Trusted virtual machine monitor-based group signature architecture

D. Zou¹  H. Jin¹  J.H. Park²  H.-C. Chao³,*  Y. Li¹

¹Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, People’s Republic of China
²Department of Computer Science and Engineering, Kyungnam University, Kyungnam, Korea
³Department of Electronic Engineering and Institute of Computer Science & Information Engineering, National Ilan University, I-Lan, Taiwan
*Department of Electrical Engineering, National Dong Hwa University, Hualien, Taiwan
E-mail: hcc@niu.edu.tw

Abstract: Group communication is an important technique for many network computing applications. In group communication, a member in a group sends a message to others normally by multicast. Group signature guarantees the integrity of the exchanged data and provides source authentication. In a virtual machine (VMs) based computing system, a virtual machine monitor (VMM) allows applications to run in different VMs strongly isolated from each other. A trusted VMM (TVMM) based platform can provide stronger security protection for group signature systems than traditional computing platforms can. The authors first introduce a TVMM-based group signature architecture and a TVMM security protection mechanism for group signature components. Then, the authors propose a group signature scheme using the function of message checking based on the discrete logarithm problem. Finally, the authors prove the correctness of the group signature scheme and analyse its security in virtual computing environments.

1 Introduction

Group communication is a highly efficient message-exchanging mechanism for multiple participants involved in a task, and group security aims to protect the exchanged message in group communication systems. Since in traditional computing environments, an operating system (OS) directly runs on the hardware, it is difficult for the OS to protect itself from being attacked. Applications running on an untrusted operating environment are vulnerable to attacks. A suspicious application can easily attack other applications because of the lack of an effective isolation mechanism. Consequently, when group communication applications are implemented on top of an untrusted operating platform, sensitive information will be easily disclosed by the attackers even if group management itself can provide strong security.

In a virtual machine (VM) based computing system, multiple applications run in different VMs with their hardware protection domains strongly isolated from each other. Furthermore, migration can be easily implemented based on the virtual machine monitor (VMM) architecture. Normally, the VMM platforms, such as VMW and XEN, mainly provide VM management and physical resource allocation for each VM, and are simpler and stronger than traditional OSs. With its underlying trusted computing hardware, the trusted computers can provide security support for applications. The VMM architecture enables trusted computers to provide more flexible, isolated environments for each application than traditional architectures can do.

The group signature technique guarantees the integrity of exchanged data and provides source authentication. There are many group signature schemes, such as the group signature scheme based on a discrete logarithm, the threshold group signature scheme, the group blind signature scheme and the forward-secure group signature scheme. Although these schemes can meet the security requirements of some applications in some aspects, they fail to consider how to
use a trusted platform to protect the signature components from being attacked. For such applications, with trusted members and platform, using a single signer is more efficient and secure than using more than one signers. In this paper, we propose a group signature scheme with the function of message checking based on the discrete logarithm problem (DLP), and individual group members can sign a message on behalf of the group and only the specified receiver can recover, verify and check the message. Moreover, in case of disputes, the group controller can reveal the identity of the signer, and existing group signature schemes lack such a checking function. Additionally, in our trusted VMM (TVMM) based group signature architecture, the trusted platform can protect the group controller, group members and its sensitive information, such as a private key, residing on it.

The rest of this paper is organised as follows. We discuss related work in Section 2. In Section 3, we introduce a group signature architecture and a group signature scheme with the function of message checking in the trusted virtual computing environment. In Section 4, we prove the correctness and security of our scheme, and analyse its security in TVMM-based architecture. Finally, the conclusions and future work are presented in section 5.

2 Related Work

In the 1970s, VM was a software replica of an underlying real machine, and multiple isolated VMs could operate on the same host machine concurrently [1, 2]. Over the past few years, with the advent of (multicore) technologies, the VM has regained a great deal of attention. In fact, a VM environment is created by a VMM, referred to as ‘operating system for operating systems’ [3]. The monitor creates one or more VMs on top of a single real machine. Each VM provides facilities for an application or a ‘guest system’ regarded as an execution in a normal hardware environment.

