Hindawi
Scientific Programming
Volume 2021, Article ID 5542852, 8 pages
https://doi.org/10.1155/2021/5542852

Research Article

TAME^C: Trusted Augmented Mobile Execution on Cloud

Syed Luqman Shah,[1] **Irshad Ahmed Abbasi**,[2] **Alwalid Bashier Gism Elseed**,[2] **Sikandar Ali**,[3,4] **Zahid Anwar**,[5] **Qasim Rajpoot**,[5] **and Maria Riaz**[5]

[1]*Department of Information Technology, University of Haripur, Haripur, Pakistan*
[2]*Department of Computer Science, Faculty of Science and Arts at Belgarn, University of Bisha, P.O. Box 60, Sabt Al-Alaya, Bisha 61985, Saudi Arabia*
[3]*Department of Computer Science Technology, China University of Petroleum-Beijing, Beijing 102249, China*
[4]*Beijing Key Lab of Petroleum Data Mining, China University of Petroleum-Beijing, Beijing 102249, China*
[5]*School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Islamabad, Pakistan*

Correspondence should be addressed to Irshad Ahmed Abbasi; aabasy@ub.edu.sa and Sikandar Ali; sikandar@cup.edu.cn

Received 21 January 2021; Revised 8 February 2021; Accepted 17 February 2021; Published 8 March 2021

Academic Editor: Shah Nazir

Cloud computing has emerged as an attractive platform for individuals and businesses to augment their basic processing capabilities. Mobile devices with access to Internet are also turning towards clouds for resource-intensive tasks by working out a trade-off between resources required for performing computation on-device against those required for off-loading task to the cloud. However, as with desktop clients, mobile clients face significant concerns related to confidentiality and integrity of data and applications moved to and from the cloud. Cloud-related security solutions proposed for desktop clients could not be readily ported to mobile clients owing to the obvious limitation in their processing capabilities and restrained battery life. We address this problem by proposing architecture for secure exchange and trusted execution between mobile devices and cloud hosts. We establish a symmetric-key-based secure communication channel between mobile and cloud, backed by a trusted coordinator. We also employee a Trusted Platform Module- (TPM-) based attestation of the cloud nodes on which the data and applications of mobile device will be hosted. This gives a comprehensive solution for end-to-end secure and trusted interaction of the mobile device with cloud hosts.

1. Introduction

The enhanced capabilities, improved connectivity, and increasing accessibility of smart phones have triggered a significant growth in the development of mobile applications. Mobile devices are equipped with a wide array of sensors and gadgets which constantly challenge the developers to come up with innovative products to capture the attention of mobile users as well as the market share. Application stores [1,2] are filled with applications that not only appeal to the entertainment-savvy users, but also cater to the information-sensitive domains of finances and health management.

Mobile devices have successfully captured a major chunk of users' attention away from the desktop machines and offer competitive Internet connectivity to add to their appeal. However the catch lies in the constrained processing capabilities and limited battery life of these devices. This is where mobile devices have to look towards their more grounded and plugged computing counterparts for assistance. Since the mobile devices can connect to more powerful machines via GPRS, Wi-Fi, and related networking technologies, it is a natural consequence to try augmenting the processing capabilities of mobile devices by off-loading resource-intensive tasks to resource-rich machines. Cloud computing [3] offers an attractive platform of choice for hosting resource-intensive applications on behalf of mobile devices. Solutions have already been proposed for augmented execution of mobile applications on clouds [4]. Resource-intensive tasks can be off-loaded to the cloud at

various levels of granularity by working out a trade-off between resources required for performing computation on-device against those required for off-loading task to the cloud.

Mobile phone users may utilize the software services hosted on cloud such as Google Apps [5] or even deploy a clone of the mobile phone on the cloud such as Amazon EC2 [6] garnering the benefits of closed-box execution of mobile applications. However, as with other cloud clients, the concerns of privacy and security associated with sending data/applications to the cloud prevail [7]. Even for an average user, the mobile applications and the data on which they operate are of very personal and private nature. A root-of-trust and associated chain-of-trust needs to be established to ensure the integrity and confidentiality of mobile users' data and applications running on the cloud. In case of desktop clients, standardized solutions based on remote hardware-based attestation in the form of Trusted Platform Modules (TPMs) are available [8] and may be extended to set up trusted clouds [9]. However, for mobile devices, these solutions are not applicable due to lack of hardware-based attestation support.

