DMA: Mutual Attestation Framework for Distributed Enclaves

Peixi Li[1], Xiang Li[2(✉)], and Liming Fang[1,3(✉)]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China
fangliming@nuaa.edu.cn
[2] City University of Hong Kong, Hong Kong, China
sean.lixiang97@gmail.com
[3] Shenzhen Research Institute, Nanjing University of Aeronautics and Astronautics, Shenzhen 518000, China

Abstract. Remote attestation is a key mechanism for establishing trust between multiple enclaves that require interaction. Existing approaches establish trust by integrating remote attestation and TLS handshakes with the help of a centralised attestation service. However, such an approach lacks freshness and struggles to overcome advanced network attacks. In addition, over-centralised trust determination cannot satisfy scenarios requiring frequent attestations, even leading to incorrect trust decisions. For this reason, we propose DMA, which provides strong freshness binding of the attestation evidence and uses consensus algorithms to ensure balanced trust across network domains. We implement a prototype to demonstrate the scalability and efficiency resulting from the decentralised design of DMA.

Keywords: Remote attestation · Intel SGX · Distributed system.

1 Introduction

Trusted Execution Environment (TEE), such as Intel SGX, combines hardware and software to provide confidentiality and integrity protection for data and code in isolated containers known as enclaves. In contrast to cryptographic methods such as MPC [14], TEE-based secure computing offers a departure from over-reliance on complex cryptographic primitives. It demonstrates the capacity to approach plaintext computation's performance in numerical and logical scenarios, particularly in large-scale specialised and general-purpose computational tasks tailored to distributed environments. This property has attracted considerable attention in various domains, such as machine learning [6,16], data analytics [11,22] and cloud computing [32].

In these application scenarios, it is often necessary to share data among different enclaves. Relying on remote attestation, enclaves can establish mutual trust to facilitate such interactions. Simply speaking, attestation is the process of verifying an enclave's identity to another party, confirming its protection by a

truly TEE-supported platform. Furthermore, attestation helps to create secure communication channels between enclaves, thus guaranteeing the confidentiality and integrity of the data during transmission.

RA-TLS [20] and RATS-TLS [7] present a generic approach with the help of TLS to enhance remote attestation. To specify, an enclave generates a key pair at startup and then produces evidence (i.e., the SGX quote) containing the hash of the public key. This quote is embedded in a TLS certificate, allowing the opposite enclave to verify it against an attestation service and make a trust decision. The roles of the two enclaves are then exchanged to establish mutual trust. This process is repeated to establish trust among all enclaves. However, such a scheme still has some issues that need to be addressed.

Lack of evidence freshness. The quote used for attestation is pre-generated and can be reused when the channel is re-established. This way, the quote is bound to the enclave but not the channel. A more sophisticated adversary could intercept and replay [10] quote obtained during attestation through man-in-the-middle attacks. Furthermore, the adversary can perform a relay attack [13] by impersonating an intermediate entity. This could deceive the enclave and result in the disclosure of private data within the enclave.

Over-centralised trust determination. In remote attestation, trust determination between enclaves relies heavily on a specific verifier, known as the attestation service. Note that trust determination is over-centralised, as the attestation service is usually controlled by a specific entity, such as a cloud service provider (e.g. Azure [24]) or the manufacturer of the TEEs (e.g. Intel [18]). As noted by [33], the monopolisation of trust can potentially obscure the judgment of the enclave relying party, leading to incorrect trust decisions. Centralised trust determination is even unacceptable in computing scenarios similar to MPC.

Lack of scalability and efficiency. As the number of enclaves required for computing increases, so does the complexity of the trust relationships that need to be managed. This trend is particularly significant in scenarios that need flexible allocation of computing resources. Enclaves often require frequent access to the attestation service to establish trust. However, the centralised attestation service may cause performance issues and hinder system efficiency.

In this paper, we propose DMA, a Decentralised Mutual Attestation framework, to address these issues. Our approach allows distributed enclaves to establish mutual trust in a more secure and flexible manner. Specifically, our contributions are as follows:

- We provide a robust freshness preservation mechanism to prevent evidence reuse and trust spoofing by binding evidence to each TLS handshake with a revocation mechanism.
- We use consensus algorithms to transfer enclave trust across multiple network domains, avoiding the use of a unique attestation service, thus ensuring a balanced level of trust.
- We provide a decentralised attestation framework for distributed enclaves, allowing flexible trust establishment and management during multi-party interactions.

The rest of the paper is structured as follows. Sec. 2 provides background on attestation, secure channel techniques and consensus algorithms. Sec. 3 describes the system architecture, security model and detailed designs. Sec. 4 explains the workflow. Sec. 5 performs security review and analysis. Sec. 6 presents implementation and evaluation. Sec. 7 discusses some design considerations. Sec. 8 presents relevant studies and Sec. 9 summarises our work.

2 Background

2.1 Attestation in Intel SGX

Intel SGX provides two forms of attestation: local attestation and remote attestation. In order to feature attestation, Intel provides two measurement systems using the hardware-supported SHA-256 algorithm to create cryptographic identities for enclaves. The Enclave Measurement (MRENCLAVE) describes information about the code and initial data residing in the enclave, including their expected order, page position, and the security attributes of these pages. The Sealing Authority (MRSIGNER), on the other hand, is linked to the enclave's authorship, typically represented as the hash value of the public key used to sign the enclave before distribution.

Local attestation enables an enclave to create a signature structure known as a report, which another enclave can verify to demonstrate that both of them have been established on the same platform. The report, generated by the hardware, contains the measurements, enclave attributes, hardware TCB integrity, and a custom string called UserData. The attester enclave signs the report by generating a MAC using a platform-independent key, which is accessible solely to that enclave and the target enclave. The target enclave then verifies the report by computing the MAC afresh, thereby making a trust decision based on the information contained in the report.

The Remote ATtestation procedureS (RATS) architecture [1] describes remote attestation as the process by which an attester produces evidence of its trustworthy information, allowing a relying party located at the remote end to decide whether to trust the entity with the help of endorsements and a verifier. Intel provides a particular enclave known as the Quoting Enclave (QE) to verify other enclaves' reports through local attestation and creates a signature with a device-specific asymmetric private key in place of the MAC in the report. The output is called a quote. The quote is then transmitted to a remote relying party, which can request an attestation service holding the public key to verify the quote and make a trust decision on that enclave with the verification report signed by the attestation service.