There are two different methods to build a VM system. One is to implement the VMM between the hardware and the guest systems; examples include as a Xen [4] and VMware ESX Server [5]. The other is to implement the VMM as a normal process on top of a real OS, as adopted by VMware Workstation [6] and User-Mode Linux [5]. Some security research was conducted based on the VMM architecture. For example, Dunlap et al. proposed to use VMs to enhance system security. In [7], Revirt, an intermediate layer between the monitor and the host system, captures data sent to the host system through the VM’s syslog process (the standard UNIX logging daemon). In the case of the virtual system being compromised, the invader may manipulate the log messages and impair their reliability. In [8], a virtual machine introspection intrusion detection system is described for searching intrusion evidences. In the system, the intrusion detection system executes in a privileged VM and scans data extracted from other VMs. The secure hypervisor (sHype) project [9] aims to support controlled sharing of resources among VMs on a platform, such as memory, CPU cycles and network bandwidth. In [10], a simple yet effective usage control model UCONKI with unique properties of decision continuity and attribute mutability is proposed for OS kernel integrity protection. Furthermore, to enforce UCONKI security policies, a VMM-based architecture is isolated and protected from other untrusted processes inside a VM. In [11], the risk flow policy describes the authorised risks because of covert flows. In this paper, we examine the ability of four policy models to express risk flow policies. The above-mentioned projects have ignored the security of the VMM itself.

The trusted computing group (TCG) defines a set of specifications to provide hardware-based root of trust and a set of primitive functions to propagate trust to application software as well as across platforms [12, 13]. The root of trust in the TCG is a hardware component on the motherboard of a platform called the trusted platform module (TPM). The TPM provides protected data (cryptographic secrets and arbitrary data) by never releasing a root key outside the TPM. In addition, the TPM presents some primitive cryptographic functions, such as random number generation, RSA key generation and RSA asymmetric key algorithms. Most importantly, a TPM provides a mechanism of integrity measurement, storage and reporting of a platform achieving strong protection capabilities and attestations. To utilise the functions provided by TPM, TCG defines TCG Software Stack (TSS) specification [14]. As an integral part of each platform, the TSS in this specification provides functions that can be used by enhanced OSs and applications, and supplies one entry point for applications to the TPM functionality.

In [15], the design and implementation of a TPM facility is presented. In this module, the TPM is virtualised and can support higher level services. Moreover, it can also support suspend and resume operations, as well as migration of a virtual TPM instance with its respective VM across platforms. In [16], a flexible architecture for trusted computing is presented referred to as Terra. On Terra, applications with a wide range of security requirements run simultaneously on the current OS over commodity hardware implemented by a TVMM. As a high-assurance VM, the TVMM partitions a single tamper-resistant, general purpose platform into multiple isolated VMs.

A group signature is first introduced by Chaum and van Heyst in [17], allowing each group member to sign messages on behalf of a group anonymously and unlinkably. However, in case of later disputes, a designated group manager can open a group signature and then identify the true signer.

In 1998, Lee and Chang [18] presented an efficient group signature scheme based on the DLP. Since two same pieces of information are included in all group signatures generated by the same group member, their scheme is obviously linkable. Therefore it needs to be further improved. To provide unlinkability, an improved group signature scheme is proposed in [19]. Unfortunately, the improved scheme is still...
shown to be linkable [20]. Therefore based on Shamir's idea of identity (ID) based cryptosystems [21], Tseng and Jan [22] proposed an ID-based group signature scheme. In this ID-based group signature scheme, anyone (not necessarily a group member) is able to generate a valid group signature on any message, which cannot be opened by the group manager. Therefore this scheme is forgeable. To solve the problem, in [23, 24], Tseng and Jan revised their schemes, and Popescu presented a modification to the Tseng-Jan ID-based scheme [25]. After that, Xian and You [26] proposed a new group signature scheme with strong separability such that the group manager can be split into a membership manager and a revocation manager. In addition, based on the above schemes, Wang [27] presented a security analysis for these group signature schemes. In this paper, we design a TVMM-based group signature scheme, and propose a group signature scheme with an additional function of message checking.

3 TVMM based group signature architecture

In this section, we describe TVMM-based group signature architecture, as depicted in Fig. 1, and introduce a group signature scheme. At the heart of the system architecture, the TVMM can virtualise machine resources by enabling VMs to run independently and concurrently.

3.1 TVMM-based group signature architecture

We first introduce the main components of TVMM-based group signature architecture and secure migration for the group controller, and then propose a group signature scheme.

3.1.1 Components of TVMM-based group signature architecture: There are three levels in the architecture, including hardware platform, TVMM and secured VM for group signature components.

Hardware platform: In the process of attestation, hardware embedded with cryptographic keys is the trust base of the attestation chain. The hardware supports virtualisation, secure I/O and device isolation. The hardware vendor signs their products to prevent leaking of privacy through the hardware private key in the process of attestation. The security chipsets have a set of rich cryptographic operations defined by the TCG and store small amounts of information such as cryptographic keys. Embedded in hardware with a carefully designed interface, the TPM is resistant to software attacks.