In this paper, we propose a model for securely hosting and executing mobile applications on clouds. We have extended the notion of secure clouds to mobile clients by providing secure communication and execution channels while keeping in view the practical limitations of mobile devices. In Sections 2 and 3, we explain the related work and background on which we build our solution. Section 4 describes our proposed security architecture in detail giving insight to our approach for ensuring secure communication and privacy-protecting data processing. Section 5 gives the security evaluation of the proposed system and summarizes the current progress. Section 6 concludes our work highlighting the future directions.

2. Related Work

The need to augment computing capabilities of a mobile device arises not only to host resource-heavy user applications but also to provide integrity measures to keep the device itself in a secure state. Such measures include running periodic virus scans and performing checks on the integrity of installed mobile applications [10]. The significance of these integrity measures has increased greatly with the widespread use of downloaded mobile applications [11] that may compromise mobile users' data and privacy. Keeping data synchronized between mobile and other devices of a user is another long-standing usage of mobile applications that augments mobile phone capabilities. CloneCloud [12] is a prominent effort in this regard for porting the execution of applications from mobile phones to their "clones" in the clouds, any time during execution. The concept of "elastic execution" proposed by the authors builds on the provision for migrating code running on a host virtual machine (VM) to a target VM at runtime. Since most mobile phones run applications on an underlying application-layer VM [13], this solution has wide-scale applicability. Their solution essentially opens up the Infrastructure as a Service (IaaS) capabilities of the cloud to mobile users. However, it does not focus on the security aspects of hosting data or complete clone of the mobile device on the cloud.

For desktop clients of clouds, a model for trusted clouds has been proposed [9] which ensures confidential execution of guest VMs on clouds. The proposed model extends the notion of remote attestation via TPMs to cloud computing paradigm. A cloud consists of a number of nodes on which the client's VM could be hosted. Every time a VM has to be launched or migrated from one node to the other in the cloud, it is ensured that the node is trustworthy. The focus is on enabling the IaaS providers such as Amazon EC2 to provide closed-box execution environment to their clients. However, the same approach cannot be readily applied to mobile phone clients due to their limited resources and lack of support for hardware-based attestation. In [14], the authors propose an architecture exploiting sealed storage mechanism that attests the cloud node prior to launching customer data and ensures confidentiality of customers' data even if the cloud node becomes compromised. In [15], the authors propose trusted attestation architecture for Infrastructure as a Service. The architecture ensures the attestation of virtual machine, hypervisor, and host operating system using the Trusted Platform Module (TPM) and Virtual Trusted Module (vTPM).

Standards for Mobile Trusted Modules (MTMs) [16] have been proposed for hardware-based attestation of mobile devices, and standardized hardware implementations are expected to be available soon. In [17], the authors have developed an emulator based on a subset of MTM standards for remote attestation of Android platform [13]. The work focuses on establishing trustworthiness of the applications running on Android platform including the Android Dalvik VM. Additionally, the remote attestation process is able to attest the classes loaded by the VM to establish the validity of applications running on top of the virtual machine. It is a significant improvement in the attestation process which is often limited to applications launched directly by the operating system. Their architecture can be readily ported to work with actual hardware MTMs once available. However, the confidentiality and integrity of key-pairs and hashes stored on disk cannot be ascertained. The trustworthiness of a mobile device is not guaranteed any further than the trustworthiness established via maintaining traditional key-stores at the mobile device.

In this paper, we combine the concepts presented in [9, 12] and add in the missing ingredient of trusted execution of mobile applications on clouds to provide a comprehensive end-to-end security architecture for mobile clients. In Section 4, we discuss our approach for addressing the security and trust establishment issues faced by mobile clients when hosting data and applications on cloud. Most of the work carried out in porting mobile applications to clouds focuses on Android-based mobile phones. This is due to the open platform that Android offers as well as the popularity and increasing market share of Android phones [11]. In this context, the focus of our proposed architecture is on securely augmenting capabilities of Android-based mobile devices by employing trusted cloud backend.