Intel supports two types of remote attestation. The first is based on Intel Enhanced Privacy ID (Intel EPID [19]). In this scheme, the attestation service is a centralised system known as IAS. This scheme protects the privacy of user devices by leveraging the anonymity, unforgeability, and revocation capabilities of the EPID group signature algorithm [2]. To better support remote attestation

within on-premise networks such as data centres, Intel introduces a more flexible scheme based on Intel Data Center Attestation Primitives (Intel DCAP [30]). This scheme enables third-party service providers to verify an enclave's quote using public-key infrastructure (PKI), enabling trust decisions independently of Intel at runtime.

2.2 Secure channel between enclaves

It is crucial to protect data from potential threats during transmission to facilitate secure collaboration among enclaves. Transport Layer Security (TLS) emerges as a widely recognised industry standard for secure communications that uses highly secure encryption algorithms to ensure confidentiality.

TLS facilitates authentication at both ends of the communication channel by exchanging X.509 digital certificates, a process known as the TLS handshake. In a typical client-to-server interaction scenario, the client initiates the handshake by sending a ClientHello message containing a random number. Upon receiving the ClientHello message, the server responds with a ServerHello message carrying another random number. These random numbers are intended to prevent replay attacks [8,29]. The server then sends a certificate that contains its public key and identity information. The client can verify the legitimacy of the server's identity by checking the certificate, the Certificate Authority's (CA) signature, and the expiration date. TLS also offers the option of a two-way handshake by requiring the client to provide a certificate, enhancing both entities' trustworthiness.

To protect inter-enclave communication from man-in-the-middle (MITM) attacks, RA-TLS [20] and RATS-TLS [7] propose using the handshake process described above to integrate the remote attestation process with the existing TLS protocol. This entails generating a pair of public and private keys at enclave startup, binding the public key within this key set to a report material during local attestation, requesting an attestation verification report from IAS using a quote generated by QE, and employing an X.509 certificate embedded with the attestation verification report as the certificate of the inter-enclave channel.

In contrast to the conventional paradigm, the RA-TLS approach eliminates the need for a CA by making the SGX the root of trust (RoT). It uses self-signed certificates created with the private key generated during enclave startup. Both public and private keys are confined within the enclave, ensuring the private key is never exposed outside and only accessible to code within the enclave.

2.3 Consensus algorithms

Consensus algorithms are widely used to enable consistent decision-making among multiple nodes about a value or a set of operations without central control. Typically, fault tolerance consensus algorithms can tolerate some node failures in the system and continue to operate normally. The classic algorithms are Paxos [21] and Raft [28]. Byzantine tolerant algorithms go further to ensure consistency in the event of possible malicious behaviour of a node.

3 System Design

3.1 DMA Architecture

The architecture of DMA is illustrated in Fig. 1. In each network domain, we assume all nodes are classified into two types: authentication and user nodes. The former is uniquely present in each domain, while the latter can exist in one or more than one. We introduce two DMA architectural enclaves: the authentication enclave (AuthE) and the attestation enclave (AttestE). AuthEs are placed on authentication nodes, while AttestEs are placed on user nodes. In addition, function-specific user-level enclaves, referred to as user enclaves (UserEs), are also deployed in user nodes.

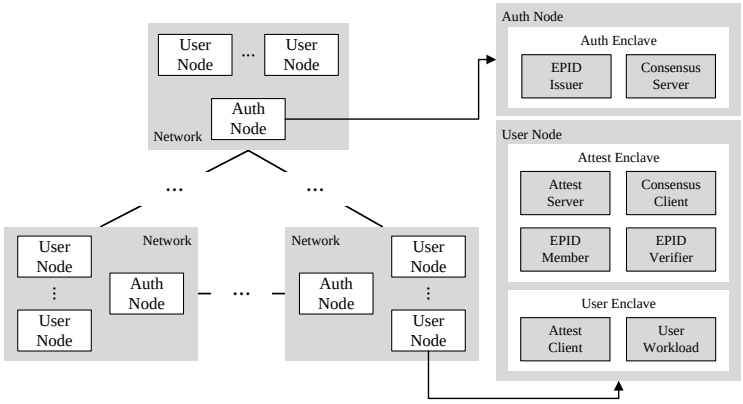


Fig. 1. Architecture of DMA

DMA manages trust across network domains using the EPID group signature scheme and consensus algorithms. The AuthEs are responsible for managing a unified EPID group, issuing EPID memberships to the AttestEs, and synchronising EPID group identities and revocation lists across all AuthE peers. On the other hand, the AttestEs are responsible for providing endorsements to UserEs using their EPID memberships and verifying those created by other AttestEs.

3.2 Security model

We assume that the SGX hardware can maintain the integrity and confidentiality of the code and data within the enclave while in use, as designed by Intel. While certain attacks [25,31,3,17] may expose the contents of the enclave's encrypted memory, the SGX platform can mitigate these vulnerabilities through TCB recovery [30]. Consequently, the latest evaluation results of the hardware obtained from Intel will be cached to evaluate the hardware TCB. As SGX is designed, the host application and the operating system are not trusted.

For components in Intel's chain of trust, such as Intel Provisioning Enclave (PvE), QE, and IAS, which support EPID-based remote attestation, and Provisioning Certification Service (PCS), QE3 (distinct from QE above), and Quote Verification Enclave (QvE), which support DCAP-based remote attestation, we trust them to honestly follow predefined protocols to correctly identify the SGX hardware and the enclave running on it. However, as OPERA [5] points out, they may gather some information about the SGX platforms and the enclaves during attestation verification.

We consider an adversary $\mathcal{A}$ who is interested in AuthEs running on a set of $n = 2f + 1$ SGX-enabled machines, and at any given time, adversary $\mathcal{A}$ can control at most f of these machines. Adversary $\mathcal{A}$ can start, pause, resume, and terminate enclaves at his will to compromise the state within them. Furthermore, we assume a Dolev-Yao [9] adversary $\mathcal{A}$ attempts to spoof the attestation process, replay used quotes, snoop on private data, or manipulate the results of DMA. Denial of Service (DoS) attacks are not considered.

