NC} \) with valid \( PK_{EK} \). Use \( sml \) recalculate platform integrity measurement \( ML_{NC} \), compare \( ML_{NC} \) with \( ML_{NC} \). At last verify whether NC is at trusted state with \( ML_{NC} \).

vii) CI verifies \( U \) and \( N_j \) through Zero-knowledge proof.

viii) CI generates DAA certificate. Generate a random integer \( \hat{v} \) and a random prime \( e \) and compute:
\[
\hat{v} = v + 2^{e_{|v|}}
\]
\[
A = (\frac{Z}{US^v})^{\hat{v}} \mod n
\]

ix) CI generates SLA certificate. Compare \( cv_i \) with security level attribute configuration database, get the corresponding value of security level attribute configuration \( sv_i \). Generate random primes \( e_i, v_i \), their length require that:
\[
l_i \geq \max\{l_v, l_{cv}, ML_{NC}\} l_i \geq l_i + \max\{l_v, l_{cv}, ML_{NC}\} + 1
\]
Compute:
\[
A_i = (\frac{Z}{UR_0^{l_i} R_2^{l_{sv_i}} R_3^{l_{exdate}} S^{v_i}})^{1/v} \mod n
\]
\( exdate \) is the service life of certificate, \( exdate \in \{0,1\}^{l_i} \).

x) CI encrypts DAA certificate, SLA certificate, \( sv_i \) and \( exdate \) with \( PK_{EX} \) then send \( \{A, e, v, \{A, e, v, sv_i, exdate\}_{PK_{EX}}\} \) to NC.

4.2 SLA Signing Protocol

The signature procedure of SLA certificate is between TPM and host. TPM input \( SK_{EK} \) host input \( \{A, e, v, sv_i\} \). Common input: \( PK_i = (n, g, g, h, S, Z, R_0, R_1, R_2, R_3, r, \Gamma, \rho) \), commitment parameter: \( \text{par}_{com} = (g, h, P, Q), cv, ML_{NC} \), \( sv, exdate, n \) (\( n \) is provided by the verifier).

i) The TPM performs as follows:

a) Generate a random \( N_i \in \{0,1\}^{l_{|N|}} \).

b) Generate a random \( r \in \{0,1\}^{l_{|r|}} \) and compute the commitment \( C_{sv_i} = g^{cv_i} h^{r} \mod P \).

c) Generate a TPM signature
\[
\sigma_M \Rightarrow \text{Sign}_M(SK_{SK}, \text{par}_{com} || C_{sv_i} || ML_{NC} || exdate || n, || N_i)
\]
Send to host $\sigma_M$ with the values $C_{sv}$, $ML_{NC}$, $N_j$ and exdate.

ii) The host performs as follows:

a) Generate a random $\omega \in [0,1)^{\ell_1}$.

b) Compute: $\hat{A} = A S^\omega \mod n$, $\hat{v} = v_i - \omega e_i$.

c) Create a masked signature: $\hat{\sigma}_{CL} \Rightarrow (\hat{A}, e_i, \hat{v})$.

iii) The host computes the signature of knowledge protocol:

\[
SPK[(c_{vi}, ML_{NC}, e_i, \hat{v}, r); Z = \frac{Z}{R_{\hat{r}}} \mod n]
\]

\[
\mod P \\{c_{vi}, ML_{NC}, \} \\in [0,1]^{\ell_1 + \ell_2 + \ell_3 + 1} \land (e_i - 2^{\ell_1}) \in [0,1]^{\ell_1 + \ell_2 + \ell_3 + 1} \}(n_j, N_j)
\]

steps as follow:

a) Compute $Z = \frac{Z}{R_{\hat{r}}} \mod n$.