3. Background

3.1. Virtualization. Virtualization is a mechanism of dividing hardware or some subset of hardware of a computer system among several virtual machines (VMs), which bear resemblance to the physical system. Virtualization is carried out by a virtual machine monitor (VMM), a layer that divides the hardware. XEN [18] provides a virtualization layer that ensures the separation of VMs and executes instructions on their behalf. At the same time, XEN manages a domain (Dom0), which controls the access of VMs to physical hardware and facilitates communication among VMs hosted on the shared hardware. The virtualized system attempts to provide virtualized network and storage devices, thus providing an environment similar to that of a real system. The frontend drivers communicate with backend drivers via sockets. Normally, backend drivers are provided within Dom0 of XEN, but these drivers can also run within other virtual machines. In order to ensure the security, the applications running in one virtual machine should not be interfered by the applications executing on another VM; it is achieved by isolating the VMs by virtual machine monitor. This separation is maintained through different in-built privileged levels in the CPU. This separation can only be trusted if the whole software stack loaded into the system, right from BIOS to the VMM and over the Dom0, is relied upon. Trusted Computing Base (TCB) for a VM is formed only if the software running on the system is correct and works as expected. This issue has been addressed by measurement techniques provided by Trusted Computing Group (TCG).

3.2. TCG. The specifications published by TCG [19, 20] provide a mechanism to attest the computing platforms based on a Trusted Platform Module (TPM), a root of trust, and a co-processor mounted on the motherboard of the computer. As per latest specifications [21] TPM contains 24 Platform Configuration Registers (PCRs) that get filled with cryptographic hashes of the software stack loaded and running on the system. These PCRs are automatically established when the computer goes through the boot cycle; values in these registers can be extended later on but cannot be replaced with chosen values. The BIOS code forms the Core Root of Trust for Measurement (CRTM), and then every link in the chain is measured by the prior one. CRTM first calculates the BIOS, extends values in relevant register, and then transfer controls to the BIOS. Then, BIOS measures ROM configurations/data and extends values to the TPM registers. Similarly, all the components are measured right from BIOS to the operating system level and measurements are recorded in the PCRs. Even, up and above the OS, the applications can also be measured [22], stored in the kernel held Stored Measurement Log (SML), and then extended in the related register. This extension process ensures that the values in the PCRs cannot be manipulated and reached via any other route. PCR values are signed by Attestation Identity Key (AIK) residing within TPM, which allows the state of computer to be communicated with other parties, who then validates if the values and platform can be trusted. The validity of TPM is verified by these signing key certified by certification authority during remote attestation procedure.

3.3. Trusted Virtualization. A virtualized system enriched with additional components gets complex to measure. The conventional TCG scheme for measurement can be extended [23] to virtualized systems. It allows trust to be communicated over the full software stack of the system, which includes measurement of BIOS, bootloader, VMM, image for Dom0, additional kernel modules, and associated configuration files. The attestation of the system up to the base level can be achieved through this process; however, the end user may also want to attest the guest operating system, which can be achieved by further extending PCR measurements.

The already attested measurement agent in the Dom0 can give directions for the measurement and communication of guest operating systems. Virtual TPM (vTPM) [24] is just an extension of this idea where measurement services along with other TPM functions can be provided by each domain. These vTPMs can be made available from within Dom0 or from within a separate virtual machine linked to the VM.

4. The Architecture

The main objective of this work is to provide a mechanism for secure transfer and trusted execution of mobile users' data and applications on the cloud. In order to guarantee end-to-end security, the proposed system establishes the trustworthiness of the cloud node before giving access to the data via a secure channel. The three main participating entities are (a) the Android-based mobile client, (b) a number of cloud nodes ($N_1, \ldots, N_n$), managed by the Cloud Manager (CM), on which the processing is off-loaded, and (c) a Trusted Coordinator (TC) that acts as the trusted third party and attests the cloud nodes. TC is similar to certificate authority in the TCG adopted Direct Anonymous Attestation protocol [25].

We assume that the cloud nodes, as well as the trusted coordinator, are TPM-enabled platforms and can participate in carrying out remote attestation of the nodes. We also assume that a key-store containing symmetric-keys associated with the mobile client is available at the mobile device. Figure 1 provides an overview of our proposed system architecture. The selection of encryption and attestation technologies is made in keeping with the actual capabilities of each participating entity while ensuring a high level of trust. The Android client can generate and store symmetric keys (AES_{key}) in its key store along with the trusted *key* ($TK_{TC}{}^{public}$) of the TC. The details regarding message exchange between each set of communicating party and the role of different keys are discussed in the following sections.

The configuration files at the mobile client provide basic information related to off-loading data and applications to the cloud backend. A detailed model of elastic execution between mobile client and cloud host is given in [5, 12].