**Trusted virtual machine monitor:** The VMM provides VM management with interfaces to create and manage VMs. Additionally, the TVMM provides security-enhanced functions such as interposition, I/O sealing, isolation and attestation. We introduce these functions as follows:

i. **Interposition mechanism.** This mechanism takes charge of kernel-user switches, saves/restores the CPU context owned by a trusted process and hides some general purpose registers to the OS kernel to avoid a replay attack.

ii. **I/O sealing mechanism.** This mechanism transparently encrypts and decrypts sensitive I/O data to prevent the OS kernel from observing the data. During being transferred to physical storage, sensitive data of a trusted process is transparently encrypted. The TVMM encrypts each of the I/O system call parameters before passing them to the OS kernel, and intercepts the memory-mapped I/O page table updating requests and decrypts the data on the first page fault.

iii. **Isolation mechanism.** The hardware protection domain can isolate its corresponding VM from other VMs, and a secure isolation is important for the confidentiality and integrity of the VM.

iv. **Attestation mechanism.** This mechanism can convince remote parties that the VM or an application is not tampered. Specifically, an application running in a VM can authenticate itself to a remote party, and then the remote party puts trust in the application and has faith that the application will behave as desired. By this mechanism, our proposed group signature architecture can provide application-dependent attestation between the group controller and the group members.

**Secured VM for group signature components:** The open VM can provide the semantics of today’s open platforms running OSs. The TVMM offers strong security functions to all VMs. OSs running in VMs may be either as simple as a boot loader plus application code or as complex as a commodity OS with only one application running. On the other hand, applications can completely tailor the OS to their security needs.

There are two kinds of VMs, which are the privileged and normal VMs. For example, Dom 0 is a privileged VM and Dom U is a normal VM in the VM platform, XEN. The privileged VM can manage other normal VMs. For instance, a privileged VM can create, pause, resume and destroy a normal VM. Normally, the privileged VM utilises a tailored
OS, and an application with high-security requirements runs in a normal trusted VM with an embedded, tailored OS.

The group signature provides message integrity and source authentication. The normal trusted VM provides a secure environment for group signature components, including the group signature controller and group signature management. Normally, the group signature controller is taken as the server side of a group communication application, and group signature management is taken as its client side. A group member in the client side can join a group managed by the group controller in the server side. Group signature management is used by group members to sign or verify the exchanged message, and the group signature controller is used by the group controller to manage the message exchanged among group members.

3.1.2 Secure migration for the group signature controller: To guarantee the availability of the group signature controller, the VM where the group signature controller is located should have a secure migration from one trusted platform, named source platform A, to another, named destination platform B.

There are four stages during the whole secure migration, including attestation between the two platforms, the secure transfer of the VM image, VM image verification and restoration of the VM image. As depicted in Fig. 2, two processes on both sides, including the migration controller on source platform A and that on destination platform B, are responsible for secure migration. We explain how it works as follows.

In the first phase, with trusted computing technology, platforms can authenticate each other with attestation identity key (AIK) credentials. AIK credentials are issued by a privacy CA, which plays the role of a trusted third party. Two arbitrary numbers, Nonce 1 and Nonce 2, are generated and used to enhance the communication security.

In the second phase, a VM image with the group signature controller serialised and a symmetric key created, is encrypted with the symmetric key, and a digest for the image is generated. To keep the confidentiality of the symmetric key, it is encrypted with the public key of platform B.

In the third phase, after the migration controller on platform B receives the data, the symmetric key is decrypted with the private key of platform B, the image is decrypted with the symmetric key, and the digest is re-calculated to verify whether the VM image was tampered or not.

In the last phase, after the VM image is verified, the VM image is securely restored. The group controller continues working and provides group control service for the group members again. This is an important procedure to guarantee the availability of the group controller.

XEN supports image migration, and we extend it to support the secure migration protocol shown in Fig. 2.

3.2 Group signature scheme: Based on the group signature architecture in Section 3, we propose a group signature scheme with three phases, including the parameter generating phase, the signature generating phase
and the signature verification phase. The group signature scheme is described as follows:

**Parameter generating phase:** Assume \( p \) and \( q \) are two large prime numbers such that \( q | p - 1 \), and \( g \) is a generator of order \( q \) in \( \mathbb{GF}(p) \).