3.3 Establish trust between enclaves

The remote attestation protocol used in our design can be categorised into two types. The original SGX remote attestation establishes trust between DMA architectural enclaves, while DMA remote attestation facilitates trust between UserEs. We integrate the attestation process with the TLS handshake, conceptualising the TLS handshake as a form of attestation. This integration eliminates the need for two-way attestation after the TLS channel is established, thereby improving efficiency and, more importantly, helping to enhance security during the trust establishment.

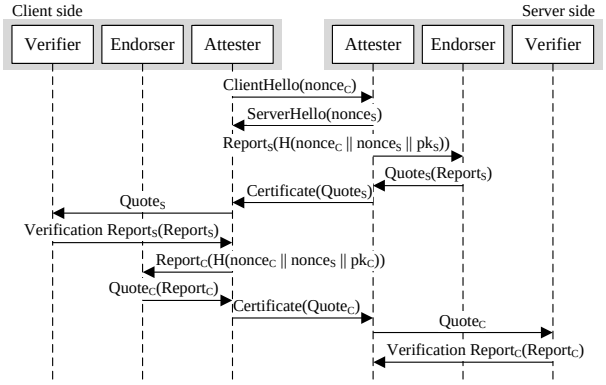


Fig. 2. Attestation process with TLS handshake[1]

[1] In the original SGX attestation, the Endorser is QE, and the Verifier is IAS. In DMA attestation, the Endorser and the Verifier are both AttestEs.

The attestation process with TLS handshake is shown in Fig. 2. After startup, each enclave generates a pair of public and private keys (pk, sk) in its encrypted memory and regenerates them upon restart. During quote generation, the public key pk is included in a manifest (i.e., the UserData), along with the temporary nonces $nonce_C$ and $nonce_S$ exchanged in the ClientHello and ServerHello messages. The enclave then computes a SHA-256 hash value $h = H(nonce_C||nonce_S||pk)$ as the exact UserData and asks the hardware to generate an SGX report against itself.

After that, a DMA architectural enclave can request the Intel QE on its platform to sign an SGX quote, while a UserE can request the AttestE to sign a DMA quote. The quote is then embedded as an extension field in an X.509 certificate structure, signed with its private key sk, and sent to the peer. The receiving enclave retrieves the peer's quote from the specific extension field, recomputes the SHA-256 hash values, and verifies the consistency between the hash value in the quote and its recomputed once.

In contrast to pre-handshake attestation protocols such as RA-TLS [20], which only embed the enclave public key into the quote, DMA also embeds the channel's nonce. In addition, obtaining IAS-signed attestation verification reports shifts from server-side to client-side execution. This shift allows the channel's nonce to be passed to the IAS, thereby including it in the signed response. These initiatives enable the binding of the quote to the enclave and the channel, thus enhancing the quote's freshness and avoiding its reuse.

3.4 Transferring distributed trust

To facilitate the transfer of trust across different network domains, it is imperative to negotiate a unified EPID group encompassing the entire network. Consequently, any network domain integrated into the DMA can effectively manage trust by leveraging the collectively negotiated EPID group. This approach aims to decentralise the authority of the EPID group across different network domains, thereby ensuring an equitable distribution and balance of trust.

A consensus algorithm is used to determine the creator of the EPID group identity among all AuthE peer nodes. This process selects a representative node from the distributed AuthE cluster, similar to a leader election. Once the EPID group is initialised by the chosen creator, the group identity, consisting of the group public key and group private key, is synchronised across all AuthE nodes within the cluster. To this end, we use the leader election algorithm within the Raft protocol [28] to orchestrate this task.

In tandem with establishing a unified EPID group across the network, maintaining network-wide synchronised revocation lists (RLs) is also crucial to transfer trust. RLs record pertinent pieces of information, such as the signers' private keys and the quote's signatures, to manually flag an enclave or a quote as untrustworthy, thereby safeguarding the overall security and integrity of the system, particularly in scenarios where enclave behaviour poses potential threats. The considerations of the revocation mechanism are described in detail in Sec. 3.5.

Algorithm 1 Lightweight EPaxos for synchronising revoked private keys and signatures

Phase 1

Any replica L, **on receive** $\langle Req, \gamma \rangle$ **from** client
  becomes the command leader of γ
  $i_{L,I} \leftarrow i_{L,I} + 1$ ▷ increment instance id
  $cmds_L[i_L][i_{L,I}] \leftarrow (\gamma,$ **accepted**$)$
  send $\langle Acc, L, i_L, i_{L,I}, \gamma \rangle$ to quorum Q
  ▷ resend if not receiving any $AccOK$ in a while
Any replica R, **on receive** $\langle Acc, L, i_L, i_{L,I}, \gamma \rangle$
  ▷ record instance id for state recovery
  $seq_R[i_L] \leftarrow \max(seq_R[i_L], i_{L,I})$
  $cmds_R[i_L][i_{L,I}] \leftarrow (\gamma,$ **accepted**$)$
  $t_{Rec}[i_L][i_{L,I}] \leftarrow timer()$
  reply $\langle AccOK, i_{L,I} \rangle$ to replica L

Phase 2

Replica L, **on receive** at least $\lfloor N/2 \rfloor$ $\langle AccOK, i_{L,I} \rangle$
  $cmds_L[i_L][i_{L,I}] \leftarrow (\gamma,$ **committed**$)$
  send $\langle Done, \gamma \rangle$ to client
  send $\langle Com, L, i_L, i_{L,I}, \gamma \rangle$ to all replicas
  ▷ resend if not receiving N $ComOK$ in a while
Any replica R, **on receive** $\langle Com, L, i_L, i_{L,I}, \gamma \rangle$
  clear timer $t_{Rec}[i_L][i_{L,I}]$
  $cmds_R[i_L][i_{L,I}] \leftarrow (\gamma,$ **committed**$)$
  reply $\langle ComOK, i_{L,I} \rangle$ to replica L

Phase 3

Replica R, **on** $t_{Rec}[i_L][i_{L,I}]$ **timeout**
  if $cmds_R[i_L][i_{L,I}]$ **not** committed **then**
    take over as the command leader
    $\lambda \leftarrow cmds_R[i_L][i_{L,I}]$
    send $\langle TryAcc, R, i_L, i_{L,I}, \lambda \rangle$ to quorum Q
    ▷ resend if not receiving any $TryAccOK$ in a while
Any replica O, **on receive** $\langle TryAcc, R, i_L, i_{L,I}, \lambda \rangle$
  ▷ record instance id for state recovery
  $seq_O[i_L] \leftarrow \max(seq_O[i_L], i_{L,I})$
  $cmds_O[i_L][i_{L,I}] \leftarrow (\lambda,$ **accepted**$)$
  reply $\langle TryAccOK, i_L, i_{L,I} \rangle$ to replica R

