Exploring the use of Intel SGX for Secure Many-Party Applications

Kubilay Ahmet Küçük[1], Andrew Paverd[w], Andrew Martin[1],
N. Asokan[2], Andrew Simpson[1], Robin Ankele[1]
[1]University of Oxford, United Kingdom
{kucuk, andrew.martin, andrew.simpson, robin.ankele}@cs.ox.ac.uk
[2]Aalto University, Finland
andrew.paverd@aalto.fi, asokan@acm.org

Abstract

The theoretical construct of a Trusted Third Party (TTP) has the potential to solve many security and privacy challenges. In particular, a TTP is an ideal way to achieve secure multiparty computation — a privacy-enhancing technique in which mutually distrusting participants jointly compute a function over their private inputs without revealing these inputs. Although there exist cryptographic protocols to achieve this, their performance often limits them to the two-party case, or to a small number of participants. However, many real-world applications involve thousands or tens of thousands of participants. Examples of this type of *many-party* application include privacy-preserving energy metering, location-based services, and mobile network roaming.

Challenging the notion that a *trustworthy* TTP does not exist, recent research has shown how trusted hardware and remote attestation can be used to establish a sufficient level of assurance in a real system such that it can serve as a trustworthy remote entity (TRE). We explore the use of Intel SGX, the most recent and arguably most promising trusted hardware technology, as the basis for a TRE for many-party applications.

Using privacy-preserving energy metering as a case study, we design and implement a prototype TRE using SGX, and compare its performance to a previous system based on the Trusted Platform Module (TPM). Our results show that even without specialized optimizations, SGX provides comparable performance to the optimized TPM system, and therefore has significant potential for large-scale many-party applications.

http://dx.doi.org/10.1145/3007788.3007793

1 Introduction

An ideal Trusted Third Party (TTP) is a theoretical construct used to describe the ideal solution to many types of security and privacy challenges. It is assumed to be an entity that is universally trusted by all participants, without having to provide any evidence of its trustworthiness. A well-known example of a class of problems that could be solved using a TTP is secure multiparty computation (SMPC). As introduced by Yao [23], a multiparty computation problem consists of two or more participants, each holding some individual input, who wish to jointly compute some function over their inputs. However, since the participants are mutually distrusting, they do not want to reveal their private inputs to one another.

The ideal solution to this problem is for each participant to send their individual input to TTP, which can compute the function. Although the use of TTPs is inevitable in some real-world applications (e.g. certificate authorities), conventional wisdom calls for solutions that avoid them, since they are assumed to be difficult to realize. Consequently, researchers have proposed numerous protocols for achieving different types of SMPC based cryptographic primitives such as oblivious transfer [19] and homomorphic encryption [9]. With current techniques, these protocols have significantly worse performance (due to computational and network overheads) than the ideal case using a TTP. In particular, these performance constraints often limit these protocols to the two-party case, or to some small number of participants. However, a large class of real-world applications involve thousands or tens of thousands of participants. We refer to these as *many-party* applications.

Many-party applications. Many-party applications can often be modelled as communication systems, in which *many* mutually distrusting participants [2] wish to communicate with each other without diminishing their privacy. For example, users of a location-based service (LBS) may send their locations to a service to receive location-relevant information, but this diminishes their privacy. Similarly, in the smart energy grid, the fine-grained energy consumption measurements from smart meters could be used to infer personal information about consumers and profile their behaviour. Communicating this information to the service provider is necessary for the smart grid to function, but also diminishes users' privacy [4, 5]. In both cases, privacy-preserving operations could be performed on the communicated information to hide individuals within a group: in the smart grid, measurements could be aggregated over multiple consumers [18, 4], and in LBS, the user's location could instead be reported as an area containing multiple other users (i.e. location cloaking [7]). In both of these cases, these groups could contain many participants (e.g. thousands or tens of thousands). The fundamental question is therefore how to perform these privacy-preserving operations *securely at the required scale.* Current cryptographic approaches do not provide sufficient performance for these many-party applications.