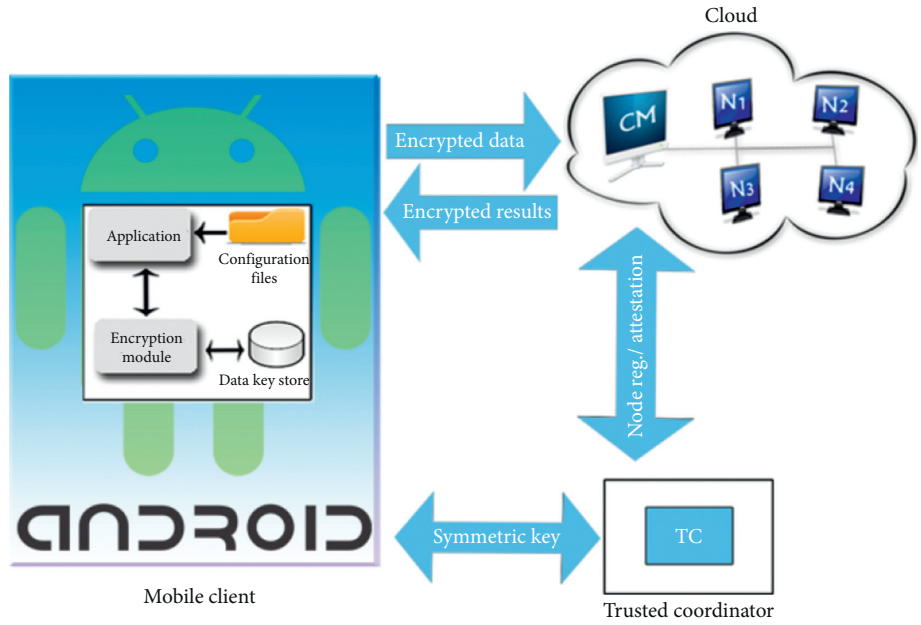


Figure 1: Overview of the system architecture.

4.1. Establishing Secure Channel. The data sent from the mobile client to the cloud nodes are encrypted using symmetric key cryptography. Owing to the security, fast speed, and low RAM requirements of AES, it is a suitable encryption algorithm for mobile clients; hence, we use AES key for encryption of the communication channel between mobile client and cloud. Moreover, AES supports encryption of large chunks of data without significant performance overhead at the client. The AES key (AES_{key}) used by the mobile client for encryption, however, is not readily made available to the cloud node receiving the encrypted message. The key is itself encrypted and a node can only gain access to it after it has successfully attested itself to the TC (explained in Section 4.3).

It is important to mention that we are interested not only in establishing a secure channel between the mobile client and the cloud node but also in restricting untrusted nodes from gaining access to the data. As the client does not know which node will be selected to process its data, we have not used SSL [26] which enables secure data communication between already trusting parties.

4.2. Registration of Cloud Nodes. In the proposed solution, registration of each cloud node is a required step that is performed before the node can be used in the cloud to perform any computations on behalf of the mobile clients. At the end of registration, the public part of an RSA key pair (TK_{Ni}^{public}, i.e., specific to a particular node) generated by a node and stored in its volatile memory is known to TC that is later used to exchange the key between node and TC as explained in next section. The reasons to store this key pair in volatile memory are (1) to require the node to get itself re-registered whenever its state is changed or is rebooted and (2) to prohibit the key to be used once a node's state is modified which is achieved because only the trusted applications are running on each node.

In our solution, both the cloud nodes ($N_1, \ldots, N_n$) and the trusted coordinator (TC) contain TPM chip for hardware attestation and hence can utilize the TCG remote attestation mechanism [19] along with registration of the cloud nodes. Instead of encrypting the complete Stored Measurement Log (SML) of the node (SML_{Ni}, i.e., specific to a particular node) and the TC (SML_{TC}), we encrypt the value of PCR-10 register containing the aggregate hash of the platform's state. This enables the receiving party to verify the SML of the sending party against the value of PCR-10 to check if a node maintains a trusted configuration [27]. Figure 2 shows the messages exchanged between the nodes and TC during the registration process.

Attestation Identity Key (AIK) is used for authentication purpose by the TPM to sign the PCR values and its corresponding public part is used by the other party to verify signatures. It is assumed that the public part of the AIK (AIK^{public}) of each node is already known to the TC and vice versa. Moreover, each node is also aware of a Trusted Key (TK_{TC}^{public}) which will be used to encrypt any data to be sent to TC that could be decrypted outside TPM, since the data encrypted with AIK can only be decrypted within the TPM.