Phase 4

Replica R, **on receive** at least $\lfloor N/2 \rfloor$ $\langle TryAccOK, i_L, i_{L,I} \rangle$
  $cmds_R[i_L][i_{L,I}] \leftarrow (\lambda,$ **committed**$)$
  send $\langle TryCom, R, i_L, i_{L,I}, \lambda \rangle$ to all replicas
  ▷ resend if not receiving N $TryComOK$ in a while
Any replica O, **on receive** $\langle TryCom, R, i_L, i_{L,I}, \lambda \rangle$
  $cmds_O[i_L][i_{L,I}] \leftarrow (\lambda,$ **committed**$)$
  reply $\langle TryComOK, i_L, i_{L,I} \rangle$ to replica R

In our design, the consensus algorithm is also used to synchronise revoked private keys and signatures in different network domains to the entire network. We utilise a lightweight EPaxos [26] consensus algorithm, characterised by a two-round commit scheme augmented with a retry mechanism to ensure accurate and timely synchronisation of all revocation proposals across the cluster, as described in Algorithm 1.

3.5 Revocation mechanism

When a UserE is disconnected, the quote used during the attestation process can be exploited by malicious actors to perform replay attacks. To mitigate the risk, it is essential to revoke the signature of the quote used in the attestation process with the disconnected UserE, thereby preventing the quote from being reused in subsequent attestation processes. Moreover, in cases where a AttestE is compromised, exposing its EPID member identity and posing a risk of exploitation, it is necessary to revoke that AttestE so that it can no longer participate in the attestation process.

In the first case, the UserE counterpart at the opposite end of the channel can propose a revocation request with the used quote to the AttestE within its platform. The AttestE then forwards the request to the AuthE within its network domain. In the second case, the AuthE could internally initiate a revocation request using the compromised AttestE membership. Through the consensus algorithm within the AuthE cluster, values to be revoked are synchronised across all AuthEs.

The signature-based revocation in our design focuses primarily on ensuring the non-reusability of the quote associated with the signature, thereby avoiding

potential advanced replay attacks during the trust establishment. However, in the EPID scheme, signature-based revocation allows an issuer to revoke a member's signature capability based on a signature generated by that member if the issuer does not know the member's private key. Therefore, we redesigned the signature revocation list and inserted our revocation verification process before the signature verification process introduced by the EPID scheme. However, for the revocation of group members, we still follow the EPID scheme, enabling the revocation of the member via the member's private key and its signed EPID signatures.

Since the values to be revoked can come independently from any AuthE or AttestE in different network domains, there is no dependency between each value, and revocation does not need to be performed in any particular order. Such a feature leads to the intuitive insight that we can use a two-dimensional log table to record each value in the AuthE. As described in Algorithm 1, the first dimension represents different AuthE nodes, while the second dimension denotes the values proposed to be revoked by each AuthE. By tagging the revocation type, these values can be used in different ways to perform revocation once the AuthE cluster reaches consensus.

After consensus is reached within the AuthE cluster, the values to be revoked are stored within the AuthEs using signature revocation lists (SigRLs) and private key revocation lists (PrivRLs). In the meantime, when an EPID member receives a signature request or an EPID verifier receives a signature verification request, the latest SigRL and PrivRL are retrieved from the AuthE to ensure that the signing and verification process can be performed correctly and securely.

3.6 State recovery

The state of the DMA system includes elements such as the EPID group identity, the revoked private keys, and the revoked signatures in AuthEs within each network domain. We still use the recovery mechanism in the consensus algorithm to repair the inconsistent state due to crashes since the state held by each AuthE depends on the latest state in all network domains connected to DMA. In addition, the consensus algorithm in AuthE only needs to satisfy the fault tolerance requirement because, in our security model, an adversary cannot break the protection of SGX and control AuthE to send false messages to other nodes in the cluster.

Regarding the identity of the EPID group, when a non-leader node crashes, it can still receive the heartbeat message broadcasted by the current leader upon restarting. Conversely, if a leader node crashes, the remaining AuthEs will again elect a new leader using the leader election algorithm and broadcast the already recognized group identity. For private keys and signatures revoked by the AuthE cluster, the crashed node will proactively retrieve the latest log table from the other AuthEs upon restart and immediately commit those values locally. This way, the already created group identity, the revoked private keys and signatures can always be restored to the crashed AuthE.

4 Workflow

4.1 Overview

The workflow of DMA encompasses three distinct phases: preparation, provisioning, and attestation.

- The **preparation phase** initiates all AuthEs and establishes mutual trust between them across diverse network domains using the original SGX remote attestation protocol. Subsequently, a unified EPID group is created through the consensus algorithm.
- The **provisioning phase** initialises all AttestEs. Once mutual trust is established between AttestEs and AuthEs within their respective network domains, each AttestE becomes a member of the EPID group. Then, each AttestE creates the necessary EPID member and EPID verifier components to facilitate subsequent attestation processes.
- The **attestation phase** initialises all UserEs. After completing local attestation procedures with the AttestEs within its platform, each UserE requests quotes signed by the AttestE. Finally, each UserE contacts its peers within the current or other network domains and exchanges quote materials to establish mutual trust.

4.2 Preparation phase

The process of the preparation phase is shown in Fig. 3. After launching on the authentication node platform through an untrusted host application, each AuthE establishes connections with its peer entities in other network domains. This connection results in mutual trust between all AuthEs, achieved through the TLS handshake with original SGX remote attestation.

After completing the TLS handshake between all AuthEs, each AuthE initialises a random timer, a key component outlined in the Raft leader election algorithm. Once the timer expires, such AuthE increments its internal term number, votes for itself, and sends leader election requests to all other AuthEs in the cluster. If other AuthEs have yet to vote during the current term, they will vote for the first leader election request received. When the votes of a majority of AuthEs have accumulated, the designated AuthE becomes the first leader within the cluster.