Trustworthy Remote Entities. Challenging the conventional wisdom regarding TTPs, various research efforts have shown how trusted hardware and remote attestation can be used to establish the trustworthiness of a remote system, to the extent that it could serve as a TTP. Paverd et al. [17, 18] proposed the concept of a *trustworthy remote entity (TRE).* The TRE is essentially a TTP that provides a very high level of assurance of its state and behaviour, thus making it *trustworthy* rather than simply *trusted.* In a many-party application, the TRE is situated as an intermediary in the communication paths between participants and performs privacy-enhancing operations on the communicated information. For example, in the energy metering scenario described above, the smart meters communicate directly with the TRE, which performs privacy-preserving operations (e.g. aggregation [15, 20]) on the sensitive energy measurements. Any participant who individually trusts the TRE can take part in the communication without loss of privacy. The principal requirement is therefore to provide all participants with sufficient evidence of the TRE's state and behaviour for them to make informed trust decisions. This is achieved using remote attestation. In their implementation of the TRE, Paverd et al. [17, 18] used Intel Trusted Execution Technology (TXT) and the Trusted Platform Module (TPM) as the basis for this remote attestation.

Using Intel SGX.

In this paper, we explore the use of Intel Software Guard Extensions (SGX) to implement a TRE for many-party applications. In particular, we consider attestation performance in terms of the time required to perform a single attestation operation and the overall rate at which a platform can perform attestations. We designed and implemented the equivalent of Paverd's TRE using SGX for the smart grid use case. Due to the fundamental differences in architecture, this was far from a straightforward porting task, and required a complete redesign of the system. Thanks to the SGX architecture, our implementation requires significantly fewer lines of code, which both reduces the burden on the developer, decreases the likelihood of code defects, and minimizes the amount of code that must be trusted by the verifier. Although our implementation targets a specific application domain, we argue that its core features are common to all many-party applications, and could thus serve as an architectural template for such applications. Using the smart grid as a case study, we performed a comparative evaluation of the performance of our SGX implementation (*SGX-TRE*) against the previous TPM-based implementation (*TPM-TRE*). Our results show that even an unoptimized SGX implementation exhibits comparable performance to the optimized TPM-TRE, whilst providing stronger security guarantees. Therefore, our principal contributions are:

- Empirical performance measurements of basic SGX operations that are used in many-party applications (Section 4).

- An architectural design and prototype implementation of a representative real-world application, which could serve as a template for other such many-party applications (Section 5).
- A systematic comparison of an SGX-TRE against a TPM-TRE, in the context of many-party applications (Section 6).

2 Background

2.1 Remote Attestation

A longstanding challenge in distributed computing is that communicating entities have little or no guarantee of what software is running at the remote computer. This may be problematic for many reasons, but in particular it means that one party has no assurance that the other will implement a security policy as expected. Remote attestation attempts to solve this problem by providing a precise account of the state of the remote system — and hence of the software which has been loaded and executed on that platform. Typically, this is achieved by providing some type of *attestation evidence* (e.g. a cryptographic hash) of the binary software that has been loaded on the remote platform. The authenticity of this evidence must be guaranteed by some *root of trust* on the platform, which is usually provided by trusted hardware.

In general, the system being attested is called the *prover*, whilst the system evaluating the attestation is called the *verifier*. In all cases, the verifier is required to decide whether or not the attested system is trustworthy. This becomes exponentially more difficult as the size of the attested component (e.g. the number of lines of code) increases. In order to use attestation successfully, it is therefore critical to minimize the size of the attested component.

2.2 Intel SGX

Intel SGX is a set of CPU extensions that allow applications to create isolated execution environments, called *enclaves*, that protect the confidentiality and integrity of their contents from all other software on the platform, including a potentially untrusted OS [16]. Since the platform's main memory may be under the adversary's control, SGX ensures that an enclave's data is encrypted and integrity protected before it leaves the boundary of the CPU. Enclaves also provide secure storage by allowing data to be *sealed* such that it can only be decrypted by a specific enclave [1]. SGX supports both local and remote attestation: enclaves can be attested by other enclaves on the same platform, or by remote verifiers.