**Step 1:** Each group member \( GM_i \) randomly chooses an integer \( x_i \) and computes the public keys \( y_i = g^{x_i} \mod p \) (\( i = 1, 2, 3, \ldots, n \)).

**Step 2:** The group controller randomly chooses an integer \( x_c \) and computes the public key \( y_c = g^{x_c} \mod p \).

**Step 3:** For each group member \( GM_i \), the group controller randomly chooses an integer \( a_i \) in \( \mathbb{Z}_q^* \), and computes \( r_i = a_i \cdot ID_i - x_c \mod q \) and \( s_i = y_c^{a_i} \mod p \) (\( i = 1, 2, 3, \ldots, n \)), where \( ID_i \) is the identity of group member \( GM_i \).

**Step 4:** The group controller sends \((r_i, s_i)\) to the group member \( GM_i \) secretly.

**Step 5:** After receiving \((r_i, s_i)\), \( GM_i \) verifies the validity by checking the following equations:

\[
s_i = (g^{r_i} \cdot y_i)^{s_i/ID_i} \mod p \quad (1)
\]

**Signature generating phase:**

**Step 1:** The \( GM_i \) computes \( M = M_{\text{check}} || M_{\text{original}} \), where \( M_{\text{check}} \) is a short checking message, \( M_{\text{original}} \) is the message that the group member \( GM_i \) wants to sign and \( || \) denotes the concatenation.

**Step 2:** The \( GM_i \) randomly chooses three integers \( b_1, b_2 \) and \( b_3 \) in \( \mathbb{Z}_q^* \).

**Step 3:** The \( GM_i \) computes

\[
\begin{align*}
\beta &= x_i \cdot b_1 \mod q \quad (2) \\
\delta &= s_i \cdot b_3 \mod p \quad (3) \\
\xi &= g^{\beta \cdot \delta} \mod p \quad (4) \\
\psi &= M \cdot y_i^b \cdot h(\beta(\xi(\delta))^{b_1}) \mod p \quad (5) \\
\rho &= b_1 - r_i \cdot b_3 \mod p 
\end{align*}
\]

where \( h() \) is a publicly known hash function.

**Step 4:** The group signature for message \( M \) is \( \eta \), where \( \eta = (\beta, \delta, \xi, \psi, \rho, M_{\text{check}}) \).

**The signature verification phase:**

The group member \( GM_j \) verifies the group signature with the following two steps.

**Step 1:** Recovers the message according to the following equation

\[
M = \psi \cdot [(g^\beta \cdot y_i^{\delta(\xi(\delta))})^{\xi(\beta(\xi(\delta))^{b_1})} \mod p] \quad (7)
\]

**Step 2:** Checks the following congruence relation

\[
M_{\text{check}} = \text{head}(M, L) \quad (8)
\]

where \( L \) is the bit number of the checking message \( M_{\text{check}} \) and \( \text{head}(M, L) \) is a function to retrieve the first \( L \) bits of \( M \). By verifying the message integrity with \( M_{\text{checks}} \), \( GM_j \) can decide to accept/reject the group signature \( \eta \).

In the case of a dispute, the signature must be opened to reveal the identity of the signer. Because the group controller has the privilege to access \((ID_i, y_i, a_i)\) of each group member \( GM_i \), the group controller can reveal the identity of the signer via the Equation \( \delta = g^{\beta \cdot a_i ID_i} \mod p \).

### 4 Analysis of TVMM-based group signature

In this section, we will analyse the correctness and security of our proposed group signature scheme first and analyse the security of the group signature architecture based on TVMM.

#### 4.1 Correctness analysis of the signature scheme

Analysis of the correctness of our presented scheme is presented as follows.

**Theorem 1:** The validity of \((r_i, s_i)\) can be verified by the group member \( GM_i \) via the equation \( s_i = (g^{r_i} \cdot y_i)^{s_i/ID_i} \mod p \).