The first leader AuthE then performs several tasks. First, it initialises an EPID issuer context within its enclave's encrypted memory. Additionally, the first leader distributes the identity information of the EPID group to all AuthEs within the cluster via a broadcast message. Upon receiving this broadcast message, the recipient AuthEs extract the group public key and group private key, thereby creating their local EPID issuer context. Finally, each AuthE initialises an empty signature revocation list and a private key revocation list. Completion of these steps signifies the transition of the AuthE to a ready state.

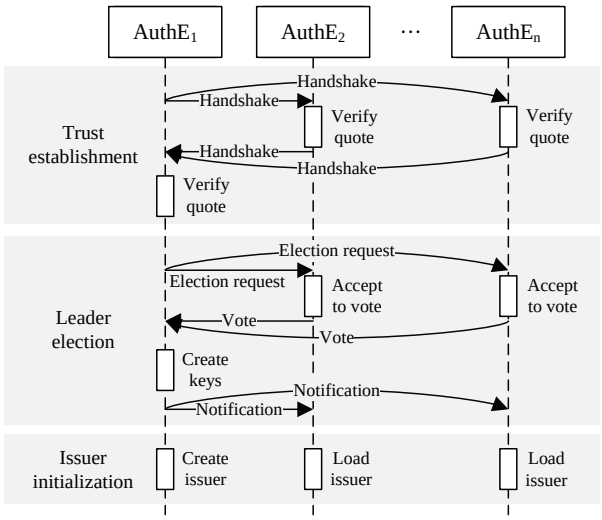


Fig. 3. Process of the preparation phase

4.3 Provisioning Phase

The provisioning phase process is described in Fig. 4. Each AttestE is launched on the user node platform through an untrusted host application. After startup, each AttestE establishes connectivity with the AuthE located within the same network domain. They then complete a two-way TLS handshake, establishing mutual trust. Hardware TCB evaluations obtained from Intel also cached while in the handshake.

Next, each AttestE generates a random private key f within its enclave memory, representing its EPID group membership. It then requests a challenge containing a random number $nonce_I$ from the AuthE. Using this received challenge and its private key f, the AttestE computes a join request in response to the challenge and sends it to the AuthE. Upon receipt of the join request, the AttestE verifies against the challenge response and then generates a membership credential for the AttestE. Details on the above join protocol can be found in Sec. 4.4 of [2]. At this point, the AttestE becomes a member of the EPID group organised by the AuthE cluster.

In the subsequent phase, the AttestE obtains the public key pk_G corresponding to the EPID group from the AuthE. Using this key, the AttestE initialises both an EPID member context and an EPID verifier context within its enclave memory. Upon completion of the provisioning phase, the AttestE enters a ready state, available to provide attestation services to any UserEs within its domain.

4.4 Attestation Phase

Each UserE is launched on the user node platform through an untrusted host application. Initially, the UserE retrieves the MRENCLAVE measurement of the

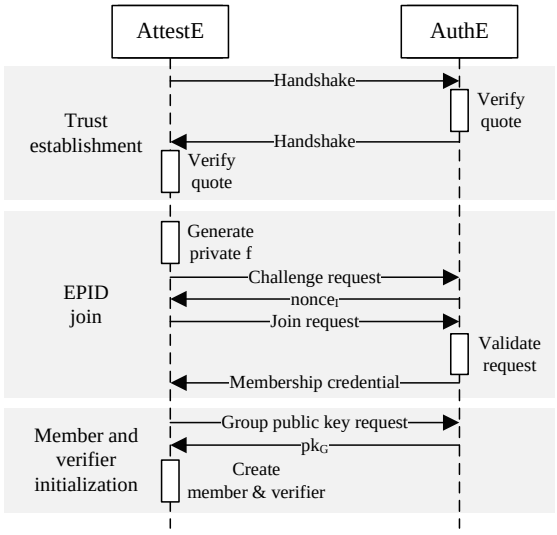


Fig. 4. Process of the provisioning phase

AttestE residing on its platform. Using this measurement as UserData, the UserE generates an SGX report, which is then sent to the AttestE. This local attestation process verifies that the UserE and the AttestE are on the same platform.

Following the completion of the local attestation, the UserE connects to its peer, creates another SGX report, and seeks the signature of the AttestE residing on its platform. Since the AttestE can trust the UserE, it signs this report using its private key from the EPID member, thereby generating a DMA quote that is returned to the UserE.

The UserE then embeds the DMA quote in an X.509 certificate, self-signs it, and sends it to its peer for the TLS handshake. Upon receipt, the peer extracts the DMA quote and forwards it to the AttestE on its platform. The AttestE retrieves the latest revocation lists from the AuthE, verifies the signatures of the DMA quote using the group public key via the EPID verifier, and generates a verification report for the UserE.

Finally, the UserE evaluates the trustworthiness of its peer entity based on the verification report, which consists of examining attributes such as the peer entity's MRENCLAVE, MRSIGNER, and other enclave attributes, including the TCB version and the debug status. Upon completing the verification process, the TLS handshake is completed, enabling the UserE to securely transmit sensitive data to its peer entity.

The procedure outlined above is similar to the one depicted in Fig. 2. However, it differs from the handshake process between DMA architectural enclaves, as it involves different entities used by UserEs for quoting and verification. That is, the endorser and the verifier utilised by the UserEs are the AttestEs, but the QE and IAS are for DMA architectural enclaves.

5 Security Analysis

In this section, we review the security properties required for DMA and analyse how our design satisfies these properties in detail. To simplify the analysis, we assume that the code in all DMA architectural enclaves and the user-level UserE is carefully checked and verified to ensure that it works as intended and does not leak additional information to any unrelated parties.

Evidence Freshness and Channel Binding. To guarantee that each remote attestation instance is linked to a particular TLS connection, various measures are implemented. Firstly, unpredictable random nonces from the ServerHello and ClientHello messages are included as the only fresh quote binding to the TLS handshake. These nonces also serve as parameters for obtaining verification reports, anchoring the entire attestation process to the specific TLS channel. Furthermore, the revocation mechanism ensures that previously used quotes cannot be reused, preventing attackers from replaying the previous attestation protocol to deceive the enclave into revealing secrets to an untrusted entity.