The SIGMA protocol [14] is used to establish secure channels and bind these to the enclave's attestation. In contrast to discrete trusted hardware (e.g. TPMs as a discrete IC), all SGX computation is performed by the main CPU, resulting in significant performance improvements. The typical SGX use case is to isolate and protect security-sensitive components of an application. Minimizing the size of these *trusted* components reduces the risk of security vulnerabilities, and reduces the burden on the verifier during remote attestation.

3 System Model & Requirements

3.1 System Model

Fundamentally, a many-party application consists of a set of participants $\mathcal{P} = \{p_1, p_2, ... p_n\}$ each of whom holds some sensitive information s_p. The overall objective is to compute some function over all participants' private inputs $f(s_{p1},\ s_{p2},\ ...\ s_{pn})$. In the most challenging case, these participants may be mutually distrusting of one another, and thus do not wish to reveal their sensitive information to one another.

All participants use remote attestation to ascertain the current state and behaviour of a common TRE. The exact mechanisms used by the participants to verify the attestation are beyond the scope of this work, but could include security audits or formal analysis of the attested software, either undertaken by the participants themselves or by an entity they trust. Having received this strong assurance that the TRE will not divulge or misuse their information, participants send their information directly to the TRE. The TRE can then compute the required function securely and efficiently.

3.2 Adversary Model

In the above system model, the aim of the adversary $\mathcal{A}$ is to learn the secret information s_p of any of the participants. We assume $\mathcal{A}$ has full control over the TRE's platform. This means that $\mathcal{A}$ is capable of loading and executing arbitrary software, modifying the OS and other system software, and/or modifying the pre-OS components (e.g. bootloader) of this platform. Since $\mathcal{A}$ also has physical access, he may add or remove hardware components, reset the platform, and exploit any software or hardware side-channel attacks. We assume that $\mathcal{A}$ has full control of all communication channels, and thus also has all the capabilities of the classical Dolev-Yao network adversary. In other words, we intend the TRE to be trustworthy even if the TRE's operator is malicious. In many-party protocols, the correctness of the TRE's output naturally depends on the correctness of the inputs provided by the participants (or if the TRE has mechanisms for detecting invalid inputs). Although this is an important consideration, it mainly depends on the specific applications and protocols, and is thus beyond the scope of this work.

3.3 Security Requirements

Although many-party applications may have various security requirements, the following are applicable to the TRE itself:

- **Secure Computation:** No external entity should be able to observe or interfere with the computation performed by the TRE. Since the adversary has physical access to the platform, the confidentiality and integrity must be protected from untrusted software on the same platform.

- **Secure Communication:** Communication between the TRE and other parties must be confidential and integrity-protected.

- **Strong Attestation:** The root of trust used for attestation must be trusted by all parties. The attestation must be unambiguously linked to the communicating entity (authenticity) to avoid masquerading attacks, and must convey the current state of the system (freshness) to prevent replay attacks.

Since the TRE's role is to perform privacy-preserving operations, various privacy requirements apply. However, these have been discussed previously [2], and are beyond the scope of this paper, since they apply to the specific operations performed by the TRE.

3.4 Performance Requirements

As an intermediary in the communication between participants, the following performance metrics are applicable to the TRE:

- **Scalability:** The overall rate at which the TRE performs privacy-preserving operations should be maximized.

- **Latency:** The time taken for each individual operation should be minimized.

In some cases, the application domain places hard constraints on these performance parameters. For example, in the smart energy grid, each smart meter produces a measurement every 30 minutes [18], so the overall rate at which the TRE can receive and process these measurements determines the number of smart meters that can be supported by a single TRE (i.e. the TRE's *scalability*).

Table 1: Measurements of basic SGX operations (average and variance over 100 runs)

Operation	Stack+Heap	Mean (ms)	Std Dev (ms)
Create Enclave	20 kB	9.986	0.488
	5 MB	24.558	2.154
Initialize Remote Attestation	20 kB	0.040	0.004
	5 MB	0.055	0.012
Initialize Secure Channel	20 kB	0.511	0.056
	5 MB	0.611	0.083
Quote & SIGMA Protocol	20 kB	33.059	1.968
	5 MB	31.764	1.250
Destroy Enclave	20 kB	0.116	0.060
	5 MB	1.158	0.103

4 Benchmarking SGX

As SGX appears to be the most suitable of the current trusted hardware technologies, we performed a series of micro-benchmarks to establish the baseline performance of the basic SGX operations required for the TRE.