**Proof:** According to the Equation \( y_i = g^{x_i} \mod p \) and \( r_i = a_i \cdot ID_i - x_c \mod q \), we can rewrite the right part of (1) as follows

\[
(g^{r_i} \cdot y_i)^{s_i/ID_i} = (g^{a_i ID_i - x_c} \cdot g^{x_i})^{s_i/ID_i} \mod p = g^{s_i \cdot x_i} \mod p = y_i^{s_i} \mod p = \text{simod} p
\]

Therefore the validity of \((r_i, s_i)\) can be verified by the group member \( GM_i \) via the above inference. \( \Box \)

**Theorem 2:** The group signature can be verified by the message receiver via (7) and (8).

**Proof:** According to (2)–(6), we can rewrite the right part of (7) as follows

\[
\begin{align*}
M &= \psi \cdot [(g^\beta \cdot y_i^{\delta(\xi(\delta))})^{\xi(\beta(\xi(\delta))^{b_1})} \mod p] \\
&= M \cdot [g^{b_1} \cdot (g^{\beta} \cdot y_i^{\delta(\xi(\delta))})^{\xi(\beta(\xi(\delta))^{b_1})} \mod p] \\
&= M \cdot [(g^{b_1} \cdot g^{\beta}) \cdot y_i^{\delta(\xi(\delta))} \cdot \xi(\beta(\xi(\delta))^{b_1}) \mod p]
\end{align*}
\]
compute (7), and obtains the equation
\[ \gamma \equiv s_i \cdot \beta \mod p \]
\[ M \equiv M \mod p \]
Consequently, the correctness of (7) is verified via the above inference. \hfill \Box

Theorem 3: The validity of the identity of the group member GM can be verified by the group controller via the equation \( \delta = g^{a_i \cdot 1D} \mod p \).

Proof:
\[ \begin{align*}
  g^{a_i \cdot 1D} \mod p &= g^{x_i \cdot a_i \cdot b_i \cdot 1D} \mod p = y_i \cdot b_i \cdot 1D, \\
  \mod p &= y_i \cdot 1D, \\
  \mod p &= \delta
\end{align*} \]
Hence, the validity of the identity of the group member GM is verified via the above inference. \hfill \Box

4.2 Security analysis of signature scheme

The security of our proposed scheme is based on the irreversibility of the DLP. In this section, we analyse the security of our proposed scheme. Before analysing, we list some possible attacks against the proposed scheme as follows.

Attack 1: An adversary intercepts a valid membership \((r_i, s_i)\), and then tries to forge a group signature.

Analysis of attack 1: If an adversary intercepts a valid membership \((r_i, s_i)\), he/she can compute \(\delta\) and \(\xi\) by (3) and (4), respectively. Although he/she can forge an integer \(x_i^*\) to compute \(\beta, \psi\) and \(\rho\) by (2), (5) and (6), he/she cannot forge a group signature by (7) without the secret key \(x_i^*\). Hence, Equation (8) holds. As a result, even if an adversary intercepts a valid membership \((r_i, s_i)\), he/she will not be able to forge a group signature.

Attack 2: An adversary tries to forge a group signature without intercepting any information.

Analysis of attack 2: Without intercepting any information, an adversary intending to forge a group signature will face the DLP. There are five situations for this.

Situation 1: An adversary chooses a message \(M = M_{\text{check}}||M_{\text{original}}\), randomly selects \(\beta, \delta, \xi, \psi\) and \(\rho\) to compute (7), and obtains the equation \(\psi \cdot g^{\psi \cdot 1D} \cdot B^{\xi \cdot 1D} \cdot \psi \cdot 1D = \theta \mod \rho\), where \(A, B, D\) and \(\theta\) are integers; \(x_i\) is a secret key of the verifier GM, and \(A = h(\beta \cdot 1D) \cdot b_i, B = g^{\beta \cdot x_i \cdot 1D} \cdot \delta, D = g^\psi \cdot \xi\). The adversary must solve the congruence relation \(\psi \cdot A^{\psi \cdot B \cdot 1D} \cdot \xi \cdot g^{\beta \cdot h(\delta \cdot 1D)} \mod \rho = \theta \mod \rho\) for \(\psi, B, \xi, g\). Because of the DLP, the adversary cannot forge the group signature and, hence, the checking Equation holds.

Situation 2: An adversary chooses a message \(M = M_{\text{check}}||M_{\text{original}}\), randomly selects \(\beta, \delta, \xi, \psi\) and \(\rho\) to compute (7) and obtains the equation \(\psi \cdot B^{\psi \cdot 1D} \cdot \psi \cdot 1D = \theta \mod \rho\), where \(A, B, D\) and \(\theta\) are integers; \(x_i\) is a secret key of the verifier GM, and \(A = h(\beta \cdot 1D) \cdot b_i, B = g^{\beta \cdot x_i \cdot 1D} \cdot \delta, D = g^\psi \cdot \xi\). The adversary must solve the congruence relation \(\psi \cdot A^{\psi \cdot B \cdot 1D} \cdot \xi \cdot g^{\beta \cdot h(\delta \cdot 1D)} \mod \rho = \theta \mod \rho\) for \(\psi, B, \xi, g\). Because of the DLP, the adversary cannot forge the group signature and, hence, the checking Equation holds.