Security of Data Transmission. Data transmission is secured by the TLS cryptographic channel in our approach. During the TLS handshake, each enclave authenticates itself using a self-signed certificate, which is subsequently verified by the counterpart to establish mutual trust. Each certificate is linked to a unique key pair generated within the enclave during startup, with the keys regenerated upon each restart. The enclave strictly confines the private key within its boundaries, ensuring it is neither exported nor persisted externally. This measure ensures the confidentiality of communications and makes data transmission between enclaves immune to man-in-the-middle attacks.

Security of Trust Transferring. The EPID group identity materials are generated within enclaves and transmitted only within the AuthE cluster via secure channels. Since all AuthEs have the same functionality, i.e. they have the same code and initial data; they have consistent enclave measurements. This means that no third party can forge the identity of an AuthE and join the cluster, thereby stealing the EPID group identity. When an AuthE fails or suffers an attack and goes down, the consensus algorithm will retrieve the latest state from AuthEs in other network domains and restore it once it is restarted. The fault-tolerant consensus algorithm ensures network-wide consistency of EPID group identities and revoked values as designed. For measures of the availability in a particular network domain, see Sec. 7.2.

Hardware Platform Freshness. Intel provides a mechanism called TCB recovery for SGX hardware devices, which mitigates potential vulnerabilities through CPU microcode version updates. While we assume that the SGX hardware can guarantee integrity and confidentiality during enclave usage, the UserE

has the right to know if the platform of its peer has been updated to the latest version. As AttestE relies on Intel's trust infrastructure to establish trust with AuthE, AttestE caches the hardware TCB trustworthiness assessment received from Intel and includes it in the DMA quote. This allows UserE peers to check the platform's security, thereby maximising the hardware security of the platform running the enclave.

Privacy. The DMA architectural enclaves use the original SGX remote attestation protocol to establish trust. However, Intel's trust infrastructure is not involved in the trust establishment for UserEs, as it relies solely on the DMA architectural enclaves. As OPERA [5] points out, Intel may collect certain information about the DMA architectural enclaves but cannot violate the privacy of the UserE. When establishing trust using DMA architectural enclaves, AttestE only collects information necessary to establish trust and releases it after generating the DMA quote. Additionally, the EPID group signature algorithm guarantees anonymity, ensuring the signer's identity cannot be determined.

6 Implementation and Evaluation

6.1 Implementation

We implement a prototype of DMA[2] with Intel SGX SDK (version 2.21.100.1), Intel EPID SDK (version 8.0.0), and Intel SGX SSL cryptographic library (version support_tls_lin_1.1.1q). While Intel provides open-source implementations for the member and verifier roles, it does not provide implementations for the issuer role. We have extended the issuer role implementation for the latest version of the EPID SDK, building on the foundation provided by OPERA [5]. All functionality has been programmed as much as possible in the enclave, with about 12,000 LOCs in total.

A set of development tools has also been implemented to support UserE development. The `create_quote` function retrieves a DMA quote from the AttestE, while the `verify_quote` function is used to verify the quote provided by the UserEs peers. The `revoke_quote` function sends a DMA quote to the AttestE to initiate a revocation process.

6.2 Evaluation

Setup. We evaluated DMA in both the user terminal and cloud scenarios. In the user terminal setup, various devices were used with an Intel Core i7-7700 or Intel Core i5-8300H CPU, 16GB RAM, 4 physical cores and 8 logical cores with SGX1 support. These devices, running Ubuntu 20.04 on Linux kernel 5.15.0, were linked on the same LAN for communication. For the cloud scenario, multiple VM instances were established across four data centres of a single cloud provider, each with 16GB RAM and 8 Intel Xeon Platinum 8369B virtual cores featuring SGX2 support, running Ubuntu 20.04 on Linux kernel 5.4.0.

[2] `https://github.com/Seix61/DMA`

Latency of basic operations. We first evaluated the latency of some basic operations. These include signing and verification in the EPID scheme, quote generation and verification in the two original remote attestation schemes and DMA, and the insertion and query operations of our signature revocation list.

The signing operation in the EPID scheme is implemented by the `EpidSign` function, while the `EpidVerify` function is used to perform verification. With an empty revocation list, the average latency to create a signature is 63.98 ms, and the average latency to verify is 19.40 ms.

The original remote attestation uses the core function `sgx_get_quote_ex` to generate quote materials based on the incoming attestation key id. For EPID-based attestation, a client is required to obtain a verification report from the IAS, while for DCAP-based, the `sgx_qv_verify_quote` function mainly performs this task from QvE. Table 1 shows the average latency of these operations. As DCAP-based attestation has a smaller computational volume, it is more efficient in both these operations. However, DCAP-based attestation is more suitable for on-premise networks and may raise privacy concerns, as discussed by [5].

Table 1. Latency of quote generation and verification

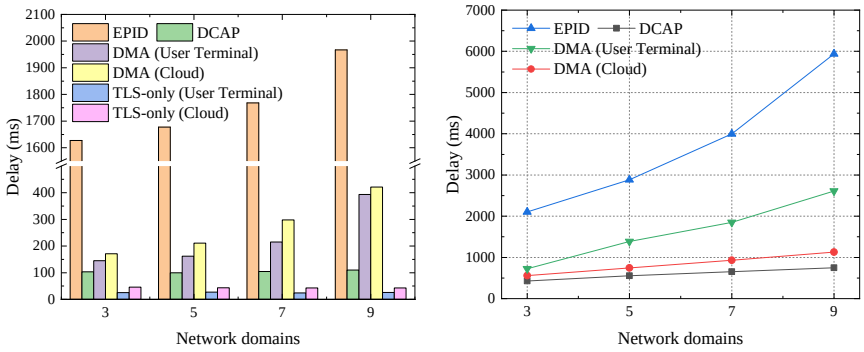
Operation	Latency (ms)		
	EPID	DCAP	DMA
Quote generation	298.50	5.66	85.37
Quote verification	1262.01	34.24	47.61

Since the signature revocation list was implemented using a hash table with constant time complexity for lookup and insert operations, the latency of adding a signature to the revocation list is only 0.005 ms, and the latency of looking up a signature from the revocation list is also 0.005 ms.

Delay of trust establishment. To evaluate the performance of trust establishment, we first evaluated network delay in both scenarios. The average RTT between all nodes in the user terminal scenario was 0.36 ms. In the cloud scenario, the average RTT between all data centres was 26.93 ms. The maximum delay from nodes in each of the four data centres to nodes in the other data centres was 44.70 ms, while the minimum value was 7.89 ms.