The results of these benchmarks are shown in Table 1. For each operation we varied the combined enclave stack and heap size, since our preliminary experiments indicated that it is this overall size that has an impact on performance. All benchmarks were performed on an SGX-enabled Dell Latitude E5570 laptop, with an Intel Skylake Core i7-6600U 2.60 GHz CPU, running Ubuntu 14.04 with the 2016-06 public SGX SDK. The platform's Enclave Page Cache (EPC) was set to 128 MB.

The first measurement shows the cost of creating an enclave using either 20 kB or 5 MB of memory. This operation would only be performed very infrequently, and will not have a significant impact on the TRE's performance. As expected, enclave creation time increases as the size of the enclave's memory increases, due to the fact that the enclave's pages are *measured* during initialization. The second, third, and fourth measurements show the cost of various steps in establishing a secure connection between the TRE and a relying party, and attesting the TRE. These must be performed at least once per relying party, but, depending on the application protocol, may be performed more frequently (e.g. in cases where it is infeasible for the TRE to maintain concurrent connections with all relying parties, or where relying parties may be periodically reset). The second measurement shows the cost of initializing a remote attestation session with a single relying party. The third measurement shows the cost of generating the first message of the Diffie-Hellmann key exchange. The fourth measurement shows the longest operation, which includes both quoting and key exchange operations for the SIGMA protocol.

For completeness, the final measurement shows the cost of destroying an enclave, although this is not a frequent operation.

5 Implementing the TRE on SGX

In this section we describe our design and prototype implementation of the TRE using Intel SGX. Although our system is designed for a particular application, the majority of the design would be common to many other application domains, and could therefore serve as a template for using Intel SGX to implement TREs or similar types of trustworthy systems for many-party applications.

5.1 Smart Grid Use Case

The smart grid scenario consists of a service provider (SP) e.g. the energy supplier, and multiple smart metering devices (SMDs). Each SMD performs frequent measurements of energy consumption (e.g. every 30 minutes) and sends these to the SP. These fine-grained consumption measurements allow the SP to monitor the energy distribution network and enable *time-of-use* (ToU) billing. However, it has been shown that such detailed energy consumption traces can be used to infer private information about the consumer, leading to privacy concerns [4, 5]. In this context, each SMD is a participant $p \in \mathcal{P}$, and the energy consumption measurements are the participant's private input s_p. The objective of the TRE is to protect the privacy of individual consumers whilst still enabling the overall functionality of the smart grid.

Figure 1 shows the core components of the TRE and interactions between the TRE and other participants in the smart grid. In this scenario, the SP and SMDs are assumed to be mutually distrusting of one another. Following the design of Paverd et al. [18], the TRE is therefore placed as an intermediary in the communication path between these entities. The arrows in the figure show the sequence of communication between the SP and SMDs. The SP send requests to the SMDs via the TRE, and the SMDs respond via the TRE.

The TRE can thus perform various types of privacy-preserving operations on the SMDs' consumption measurements, including spatial and temporal aggregation as explained below.