Situation 3: An adversary chooses a message \(M = M_{\text{check}}||M_{\text{original}}\), randomly selects \(\beta, \delta, \xi, \psi\) and \(\rho\) to compute (7) and obtains the equation \(\psi \cdot A^{\psi \cdot B \cdot 1D} \cdot \xi \cdot g^{\beta \cdot h(\delta \cdot 1D)} \mod \rho = \theta \mod \rho\), where \(A, B, D\) and \(\theta\) are integers; \(x_i\) is a secret key of the verifier GM, and \(A = h(\beta \cdot 1D) \cdot b_i, B = g^{\beta \cdot x_i \cdot 1D} \cdot \delta, D = g^\psi \cdot \xi\). The adversary must solve the congruence relation \(\psi \cdot A^{\psi \cdot B \cdot 1D} \cdot \xi \cdot g^{\beta \cdot h(\delta \cdot 1D)} \mod \rho = \theta \mod \rho\) for \(\psi, B, \xi, g\). Because of the DLP, the adversary cannot forge the group signature and, hence, the checking Equation holds.

Situation 4: An adversary chooses a message \(M = M_{\text{check}}||M_{\text{original}}\), randomly selects \(\beta, \delta, \xi, \psi\) and \(\rho\) to compute (7) and obtains the equation \(\psi \cdot A^{\psi \cdot B \cdot 1D} \cdot \xi \cdot g^{\beta \cdot h(\delta \cdot 1D)} \mod \rho = \theta \mod \rho\), where \(A, B, D\) and \(\theta\) are integers; \(x_i\) is a secret key of the verifier GM, and \(A = h(\beta \cdot 1D) \cdot b_i, B = g^{\beta \cdot x_i \cdot 1D} \cdot \delta, D = g^\psi \cdot \xi\). The adversary must solve the congruence relation \(\psi \cdot A^{\psi \cdot B \cdot 1D} \cdot \xi \cdot g^{\beta \cdot h(\delta \cdot 1D)} \mod \rho = \theta \mod \rho\) for \(\psi, B, \xi, g\). Because of the DLP, the adversary cannot forge the group signature and, hence, the checking Equation holds.

Situation 5: An adversary chooses a message \(M = M_{\text{check}}||M_{\text{original}}\), randomly selects \(\beta, \delta, \xi, \psi\) and \(\rho\) to compute (7) and obtains the Equation \(\psi \cdot A^{\psi \cdot B \cdot 1D} \cdot \xi \cdot g^{\beta \cdot h(\delta \cdot 1D)} \mod \rho = \theta \mod \rho\), where \(A, B, D\) and \(\theta\) are integers; \(x_i\) is a secret key of the verifier GM, and \(A = h(\beta \cdot 1D) \cdot b_i, B = g^{\beta \cdot x_i \cdot 1D} \cdot \delta, D = g^\psi \cdot \xi\). The adversary must solve the congruence relation \(\psi \cdot A^{\psi \cdot B \cdot 1D} \cdot \xi \cdot g^{\beta \cdot h(\delta \cdot 1D)} \mod \rho = \theta \mod \rho\) for \(\psi, B, \xi, g\). Because of the DLP, the adversary cannot forge the group signature and, hence, the checking Equation holds.

In conclusion, if an adversary does not intercept any information, it is very difficult for the adversary to forge a group signature.

Attack 3: Without any information except his/her own secret key \(x_j\), the verifier GM tries to forge a group signature.
Analysis of attack 3: Under attack 3, there are five possible situations for the verifier GMj to forge a group signature.

Situation 1: The verifier GMj chooses a message $M = M_{\text{check}} \| M_{\text{original}}$, randomly selects $\beta$, $\delta$, $\xi$, and $\psi$ to compute (7) and obtains the equation $\psi \cdot g^{\delta \xi} \cdot B^\xi \mod p = \theta \mod p$ (where $A$, $B$, and $\theta$ are integers, and $A = x_j \cdot b(\beta) \cdot \xi \cdot b_j$, $B = y_j \cdot g^{\delta \xi} \cdot \xi \cdot b(\delta \theta)$. The verifier GMj must solve the congruence relation $\psi \cdot g^{\delta \xi} \cdot B^\xi \mod p = \theta \mod p$ for $p$. However, it is very difficult to calculate the parameter $p$ when we know $\psi$, $A$, $B$, $\theta$ and $g$. Because of the DLP, the verifier cannot forge the group signature and, hence, the checking equation holds.