Function hooks were inserted to time the TLS handshake, representing the delay in trust establishment. Two main pieces of data were collected: the delay of each individual handshake and the overall delay of all handshakes between a node and all other nodes. To achieve this, we simulated the existence of 3, 5, 7, and 9 network domains by deploying 3, 5, 7, and 9 devices in these two scenarios, respectively. However, IAS can only process some verification requests in a short time if there are more than 11 devices.

Compared to the scenario where only the TLS handshake is performed without establishing enclave trust, DCAP-based attestation results in an additional



(a) Delay of each individual handshake (b) Overall delay of all handshakes

Fig. 5. Delay of trust establishment through handshakes

overhead of 60.47 ms on average, as illustrated in Fig. 5(a). As the number of network domains increases, this additional overhead becomes more evident in the EPID-based scheme. For DMA, the additional delay ranges from 100 ms to 400 ms, while in the original EPID-based attestation, this additional delay can reach up to more than 1900 ms. As previously demonstrated, the operating system needs to allocate more resources for the computationally intensive signing and verification operations for EPID-based schemes. Since the centralised IAS cannot satisfy a large number of verification requests in a short period and the delay of Internet-based quote verification is too long, it dramatically increases the extra overhead of the handshake process.

Fig. 5(b) illustrates the overall delay of trust establishment. Due to the limitations of the devices, the delay of DMA in the cloud scenario is superior to that in the user terminal scenario. Nevertheless, the delay gap between the original EPID-based attestation and DMA is gradually widening due to the decentralised trust management design of DMA. In summary, DMA is approximately 2.5 times more efficient than the original remote attestation scheme.

7 Discussion

7.1 Application scenario

As SGX offers new ideas for protecting sensitive applications by hardware-isolated enclaves, DMA offers a decentralised trust framework for secure interactive computation with enclave participation. DMA boosts trust among participants in secure multi-party interactive or privacy-preserving computations, ensuring sensitive data can be confidently passed to the enclave without concern for data compromise during the computation.

More generally, DMA is a distributed trust solution that focuses on confidential computing autonomous systems. In distributed computing systems, establishing and managing trust is vital to ensuring secure and efficient system

operation. By deploying DMA, confidential computing applications in different network domains can realise mutual attestation and trust management, thus ensuring the security of the distributed confidential computing systems.

7.2 Eliminate single point of failure

In our design, consensus algorithms are used to maintain a unified EPID group across different network domains and to synchronise values to be revoked in all specific network domain accesses to DMA. This allows each AuthE to create internal revocation lists within its respective network domains to transfer trust across different network domains through the AuthEs.

Concerning a particular network domain, consensus algorithms can also mitigate the risk of a single point of failure (SPOF) for the AuthE. To achieve this, multiple instances of AuthE can be instantiated within each network domain, synchronising group identities and revoked values similar to our design. Importantly, these AuthEs have to present a consistent external interface to other network domains, as if they existed only in that particular network domain, by selecting a representative node or load balancing.

7.3 Dependence on blockchain

We are not introducing blockchain by design, although it can be seen as a decentralised ledger to share data securely in a decentralised network environment. However, for confidential computing systems, the introduction of blockchain leads to a larger TCB. As pointed out by [23], once one component of the TCB has a security risk, the entire system's security is compromised.

In contrast, we ensure secure sharing of the system state by introducing the Raft leader election algorithm and the lightweight EPaxos consensus algorithm. This design ensures trust is established depending on the SGX architecture and does not break away from the enclave system through transfer. This is less intrusive on the confidential computing system, resulting in a smaller TCB.

8 Related work

Remote attestation serves as a key mechanism to facilitate trust between enclaves and their relying parties, also when these enclaves need to work together to perform computing tasks. The trust issue becomes more complex when large-scale computing tasks are distributed across multiple enclaves.

To enable on-premise networks such as data centres to establish their trust infrastructure, Intel has introduced a remote attestation scheme based on Intel DCAP [30], reducing the dependence on Intel's online services. OPERA [5] recognised the issues with Intel's centralised attestation service and offloaded the attestation functionality to OPERA servers via custom certificates. However, this merely shifts trust from one centralised entity to another. In contrast, JANUS [33] uses blockchain to create a bulletin board that facilitates the open

participation of relying parties in the attestation verification. As discussed in Sec. 7.3, blockchain integration may augment the enclave system's TCB.

Several attempts have been made to enhance the inherent trust of enclaves. MATEE [12] proposes a multimodal mechanism that improves the redundancy of remote attestation by creating a secondary chain of trust associated with the TPM. JANUS [33] uses the physical unclonable functions (PUFs) to establish an inherent RoT and provides additional measurement mechanisms. However, these methods cannot identify the up-to-date state of the hardware itself. Decent [34] introduces the concept of enclave components and enables distributed applications to mutually authenticate any set of enclave components without third parties. MAGE [4] presents a mutual identity inference mechanism by analysing the measurement mechanism of SGX, allowing enclaves to infer the identities of others only from their initial data. These efforts run in parallel with ours.

Efforts have also been made to integrate the remote attestation with the TLS handshake. Knauth et al. propose RA-TLS [20], which facilitates attestation by embedding pre-generated verification reports during certificate creation. However, as noted by Niemi et al. [27], this approach carries risks of replay and collusion attacks. In contrast to the pre-handshake attestation protocol, TC4SE [15] partitions the attestation into evidence generation and trust exchange, introducing an additional attestation step while preserving the invariance of the TLS protocol. TSL [27] anchors the evidence generation step to TLS handshake instances, providing robust channel binding and freshness similar to our approach but requiring the regeneration of a key pair for each handshake.

9 Conclusion

This paper introduces DMA, a decentralised attestation framework for distributed enclaves. In order to enhance the channel binding and freshness, vital materials are added to bind TLS handshakes to enclave evidence and provide a signature-based revocation mechanism. In order to ensure trust balance across network domains, a decentralised enclave trust transferring mechanism based on consensus algorithms is provided, thus avoiding the use of a centralised attestation service. DMA addresses the issues of efficiency and flexibility in trust establishment and management of distributed enclave systems, facilitating trusted interactions and secure transmissions between multi-party interactive computation participants. A multi-perspective security analysis of DMA is presented, along with a prototype implementation. Experimental results demonstrate that DMA has higher trust management efficiency than the original remote attestation scheme.