When a node wishes to register itself with the TC, (1) it generates a nonce (n_{Ni}) and sends to the TC. Upon receiving the nonce, (2) TC signs the nonce along with the value of PCR-10 register using its AIK ($AIK_{TC}^{private}$). It also sends SML_{TC} and a nonce (n_{TC}) along with the encrypted content to the node. The node verifies signature of TC using AIK of TC (AIK_{TC}^{public}) and then verifies the SML of TC against the signed PCR-10 value. Since the nonce generated by the node (n_{Ni}) is also signed along with the PCR-10 contents by the TC, node can be sure that it is a fresh value reflecting the current configuration of the TC.

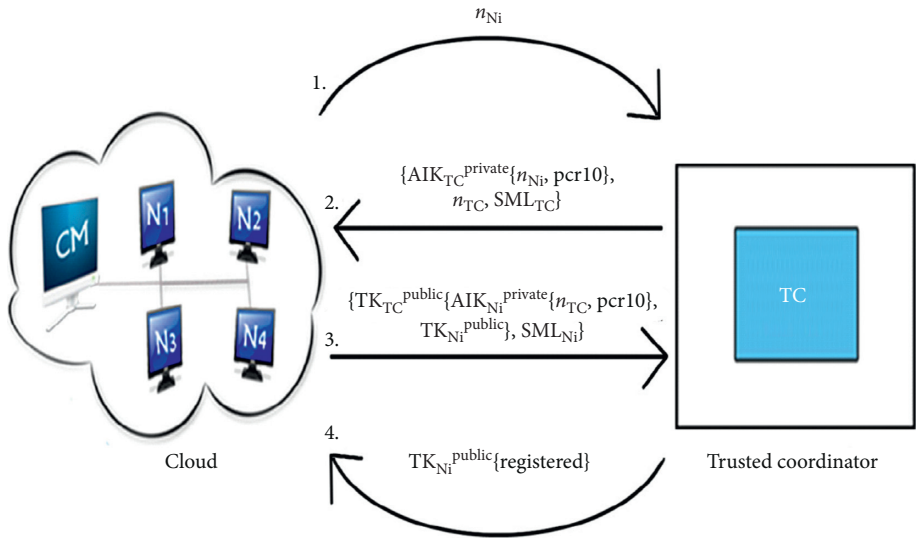


Figure 2: Registration of cloud nodes with the TC.

Afterwards, (3) the node sends the value of its PCR-10 and the nonce sent by TC (n_{TC}) signed by its AIK (AIK$_N$^{private}). It also sends the SML$_{Ni}$ so that TC can verify the SML$_{Ni}$ against PCR-10. The node also sends the public part of its Trusted Key (TK$_{Ni}$^{public}) to the TC which is stored with the TC and is used during node attestation as explained in the next section. All the message content, except for the SML$_{Ni}$, is encrypted using the TK$_{TC}$^{public} so that it is only accessible to the TC.

Upon receiving message (3), TC decrypts the contents using its TK$_{TC}$^{private}, verifies the signature on PCR-10, and matches the SML$_{Ni}$ against the PCR-10. TC also compares the signed nonce sent back by the node to ensure that the SML$_{Ni}$ are current. If all are verified, (4) TC sends the registration message signed by the newly received TK$_{Ni}$^{public} of the node. This signals the successful registration of the node with the TC along with the acceptance of the node's TK$_{Ni}$^{public}.

The node registration procedure is independent of the fact that the nodes host data/applications sent by a mobile client or a desktop client. Therefore, the notion of Trusted Clouds [19] can be utilized by the mobile clients in the same manner as other clients of cloud computing.

4.3. Trusted Computation on Cloud. This section explains the mechanism for carrying out trusted computation on the cloud on behalf of the mobile client. The mobile client, along with the cloud nodes, has access to the trusted key of TC, TK$_{TC}$^{public}. The TC also has the trusted keys of the nodes, TK$_{Ni}$^{public}. As mentioned in Section 4.1, the symmetric key (AES$_{key}$)—used by the mobile client for encrypting the data sent to cloud—is made available to the node after successful attestation by the TC. This enables the mobile client to off-load its data and applications to the cloud nodes in a secure and trusted manner.