Situation 2: The verifier GMj chooses a message $M = M_{\text{check}} \| M_{\text{original}}$, randomly selects $\beta$, $\delta$, $\xi$, and $\psi$ and $p \mod \phi(p)$ to compute (7) and obtains the equation $\psi \cdot B^{\delta \xi} \cdot A \cdot D^{\delta \xi \mod p = \theta \mod p}$ (where $A$, $B$, $D$ and $\theta$ are integers, and $A = x_j \cdot b(\beta) \cdot \xi \cdot b_j$, $B = g^{\delta \xi} \cdot \xi \cdot b(\delta \theta)$. The verifier GMj must solve the congruence relation $\psi \cdot B^{\delta \xi} \cdot A \cdot D^{\delta \xi} \mod p = \theta \mod p$ for $p$. However, it is very difficult to calculate the parameter $p$ when we know $A$, $B$, $D$, $\beta$, $\theta$ and $g$. Because of the DLP, the verifier GMj cannot forge the group signature and, hence, the checking equation holds.

Situation 3: The verifier GMj chooses a message $M = M_{\text{check}} \| M_{\text{original}}$, randomly selects $\beta$, $\delta$, $\xi$, $\psi$ and $p$ to compute (7) and obtains the equation $\psi \cdot B^{\delta \xi} \cdot A \cdot D^{\delta \xi \mod p = \theta \mod p}$ (where $A$, $B$, and $\theta$ are integers, and $A = (g^{\delta \xi} \cdot \xi \cdot b(\delta \theta) \cdot \xi \cdot b_j = \beta \mod (\delta \theta))$. The verifier GMj must solve the congruence relation $\psi \cdot B^{\delta \xi} \cdot A \cdot D^{\delta \xi} \mod p = \theta \mod p$ for $p$. However, it is very difficult to calculate the parameter $p$ when we know $A$, $B$, $\psi$, $\beta$, $\theta$. Because of the DLP, the verifier cannot forge the group signature and, hence, the checking equation holds.

Situation 4: The verifier GMj chooses a message $M = M_{\text{check}} \| M_{\text{original}}$, randomly selects $\delta$, $\xi$, $\psi$ and $p$ to compute (7) and obtains the equation $\psi \cdot A^{\beta \xi} \cdot B^{\delta \xi} \cdot D^{\delta \xi \mod p = \theta \mod p}$ (where $A$, $B$, and $\theta$ are integers, and $A = (g^{\delta \xi} \cdot \xi \cdot b(\delta \theta) \cdot \xi \cdot b_j = \beta \mod (\delta \theta))$. The verifier must solve the congruence relation $\psi \cdot A^{\beta \xi} \cdot B^{\delta \xi} \cdot D^{\delta \xi} \mod p = \theta \mod p$ for $p$. However, it is very difficult to calculate the parameter $p$ when we know $A$, $\delta$, $\xi$, $\psi$ and $\theta$. Because of the DLP, the verifier cannot forge the group signature and, hence, the checking equation holds.

Situation 5: The verifier GMj chooses a message $M = M_{\text{check}} \| M_{\text{original}}$, randomly selects $\beta$, $\xi$, $\psi$ and $p$ to compute (7) and obtains the equation $\psi \cdot A^{\beta \xi} \cdot B^{\delta \xi} \cdot D^{\delta \xi \mod p = \theta \mod p}$ (where $A$ and $\theta$ are integers, and $A = (g^{\delta \xi} \cdot \xi \cdot b(\delta \theta) \cdot \xi \cdot b_j = \beta \mod (\delta \theta))$. The verifier must solve the congruence relation $\psi \cdot A^{\beta \xi} \cdot B^{\delta \xi} \cdot D^{\delta \xi} \mod p = \theta \mod p$ for $p$. However, it is very difficult to calculate parameter $\delta$ when we know $A$, $\beta$, $\xi$, $\psi$ and $\theta$. Because of the DLP, the verifier cannot forge the group signature and, hence, the checking equation holds.

Therefore without any information except his/her own secret key $x_j$, it is very difficult for the verifier GMj to forge a group signature.

Attack 4: An adversary tries to identify the actual signer by intercepting a valid group signature $\eta = \{\beta, \delta, \xi, \psi, \rho, M_{\text{check}}\}$.