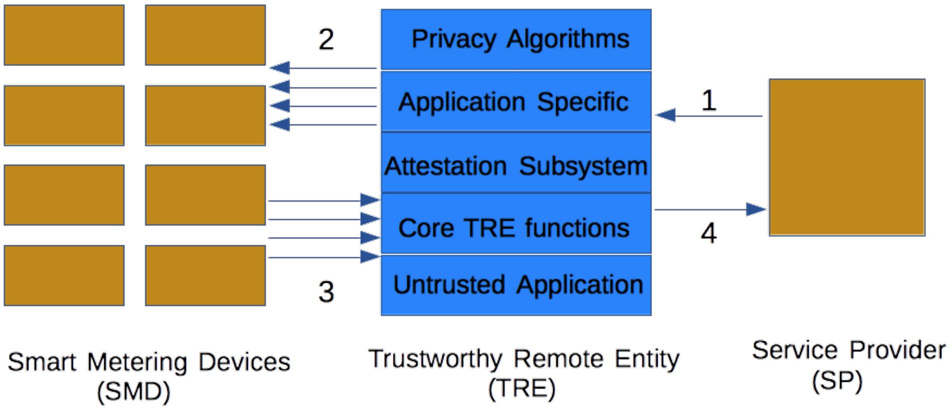

Figure 1: System overview of a TRE in the smart grid

5.1.1 Spatial Aggregation

To monitor overall consumption in an area, it is sufficient for the SP to receive the sum (S) of the private consumption values from all SMDs in that area. This protects privacy by hiding each individual's contribution within the total. Although S could be computed using cryptographic techniques, the performance overhead would be infeasibly high due to the large number of participants. In contrast, the TRE can compute S very efficiently, and can scale linearly as the number of participants increases. However, the SP must trust the TRE to perform the aggregation correctly, and the SMDs must trust the TRE not to disclose their individual measurements. Although not included in our prototype, this spatial aggregation could also make use of differential privacy [8].

5.1.2 Temporal aggregation

In ToU billing, each participant's bill is computed by multiplying the participant's consumption values for each time period by the prevailing cost per unit of energy for that time period. In this case, spatial aggregation cannot be used because each consumer must receive an individual bill. Instead the TRE performs temporal aggregation, by keeping a running total for each consumer within the TRE, and forwarding these totals to the SP at the end of the month. This facilitates ToU billing whilst protecting privacy because SP is unable to infer fine-grained consumption details from the total bill, but again requires the SP and SMDs to trust the TRE.

5.2 TRE Operation

As shown above, the TRE can be used to enhance privacy in the smart grid if it is trusted by the SP and the SMDs. The SP and SMDs are therefore the *relying parties*. By implementing the TRE using trusted hardware (e.g. Intel SGX), the relying parties can use remote attestation to establish the trustworthiness of the TRE.

Before any relying party communicates with the TRE, the relying party runs a remote attestation protocol with the TRE to ascertain the precise software running in the TRE enclave. In theory it is only necessary for each relying party to attest the TRE enclave once. However, this results in long-lived connections, which may be infeasible if the TRE has to maintain too many connections or if the relying parties are reset during the connection. In our implementation, we take the most conservative approach and assume that relying parties perform a new attestation and establish secure channel for each message sent to the TRE (e.g. once per 30 minutes). In reality, the performance of the TRE would improve if longer-lived connections are possible.

However, remote attestation is only useful if the relying parties can decide whether to trust the software in the TRE enclave. This means that the design of the TRE should aim to facilitate remote attestation and minimize the effort required to verify this attestation. Concretely, this calls for minimizing the software Trusted Computing Base (TCB) of the TRE, since this minimizes the likelihood of software defects, and makes the software more amenable to security audits or even formal analysis. The TRE's software must also be available to all relying parties, and must support reproducible builds in order to verify that the source matches the attestation. Intel SGX is therefore an ideal trusted hardware technology for the TRE due to its hardware-enforced trusted execution environment and remote attestation capabilities.

5.3 TRE Components

This subsection describes the specific components of our prototype implementation of the TRE, as shown in Figure1.

5.3.1 Untrusted Components

The TRE consists of an SGX enclave running on a (potentially untrusted) host platform. The host is responsible for establishing network connections and marshalling encrypted data between the relying parties and the trusted components of the TRE. The untrusted components therefore include the platform's hardware drivers, network communication library, and SGX interface driver. None of these components handle sensitive information in the clear.

5.3.2 Remote Attestation & Secure Channels

One of the main trusted components of the TRE is the subsystem responsible for establishing secure channels and attesting the TRE to relying parties. Our implementation follows the recommended SGX design patterns for remote attestation and secure channel establishment. Specifically, the TRE enclave performs a location attestation to the quoting enclave (QE), which verifies that the TRE enclave is a legitimate SGX enclave running on the same platform, and produces a quote. Using the functionality provided by the SGX SDK, the TRE enclave uses the SIGMA protocol to establish a secure channel with a relying party and bind this channel to the quote. This component therefore consists of the trusted cryptography and key exchange libraries. Compared to Paverd's TPM-based implementation [17, 18], this component was significantly easier to implement due to the inclusion of this functionality in the SGX SDK.