b) Generate random integers:

\[
r_i \in_R [0,1]^{\ell_1 + \ell_2 + \ell_3 + 1}, r_c \in_R [0,1]^{\ell_1 + \ell_2 + \ell_3 + 1},
\]

\[
r_{ML_{NC}} \in_R [0,1]^{\ell_1 + \ell_2 + \ell_3 + 1}, r_{exdate} \in_R [0,1]^{\ell_1 + \ell_2 + \ell_3 + 1}.
\]

c) Compute: $\hat{Z} = A^{\omega} R_0^{(e_i - 2^{\ell_1})} R_{ML_{NC}}^{exdate} S^{\hat{v}} \mod n$

\[
\hat{C}_i = g^{\omega \cdot h^i} \mod P.
\]

\[
c = Hash(PK_i || par || sv || C_{vi} || ML_{NC} || exdate || \hat{Z} || \hat{C}_i || n_j || N_j)
\]

\[
C_{sv} = r_i^{\omega \cdot \hat{v}} \cdot s_c
\]

\[
s_i = r_i^{\omega \cdot \hat{e}_i} \cdot s_c + c \cdot e_i v_i
\]

\[
s_c = r_c^{\omega \cdot \hat{v}} \cdot s_c + c \cdot \hat{v}
\]

\[
s_{ML_{NC}} = r_{ML_{NC}}^{\omega} + c \cdot ML_{NC} \cdot s_{exdate} = r_{exdate} + c \cdot exdate
\]

Generate the signature of SLA certificate:

\[
\sigma_{MPBA} \Rightarrow (\hat{A}, \sigma_M, N_j, C_{sv}, ML_{NC}, exdate, c, s_i, s_c, s_c, s_r, s_r, s_{ML_{NC}}, s_{exdate})
\]

4.3 The Verification Protocol

Verifier acquires $PK_i$ published by CI and $PK_{AK}$, $par_{com}$, $\sigma_{MPBA}$, DAA certificate, $\{PCRs\}_{PK_{AK}}$, $sm_{i}$, $n_j$ sent by NC. The verifier performing as follows:

i) Verify $\sigma_M$ with $PK_{AK}$:

\[
(par_{com} \| C_{sv} \| ML_{NC} \| exdate \| n_j \| N_j) \quad \text{and verify whether the exdate is exceed the time limit. If positive go to the next step.}
\]

ii) Compute:

\[
\hat{Z} = Z^{\omega \cdot \hat{A}} R_0^{\omega \cdot \hat{e}_i} R_{ML_{NC}}^{\omega \cdot \hat{v}} S^{\hat{v}} \mod n
\]

\[
\hat{C}_i = C_{sv} g^{\omega \cdot h^i} \mod P.
\]

iii) Verify:

\[
c = Hash(PK_i \| par \| sv \| C_{sv} \| ML_{NC} \| exdate \| \hat{Z} \| \hat{C}_i \| n_j \| N_j)
\]

\[
s_{ML_{NC}} \in [0,1]^{\omega \cdot \hat{v} + \ell_2 + \ell_3 + 1}, s_{exdate} \in [0,1]^{\ell_1 + \ell_2 + \ell_3 + 1,}
\]

\[10]
\]
\[ s, s_{\text{end}} \in \{0,1\}^{t_{\text{enc}}+t_{\text{dec}}+1}, \quad PCR = ML_{NC} \]

### 4.4 Registration Protocol

NC registers to CLC. NC prove its security attribute level to CLC, CLC add NC into its security attribute level list. As follows:

i) NC sends request to CLC. Encrypt \( e \), NC’s id with \( \text{PK}_{\text{CLC}} \), then send them to NC.

ii) After CLC receives request, generates a random \( n \), encrypts \( n \) with \( \text{SK}_{\text{CLC}} \), then sends to NC.

iii) After the decryption, NC get \( n \) and generates a attestation identity key (AIK), signs (public key of AIK) with DAA certificate. Execute Quote command to get PCRs (the value of real-time \( ML_{NC} \)). Encrypt PCRs, SLA certificate with \( \text{SK}_{\text{AIK}} \) (private key of AIK), and encrypt DAA certificate with \( \text{PK}_{\text{CLC}} \), send them to CLC.

iv) CLC verifies DAA certificate, acquire \( \text{AIK}_{\text{PK}} \). Then get PCRs and SLA certificate using \( \text{PK}_{\text{AIK}} \). Execute verification protocol in C and compare \( ML_{NC} \) with PCRs, if they have equal value, then the NC is at trusted state now. At last CLC add NC into security attribute level list.