Analysis of Attack 4: With the intercepted valid group signature $\eta = \{\beta, \delta, \xi, \psi, \rho, M_{\text{check}}\}$, an adversary computes $y_j$ by the equation $y_j = g^{\delta \xi} \mod p$ and $s_j = y_j^{\theta} \mod p$ in order to obtain the identity of the actual signer. Since $b_1$ is an unknown number, the adversary fails to obtain the signer. Without the information of $a_i$, $b_1$ and IDi, based on the equation $s_j = y_j^{\theta} \mod p$ and $\beta = x_j \cdot b(\beta) \cdot \xi \cdot b_j$ it is very difficult to calculate the parameter IDi. Therefore the adversary fails to obtain the identity of the actual signer.

Attack 5: With the interception of two valid group signatures, an adversary tries to identify whether the two group signatures were generated by the same signature or not.

Analysis of Attack 5: It is assumed that an adversary intercepts two valid group signatures $\eta = \{\beta, \delta, \xi, \psi, \rho, M_{\text{check}}\}$ and $\eta' = \{\beta', \delta', \xi, \psi', \rho', M_{\text{check}}\}$; he/she can compute $g^{\beta \xi} = g^{\beta' \xi'} \mod p$ and $\beta / \delta = (x_j \cdot b(\beta) \cdot \xi \cdot b_j = \beta \mod (\delta \theta))$. The adversary can obtain $\beta / \delta = (x_j \cdot b(\beta) \cdot \xi \cdot b_j = \beta \mod (\delta \theta))$ and check whether the equation holds or not. If the equation holds, the two valid group signatures $\eta$ and $\eta'$ were generated by the same signer. Since both the integer $a_i$ and IDi are unknown, the adversary cannot identify whether the two group signatures were generated by the same signer or not.

From the above analysis, we can conclude that our scheme satisfies the requirement of unforgeability, anonymity, unlinkability and exculpability [12]. In addition, the group controller can acquire the (IDi, $y_j$, $a_i$) of GMj, because of having an access to (IDi, $y_j$, $a_i$) of each member GMi. Hence, the group controller can determine the signer. Therefore our proposed scheme also satisfies the requirement of traceability [12].

4.3 Security analysis of group signature architecture

When requested for, the sensitive information of a group member/controller is encrypted. For group communication applications, the TVMM intercepts related system calls and encrypts the data before passing it to the OS kernel, and then decrypts the data fetched from the OS kernel before passing them to the user space. Moreover, in order to prevent information leakage, a key provided by the TVMM platform
encrypts the related program code and data when the group communication applications are loaded, and decrypts the code and data when the process is created. In addition, the architecture can transparently encrypt sensitive information above the TVMM layer by using the proposed group signature mechanism. With the above two mechanisms, the sensitive information (during) in this process is not leaked. Consequently, it is very secure for sensitive information of a group member and the group controller.

In this TVMM-based architecture, an attestation mechanism enables group communication applications to authenticate themselves to remote parties. The group controller can confirm the correctness and security of each group member based on the trusted platform. After receiving the attestation message, the group controller can trust the group members in the remote the TVMM platform. A trust chain is first built for the attestation on TVMM starting up and begins with the hardware with a private key permanently embedded in a tamper-resistant chip and with the vendor's signature. The tamper-resistant hardware certifies the system firmware (e.g. BIOS). The firmware certifies the system boot loader, which certifies the TVMM, and the TVMM certifies the VMs that it loads. In the following step, a group certificate is built for group attestation. The group controller use the group private key to sign a message and each group member can use the group public key to verify the correctness and security of the message. By using the group certificate in a certificate chain, the group controller can determine whether a group member is correct and secure.

In the TVMM-based group signature architecture, the group controller first chooses a public key for the applications and sends the message \((r_i, s_i)\) to the group member \(GM_i\) secretly. The message \((r_i, s_i)\) and the corresponding VM image data are encrypted by the symmetric key, which is created by the source platform. Then, the encrypted message is sent to the destination and only the destination platform can recover it by using the TPM. Thus, it is impossible for any adversary to decrypt the message. As a result, it is secure when the key component migrates from the source platform to the destination platform.

5 Conclusion

In this paper, we propose a TVMM-based group signature architecture and design a novel group signature scheme with message checking, achieving high security of group communication applications. We analyse how the TVMM provides the security guarantee for group signature components. The proposed secure migration supports the application server and the group signature controller. We analyse the correctness and security of our group signature scheme and the security of the group signature architecture.

6 Acknowledgment

This work was supported by National 973 Basic Research Program under Grant No. 2007CB310900 and National Science Foundation of China under Grant No. 60503040.

7 References