5.3.3 Application-Specific Components

Every TRE includes trusted components that are specific to the application domain in which the TRE is used. In this use case, the application-specific components consists of the application logic for parsing the messages from the SP and SMDs. In our implementation, the TRE communicates with SMDs using the IEC 62056 (*DLMS-COSEM*) message specification, which has been mandated as the communication format for all SMDs in the UK. DLMS-COSEM supports a request-response model, in which messages are encoded as binary data, with typical message lengths in the range of 20 to 30 bytes. In addition to the DLMS-COSEM parser, the TRE also includes the specific privacy-preserving algorithms used in this scenario (e.g. spatial and temporal aggregation).

5.3.4 Privacy-Enhancing Algorithms

In addition to the application-specific functionality, the TRE can include a library of standard privacy-enhancing algorithms, which can be re-used as necessary for different applications. For example, the functionality required to enforce differential privacy [8] could be included in this component. Although not used in our prototype, differential privacy guarantees could be applied to both spatial and temporal aggregation algorithms.

5.3.5 TRE Core

The role of the TRE core is to bring together the above functional building blocks, and to provide the trusted side of the interface between the TRE enclave and the host platform. This component therefore includes the SGX trusted interface libraries, as well as the trusted runtime libraries, such as

Table 2: TCB Sizes of different TREs

	TPM-TRE	**SGX-TRE**
Crypto Libraries	14,408	2,529
Communication	5,969	858
Memory Management	1,035	774
C/C++ Library	854	7,528
Core TRE	720	229
Application Specific	507	507
Attestation	221	364
Drivers	1,005	-
SGX Trusted	-	2,968
Total	24,719	15,757

the C/C++ standard library, and the trusted memory management subsystem. Again, compared to the TPM-TRE, the majority of these libraries are already available through the SGX SDK.

6 Evaluation

We measured the TCB size of our SGX-TRE and benchmarked its performance in a simulated smart grid scenario.

6.1 Software TCB Size

Table 2 shows the sizes of the software TCBs in the TPM-TRE [17, 18], compared to our SGX-TRE. All code was formatted using the same coding conventions and counted using SLOCCount.[1] In the SGX-TRE, the TCB measurements include both the third-party code, and the trusted library code (which was counted from the published library sources), since both are part of the TCB. This is a conservative worst-case TCB size, since it could be argued that the trusted libraries are common to all enclaves, and so will be audited/verified to a greater extent than third-party code.

The most notable differences are the cryptographic library and communication subsystems. The TPM-TRE requires a full TCP/IP stack, network interface driver, and TLS library In contrast, the SGX-TRE can outsource network connectivity to the untrusted host and only requires a small set of interface functions and cryptographic primitives provided by the `tCrypto` and `tKeyExchange` trusted libraries (i.e. to support the SIGMA protocol and send and receive encrypted messages). Furthermore, the SGX `tcrypto`

[1] `http://www.dwheeler.com/sloccount/`

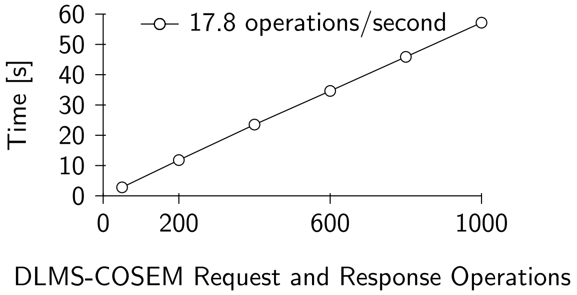

Figure 2: Overall Performance of TRE on SGX

library makes use of *AES-NI.* The TPM-TRE also includes the TPM hardware driver and library, but, in SGX, the drivers used to set-up and call the enclave are part of the untrusted host platform.

On the other hand, the SGX-TRE includes a significantly larger set of C/C++ standard library functions, since the TPM-TRE contains only a small customized subset of standard C library functions. The SGX-TRE also includes other SGX trusted libraries, such as `tservice` (incl. `selib`, `tseal` and `tae_service`) and `trts`, which are not present in the TPM-TRE.