### 4.5 Request and Verification Protocol for Client

When client requires cloud service, client submits a service request to CLC and verifies NC’s security attribute level and trusted state. Specific process is as follows:

i) Client requests a NC to CLC, which security attribute level is \( s_{\text{v}} \).

ii) CLC chooses a NC in the security attribute level list, which security attribute level is \( s_{\text{v}} \).

iii) NC generates an AIK, sign \( \text{PK}_{\text{AIK}} \) with DAA certificate, then sends it to the client.

iv) Client verifies DAA certificate and gets \( \text{PK}_{\text{AIK}} \). Generate a random number, encrypt it with \( \text{PK}_{\text{AIK}} \) and send to NC.

v) After NC gets \( \{ n \}_{\text{pk}_{\text{AIK}}} \), decrypt it with \( \text{SK}_{\text{AIK}} \). Execute Quote command to get the real-time PCRs. Encrypt SLA certificate and PCRs with \( \text{SK}_{\text{AIK}} \) and send to client.

vi) Client executes verification protocol. Verify whether NC’s security attribute level is \( s_{\text{v}} \), whether \( ML_{NC} \) is equal to PCRs and the service life of certificate. If pass the verification, NC sends the order to CLC, that run the service on the NC.

### 5. The Whole Process of CORA Model

From NC request certificates to client submit service request and running service is shown in Figure 2.
The specific process is as follows:

i) NC sends \( \{U, N_i, n, \{\{c_{i}, sm_{i}, ML_{NC}\}_{SK_{NC}}, H(PK_{NC})\}_{PK_{CLC}}\} \) to CI and requests SLA certificate and DAA certificate.

ii) CI executes Issuing DAA Certificate and SLA certificate protocol in A. NC executes SLA Signing protocol in 4.2.

iii) Execute registration protocol in 4.4.

iv) Execute request and verification protocol for Client in 4.5.

6. Performance and Security Analysis

6.1 Performance Analysis

In this paper, as our model is based on DAA and DPTA, so there is no bottleneck problem of third party. However, this will increase the complexity of the protocol, and therefore the request and issuing process of the certificate can be selected at a lower user visit (e.g. morning) time to reduce the impact on the user experience.

During the verification process, the client can dynamically verifies the node’s trusted status based on needs, this may affect the operation of the node. But nodes only need to execute extend and load PCR command and respond in millisecond level, client will not notice. The clients need to compare the real-time PCRs and \( ML_{NC} \) in the SLA certificate and verify the service life of certificate. Since only execute binary compare operating, so the execution time is negligible.

6.2 Security Analysis

When issuing certificates, CI verifies \( ML_{NC} \) and the formula \( \zeta_i^{\gamma} \mod \Gamma \), to ensure that the node is in trusted state. Because of the certificate is no longer a one-time certificate, by adding the service life of certificate to specify the period of the certificate. So the node update certificate regularly to further ensure its safety. The correctness and unforgeability of the protocol have been proved in the literature [3] [4].

In the signing process, after the node receives the SLA certificates, node configuration is hidden by using the Pedersen commitment and zero-knowledge proof. Client can’t know the specific configuration of the node, thus the node configuration concealed. Moreover, as this process uses attestation identity key AIK, so it meet un-connectivity, users can not connect to unreliable attestation process, to ensure the integrity of the attestation.

In the client and CLC verification process, they can dynamically verify the node trusted state, by compare the real-time PCRs and \( ML_{NC} \) in the SLA certificate. If PCRs is equal to \( ML_{NC} \), then the node is trusted. If not, just indicate the node may have been attacked; client can stop using this node to run the service.

7. Conclusion

In cloud environment based on Direct Anonymous Attestation (DAA) and Dynamic Property Trusted Attestation (DPTA), we propose a clients oriented remote attestation (CORA) model, enables clients to select the node in corresponding security level according to their needs and verify the node’s trusted state dynamically. And on the basis of DPTA adds the service life of certificate parameters. Attribute service life to SLA certificates. By regularly updating the certificate can enhance security of attestation. Because clients know the security level and trusted state of the node, so it enhances the clients’ trust to the cloud service provider trust and confidence to use the cloud services.
Acknowledgements

This work was supported by National Science and Technology Support Program (2013BAK07B04).

References


Authors

Liu Zhenpeng, he is the director of Hebei University computer network information center. He has research interest in cloud computing and information security.

Wang Xu, he is the postgraduate student of Hebei University. He has research interest in cloud computing and information security.