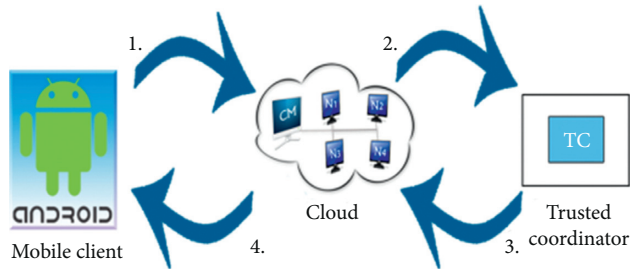
When a mobile client wishes to off-load the computation of a resource-intensive task to the cloud, (1) it encrypts the associated data using an AES$_{key}$ before sending it to the cloud. The mobile client also needs to communicate the AES$_{key}$ so that the receiving party can decrypt the information. The AES$_{key}$ is encrypted using the TK$_{TC}$^{public} restricting the cloud node to gain access to the AES$_{key}$ and hence the encrypted data, without first contacting the TC and proving that it is in trusted state. The mobile client also sends a nonce (n_U, i.e., nonce generated by the user) encrypted alongside the AES$_{key}$ to ensure the freshness of the communication.

Upon receipt of the encrypted message, the node must prove to the TC that it is trustworthy so that the TC may decrypt and send the AES$_{key}$ to the node. The various messages exchanged between communicating parties to achieve trusted computation are shown in Figure 3.

To access the AES$_{key}$ encrypted with the TK$_{TC}$^{pbulic}, (2) the cloud node passes on the encrypted key and nonce sent by the mobile user n_U to the TC. It also sends a nonce n_N and node information N to the TC using TK$_{TC}$^{pbulic}. In order to demonstrate the fact that the node is already registered with the TC, it computes the hash of the above parameters and signs it using node's trusted key TK$_{Ni}$^{private}. If the node is registered with the TC, TC is able to decrypt this part using TK$_{Ni}$^{public} of the node - provided at the time of registration (explained in Section 4.2)-decrypts the AES$_{key}$ and (3) sends back to the node the AES$_{key}$ along with n_U and n_N encrypted with the TK$_{Ni}$^{public} of the node. In order to ensure that the message is from the TC, TC also computes the hash of the contents and signs it using its TK$_{TC}$^{private}. In case the node is not already registered, it is first required to register itself with the TC as explained in Section 4.2.

Once the AES$_{key}$ is revealed to the node, it decrypts the message, performs the necessary computations on behalf of the mobile client, and (4) sends back the results encrypted by the same symmetric key (AES$_{key}$). The node also sends back the original nonce n_U to validate that the results are from a current computation.

It should be noted that the mobile device uses different AES$_{keys}$ for setting up secure communication channel for



1. $\{AES_{key}\{Data\}, TK_{TC}{}^{public}\{n_U, AES_{key}\}\}$
2. $\{TK_{Ni}{}^{private}\{SHA1\{TK_{TC}{}^{public}\{n_U, AES_{key}\}, n_{Ni}, N\}\}, TK_{TC}{}^{public}\{n_U, AES_{key}\}, TK_{TC}{}^{public}\{n_{Ni}, N\}\}$
3. $\{TK_{Ni}{}^{private}\{SHA1\{n_{Ni}, n_U, AES_{key}\}\}\}, \{TK_{Ni}{}^{public}\{n_{Ni}, n_U, AES_{key}\}\}$
4. $AES_{key}\{n_U, N, Result\}$

Figure 3: Secure data exchange between mobile client and cloud nodes.

Table 1: Threats vs. security mechanisms.

Threat	Catered	Mechanism
Threats on cloud node/TC		
Masquerading	Yes	Each node's AIK is already known to TC
Eavesdropping	Yes	Use of only trusted software ensures that no information can be manipulated or tempered with
Message tempering	Yes	
Malware	Yes	Malware cannot remain undetected since TC checks status of each node through remote attestation process
Message replay	Yes	The nonces used ensure freshness
Threats to mobile client		
Man in the middle	Yes	The processed data are being received encrypted which can only be accessed by the relevant mobile client
Result tempering	Yes	Since the data are encrypted, no one can alter the data undetectably

off-loading different computations. Therefore, if a node having access to a particular AES key is compromised, it will still not be able to decrypt the mobile devices' data for subsequent interaction.

5. Discussion

In the proposed architecture, establishment of secure communication channel ensures the confidentiality of customer data while it is under processing or in transit. Each software component loaded on cloud node is measured and extended in the TPM registers ascertaining that cloud node cannot hide or mislead about the software or code running on it. In our solution, the cloud node gets encrypted data while the key to decrypt it is encrypted with the trusted key of TC ($TC_{TK}{}^{public}$). In order to get the key to decrypt data, cloud node has to contact TC with another trusted key ($TK_{Ni}{}^{public}$, specific to each client). $TK_{Ni}{}^{public}$ is stored in the volatile memory at each node and is automatically destroyed whenever the configurations of a node changes or if the node is rebooted, requiring the node to re-register itself with TC. The use of nonce in the above protocols ensures freshness. Prior to revealing the key, TC on behalf of mobile client checks the cloud node, through remote attestation procedure, that it is in trustworthy state and is not executing any malicious software which can store or spy on mobile client data. Trusted software running at each node guarantees confidentiality of client data during computation.