The relative TCB sizes definitively show that even with our conservative posture of counting the SGX trusted libraries as part of the TCB, the SGX-TRE has a smaller TCB and is thus easier to attest than the TPM-TRE. The absolute size of the TCB is within the same order of magnitude as fully-verified systems such as seL4 [12].

6.2 End-to-End Performance Evaluation

To measure performance, we deployed our prototype SGX-TRE on the same hardware used in our initial benchmarks. We simulated the SP and SMDs using a second platform connected via a local ethernet network including a single network switch. Interaction with the Intel Attestation Server (IAS) is necessary to verify an attestation; however, since this is performed by the relying party, it does not affect the scalability of the TRE, and is thus excluded from our performance measurements.

The overall objective of this experiment was to determine how many clients a single TRE can support in a many party application. This is an important metric because it determines the total number of TREs required for the application (e.g. to handle communication with all smart meters in a country such as UK). Our test suite measured the time taken to perform the following sequence of operations for a single relying party: 1) create the TRE enclave, 2) establish a secure channel to an SMD over the network, 3) perform remote attestation, 4) encrypt, send, decrypt, and process messages, and 5) destroy the enclave. In Figure 2, the average time required for this sequence was 56.05 ms with a standard deviation of 2.653 ms (measured over

100 samples). Repeating this test for a batch of participants confirmed that this time scales linearly in the number of relying parties, as expected. In practice, the TRE enclave would not be created and destroyed between every interaction. This would reduce the time per interaction by approximately 10 ms (based on Table 1), bringing the average latency to approximately 46 ms. In the smart grid scenario, the TRE must perform two interactions with each SMD in every 30 minute period [18], and thus a single SGX-TRE can support approximately 20,000 relying parties. This is the same number of participants as the TPM-TRE, which enjoys the performance benefits of various optimizations, including a highly scalable attestation protocol [17]. We plan to investigate similar optimizations for the SGX-TRE.

In this case study, the processing performed by the TRE is not computationally intensive. As a result of this, a significant percentage of the time per interaction is spent establishing the secure channel and performing remote attestation. Although this may not always be the case, we argue that there is a large class of TRE applications for which this will apply, and therefore that it is beneficial to optimize these processes.

6.3 Security Evaluation

SGX does not protect against side-channel attacks and does not provide oblivious memory. If the control flow or data memory access pattern of the enclave depend on secret data, this secret information could potentially be leaked to the adversary through side-channel attacks [22] (since the adversary may control the host platform). However, by design neither the control flow nor the memory access pattern of our SGX-TRE depend on any secret data. The number of participants communicating with the TRE is not secret information, since this can be inferred from analysis of network traffic. Furthermore, our TRE implementation does not perform any OCALLs, and thus is not vulnerable to Iago attacks [6]. Since the TRE is event-driven [17], the SGX enclave is invoked in response to commands or messages originating from the relying parties. The enclave uses the return values of the ECALLs to issue commands to the host platform (e.g. to initiate new connections and send messages). While the TPM-TRE does not protect against hardware attacks on the platform's memory, the memory-encryption functionality of SGX ensures that the SGX-TRE's secrets are protected even against an adversary with the capability to read the platform's memory.

7 Related Work

In addition to Paverd's TRE [17], similar approaches using trusted hardware have been suggested. Koeberl et al. [13] propose a conceptual approach in which a generic compute provider, supported by a TEE, provides trust brokerage between different data providers. In particular, the TEE can apply privacy-preserving operations on the intermediate data. Similarly, Kim et al. [11] implemented network applications with OpenSGX [10], demonstrating the use of TEEs as a *middlebox*. Their work includes several case studies related to secure code execution. Moat [21], a tool to formally verify the confidentiality properties of SGX applications, could be used to verify properties of the TRE.

Atamli-Reineh and Martin [3] explored different schemes for partitioning a trusted application into multiple TEEs, and showed that the choice of partitioning scheme has an impact on security and performance.