Acknowledgments. This work is supported by the National Key R&D Program of China (No.2021YFB3100700), the National Natural Science Foundation of China (No. U22B2029, 62272228), and Shenzhen Science and Technology Program (Grant No.JCYJ20210324134408023).

References

1. Birkholz, H., Thaler, D., Richardson, M., Smith, N., Pan, W.: Remote attestation procedures (rats) architecture. RFC 9334 (2023)
2. Brickell, E., Li, J.: Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities. IEEE Transactions on Dependable and Secure Computing **9**(3), 345–360 (2011)
3. Chen, G., Chen, S., Xiao, Y., Zhang, Y., Lin, Z., Lai, T.H.: Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution. In: 2019 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 142–157. IEEE (2019)
4. Chen, G., Zhang, Y.: {MAGE}: Mutual attestation for a group of enclaves without trusted third parties. In: 31st USENIX Security Symposium (USENIX Security 22). pp. 4095–4110 (2022)
5. Chen, G., Zhang, Y., Lai, T.H.: Opera: Open remote attestation for intel's secure enclaves. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 2317–2331 (2019)
6. Chen, Y., Luo, F., Li, T., Xiang, T., Liu, Z., Li, J.: A training-integrity privacy-preserving federated learning scheme with trusted execution environment. Information Sciences **522**, 69–79 (2020)
7. Containers, I.: Rats architecture based tls using librats. `https://github.com/inclavare-containers/rats-tls` (2021)
8. Dierks, T., Rescorla, E.: The transport layer security (tls) protocol version 1.2. Tech. rep. (2008)
9. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on information theory **29**(2), 198–208 (1983)
10. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the tls 1.3 handshake protocol. Journal of Cryptology **34**(4), 37 (2021)
11. Du, M., Jiang, P., Wang, Q., Chow, S.S., Zhao, L.: Shielding graph for exact analytics with sgx. IEEE Transactions on Dependable and Secure Computing (2023)
12. Galanou, A., Gregor, F., Kapitza, R., Fetzer, C.: Matee: multimodal attestation for trusted execution environments. In: Proceedings of the 23rd ACM/IFIP International Middleware Conference. pp. 121–134 (2022)
13. Goldman, K., Perez, R., Sailer, R.: Linking remote attestation to secure tunnel endpoints. In: Proceedings of the first ACM workshop on Scalable trusted computing. pp. 21–24 (2006)
14. Goldreich, O.: Secure multi-party computation. Manuscript. Preliminary version **78**(110), 1–108 (1998)
15. Hamidy, G.M., Yulianti, S., Philippaerts, P., Joosen, W.: Tc4se: A high-performance trusted channel mechanism for secure enclave-based trusted execution environments. In: International Conference on Information Security. pp. 246–264. Springer (2023)
16. Hanzlik, L., Zhang, Y., Grosse, K., Salem, A., Augustin, M., Backes, M., Fritz, M.: Mlcapsule: Guarded offline deployment of machine learning as a service. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3300–3309 (2021)
17. Huo, T., Meng, X., Wang, W., Hao, C., Zhao, P., Zhai, J., Li, M.: Bluethunder: A 2-level directional predictor based side-channel attack against sgx. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(1), 321–347 (Nov 2019). `https://doi.org/10.13154/tches.v2020.i1.321-347`, `https://tches.iacr.org/index.php/TCHES/article/view/8401`

18. Intel: Attestation services for intel® software guard extensions. https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/attestation-services.html (2023)
19. Johnson, S., Scarlata, V., Rozas, C., Brickell, E., Mckeen, F., et al.: Intel software guard extensions: Epid provisioning and attestation services. White Paper **1**(1-10), 119 (2016)
20. Knauth, T., Steiner, M., Chakrabarti, S., Lei, L., Xing, C., Vij, M.: Integrating remote attestation with transport layer security. arXiv preprint arXiv:1801.05863 (2018)
21. Lamport, L.: Paxos made simple. ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001) pp. 51–58 (2001)
22. Li, X., Li, F., Gao, M.: Flare: A fast, secure, and memory-efficient distributed analytics framework. Proceedings of the VLDB Endowment **16**(6), 1439–1452 (2023)
23. Maene, P., Götzfried, J., De Clercq, R., Müller, T., Freiling, F., Verbauwhede, I.: Hardware-based trusted computing architectures for isolation and attestation. IEEE Transactions on Computers **67**(3), 361–374 (2017)
24. Microsoft: Microsoft azure attestation. https://azure.microsoft.com/en-us/products/azure-attestation (2021)
25. Moghimi, A., Irazoqui, G., Eisenbarth, T.: Cachezoom: How sgx amplifies the power of cache attacks. In: Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. pp. 69–90. Springer (2017)
26. Moraru, I., Andersen, D.G., Kaminsky, M.: There is more consensus in egalitarian parliaments. In: Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. pp. 358–372 (2013)
27. Niemi, A., Pop, V.A.B., Ekberg, J.E.: Trusted sockets layer: A tls 1.3 based trusted channel protocol. In: Nordic Conference on Secure IT Systems. pp. 175–191. Springer (2021)
28. Ongaro, D., Ousterhout, J.: The raft consensus algorithm. Lecture Notes CS **190**, 2022 (2015)
29. Rescorla, E.: The transport layer security (tls) protocol version 1.3. Tech. rep. (2018)
30. Scarlata, V., Johnson, S., Beaney, J., Zmijewski, P.: Supporting third party attestation for intel® sgx with intel® data center attestation primitives. White paper p. 12 (2018)
31. Van Bulck, J., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T.F., Yarom, Y., Strackx, R.: Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient {Out-of-Order} execution. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 991–1008 (2018)
32. Wu, P., Ning, J., Shen, J., Wang, H., Chang, E.C.: Hybrid trust multi-party computation with trusted execution environment. In: NDSS (2022)
33. Zhang, X., Qin, K., Qu, S., Wang, T., Zhang, C., Gu, D.: Teamwork makes tee work: Open and resilient remote attestation on decentralized trust. arXiv preprint arXiv:2402.08908 (2024)
34. Zheng, H., Arden, O.: Secure distributed applications the decent way. In: Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems. pp. 29–42 (2021)