The prototype implementation of the proposed system provides encouraging results; however, we believe that the performance can be further improved; hence, we are yet carrying out the experiments. From the initial results, we are convinced of the validity and usefulness of our proposed solution. The overhead caused is perfectly tolerable for large sized data, as the time taken for data processing is directly proportional to the size of the data to be processed, at the cost of security and processing provided for resource-starved devices. These factors provide strong basis for the practical implication of our proposed architecture. We provide detailed security analysis of the system by evaluating the system against the known threats at each stage in Table 1.

We have catered to the major threats on cloud node and on TC and the threats to mobile client by using the standard security mechanisms while exploiting trusted computing. The proposed solution provides end-to-end secure communication between mobile client and cloud by employing TPM-based remote attestation of cloud nodes ensuring the confidentiality and integrity of the computations performed by the cloud nodes. This demonstrates that the mobile devices can securely augment their capabilities by off-

loading computation to cloud. Moreover, our approach is readily useful for desktop clients lacking support for hardware-based attestation.

6. Conclusions and Future Direction

We have defined a mechanism for secure exchange of data between mobile devices and cloud hosts. It guarantees that only trusted cloud nodes are allowed access to the mobile device's data addressing the concerns related to confidentiality and integrity of the computations performed by the cloud node. By utilizing the concepts of elastic execution [9] and trusted clouds [12] and applying our model for secure communication and trust establishment, a comprehensive end-to-end security infrastructure has been provided for mobile clients of cloud computing.

The security architecture is tailored to the capabilities of the participating entities. At the mobile device's end, we have employed the AES symmetric key encryption to minimize the encryption overhead. Cloud nodes, on the other hand, are attested by a trusted coordinator TC to establish that they are in a secure and trusted configuration before gaining access to the symmetric key. Thus, only trusted nodes are selected to carry out the required computation on behalf of the mobile device.

The probability of sending any malicious data by rogue mobile clients is high, which can be harmful to the cloud and at the same time can steal the data of other customers, hence compromising the privacy and breaching the security of cloud [28]. Thus, future researchers would find it interesting to probe the dimensions of attesting the validity and reliability of the mobile clients. This will open a vast field of discussion for researchers to take into account protocols that enable mobile device verification prior to granting them access to cloud resources.

Recently, a new usage scenario that encompasses mobile applications and the cloud is emerging, that is, "testdriving" a potential application on the cloud prior to purchasing it. Consider the case of Amazon testdrive [29, 30] that allows users to preview any Android application in the App Store directly from their browsers by launching an emulated instance of Android on its EC2 cloud, allowing direct control from the browser (using Flash). The reader can easily imagine the usefulness of TAME^c in securing licensing and customized features in such a scenario. In fact, there are reports [31] that one of the primary reasons testdrive was disabled as soon as it was launched was fear of insecure data exchange.

In addition to implementing a complete prototype of our proposed system, a future direction could be identifying the failure models of the trust establishment mechanism itself. An important question arises that if a nontrusted node receives the encrypted data and is not able to attest itself to the TC, how will the mobile client be notified. In this case, there should be an upper bound on the time after which the mobile device should stop expecting to receive results from the cloud node. The TC may also notify the mobile device of a failed attestation attempt by a node. However, the issue still remains if the node receiving the data never contacts the TC. This and similar concerns are not necessarily specific to the mobile clients and offer an interesting area to explore. We hope that research in this direction will give useful insight for further enhancing the proposed security infrastructure.

Data Availability

The data used to support the findings of this study are provided within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the China University of Petroleum-Beijing and Fundamental Research Funds for Central Universities under grant no. 2462020YJRC001.

References

[1] H. Yih-Chun and A. Perrig, "A survey of secure wireless ad hoc routing," *IEEE Security & Privacy*, vol. 2, no. 3, pp. 28–39, 2004.

[2] Android market, 2021, https://play.google.com/store/apps.

[3] M. Armbrust, A. Fox, R. Griffith et al., "Above the clouds: a berkeley view of cloud computing," Technical Report No. UCB/EECS-2009-28, University of California, Berkeley, CA, USA, 2009.