8 Conclusion and Future Work

Overall, our results show that Intel SGX can be used to build a trustworthy remote entity (TRE) for use in many-party applications. Compared to the previous TPM-based design, our SGX-TRE has two main advantages. First, it requires significantly fewer lines of code, which reduces the burden on the developer, decreases the likelihood of code defects, and minimizes the amount of code that must be trusted by a relying party. Secondly, SGX memory protection enables the TRE to resist a stronger adversary, who has physical access to the platform's memory. Our initial SGX-based implementation provides the same performance (and thus scalability) as the highly optimized TPM-TRE. However, we identified that improving the performance the SGX remote attestation protocol could benefit the SGX-TRE.

As future work, we plan to optimize the design and implementation of our SGX-TRE and explore the use of multiple enclaves and different enclave structures for the TRE.

9 Acknowledgment

We acknowledge with gratitude a research grant from Intel Corporation, which supports some of the authors. We thank the reviewers for their helpful comments and we thank Çağlayan Aras for help with measurements.

References

[1] I. Anati et al. Innovative Technology for CPU based Attestation and Sealing. In *HASP@ ISCA*, 2013.

[2] R. Ankele et al. Applying the Trustworthy Remote Entity to Privacy-Preserving Multiparty Computation: Requirements and Criteria for Large-Scale Applications. In *IEEE International Conference on ATC*, 2016.

[3] A. Atamli-Reineh and A. Martin. Securing Application with Software Partitioning: A Case Study Using SGX. In *Security and Privacy in Communication Networks*. 2015.

[4] A. Bartoli et al. Secure Lossless Aggregation for Smart Grid M2M Networks. In *IEEE SmartGridComm*, 2010.

[5] J.-M. Bohli, C. Sorge, and O. Ugus. A Privacy Model for Smart Metering. In *IEEE International Conference on Communications Workshops*, 2010.

[6] S. Checkoway and H. Shacham. Iago Attacks: Why the System Call API is a Bad Untrusted RPC Interface. In *Eighteenth International Conference on ASPLOS*, 2013.

[7] C.-Y. Chow et al. A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-Based Service. In *ACM symposium on Advances in Geographic Information Systems*, 2006.

[8] C. Dwork. Differential Privacy. In *Automata, Languages and Programming*. 2006.

[9] C. Gentry et al. Fully Homomorphic Encryption using Ideal Lattices. In *STOC*, volume 9, 2009.

[10] P. Jain et al. OpenSGX: An Open Platform for SGX Research. In *NDSS*, 2016.

[11] S. Kim et al. A First Step Towards Leveraging Commodity Trusted Execution Environments for Network Applications. In *ACM Workshop on Hot Topics in Networks*, 2015.

[12] G. Klein et al. seL4: Formal Verification of an OS Kernel. In *ACM SIGOPS 22nd SOSP*, 2009.

[13] P. Koeberl et al. Time to Rethink: Trust Brokerage Using Trusted Execution Environments. In *TRUST*. 2015.

[14] H. Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its use in the IKE Protocols. In *CRYPTO*. 2003.

[15] K. Kursawe et al. Privacy-Friendly Aggregation for the Smart-Grid. In *PETS*, 2011.

[16] F. McKeen et al. Innovative Instructions and Software Model for Isolated Execution. In *HASP@ ISCA*, 2013.

[17] A. Paverd. Enhancing Communication Privacy Using Trustworhy Remote Entities. DPhil Thesis, University of Oxford, 2016.

[18] A. Paverd, A. Martin, and I. Brown. Privacy-Enhanced Bi-Directional Communication in the Smart Grid using Trusted Computing. In *IEEE SmartGridComm*, 2014.

[19] M. O. Rabin. How To Exchange Secrets with Oblivious Transfer. *IACR Cryptology ePrint Archive*, 2005, 2005.

[20] A. Rial and G. Danezis. Privacy-preserving smart metering. In *ACM workshop on Privacy in the Electronic Society*, 2011.

[21] R. Sinha et al. Moat: Verifying Confidentiality of Enclave Programs. In *22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.

[22] Y. Xu, W. Cui, and M. Peinado. Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems. In *IEEE Symposium on Security and Privacy*, 2015.

[23] A. C. Yao. Protocols for Secure Computations. In *Foundations of Computer Science*, 1982.