[4] B. G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proceedings of HotOS'09: 12th Workshop on Hot Topics in Operating Systems*, Monte Verità, Switzerland, May 2009.

[5] Google Apps, 2021, https://play.google.com/store/apps/.

[6] Amazon Elastic Compute Cloud (Amazon EC2), 2021, https://aws.amazon.com/ec2.

[7] M. B. Mollah, M. A. Kalam Azad, and A. Vasilakos, "Security and privacy challenges in mobile cloud computing: survey and way ahead," *Journal of Network and Computer Applications*, vol. 84, pp. 38–54, 2017.

[8] Trusted platform module, 2021, https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary.

[9] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in *Proceedings of the 2009 Conference on Hot topics in Cloud Computing: HotCloud'09*, Berkeley, CA, USA, June 2009.

[10] Mcafee, 2021, https://www.mcafee.com/en-us/index.html.

[11] Android market statistics, 2021, http://www.androlib.com/appstats.aspx.

[12] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: elastic execution between mobile device and cloud," in *Proceedings of the 6th European Conference on Computer Systems (EuroSys 2011)*, Salzburg, Austria, April 2011.

[13] Android DalvikVM specs, 2021, https://source.android.com/devices/tech/dalvik.

[14] G. Cheng and A. K. Ohoussou, "Sealed storage for cloud computing," in *Proceedings of the International Conference on Computer Design and Applications (ICCDA)*, Qinhuangdao, China, June 2010.

[15] X. Jin, X. Chen, C. Zhao, and D. Zhao, "Trusted attestation architecture on an infrastructure-as-a-service," *Tsinghua Science and Technology*, vol. 22, no. 5, pp. 469–477, 2017.
[16] Mobile Trusted Module, 2021, https://trustedcomputinggroup.org/resource/mobile-phone-work-group-mobile-trusted-module-specification/.
[17] M. Nauman, S. Khan, X. Zhang, and J. P. Seifert, "Beyond kernel-level integrity measurement: enabling remote attestation for the android platform," in *Proceedings of the 3rd International Conference on Trust and Trustworthy Computing (Trust 2010)*, Berlin, Germany, June 2010.
[18] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in *Proceedings of 19th ACM Symposium on Operating Systems Principals (SOSP '03)*, Bolton Landing, NY, USA, October 2003.
[19] Trusted Computing Group, *TCG Pc Specific Implementation Specification*, Trusted Computing Group, Beaverton, OR, USA, 2003.
[20] C. Mitchell, *Trusted Computing (Professional Applications of Computing)*, IEEE Press, New York, NY, USA, 2005.
[21] TPM Main Specification Level 2 Version 1.2, 2021, http://www.trustedcomputinggroup.org/resources/tpm_main_specification.
[22] R. Sailer, T. Jaeger, X. Zhang, and L. van Doorn, "Attestation-based policy enforcement for remote access," in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*, Washington, DC, USA, October 2004.
[23] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 193–206, 2003.
[24] S. Berger, K. G. Goldman, R. Caceres, R. Perez, R. Sailer, and L. van Doorn, "vTPM: virtualizing the platform module," Technical Report RC23879, IBM Research, Yorktown Heights, NY, USA, 2006.
[25] E. Brickell, C. Jan, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*, Washington, DC, USA, October 2004.
[26] Guide to SSL VPNs, 2021, http://csrc.nist.gov/publications/nistpubs/800-113/SP800-113.pdf.
[27] R. Sailer, X. Zhang, T. Jaeger, and L. V. Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *Proceedings of the 13th Conference on USENIX Security Symposium (SSYM 04)*, Berkeley, CA, USA, August 2004.
[28] E. T. Ristenpart, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security CCS'09*, Chicago, IL, USA, November 2009.
[29] D. Etherington, Amazon's TestDrive is the real strength of appstore, 2021, https://gigaom.com/2011/03/28/amazons-testdrive-is-the-real-strength-of-appstore/.
[30] J. Kincaid, Amazon's android App store launches: test drive Apps directly from your browser, 2021, https://techcrunch.com/2011/03/22/amazon-android-app-store-3/.
[31] S. Anthony, Amazon Appstore for Android Test Drive Hands on: Surprisingly Cool, but Still US-Only, 2021, http://downloadsquad.switched.com/2011/03/28/amazon-appstore-for-android-test-drive-hands-on-surprisingly-co.