Oblivious Digital Tokens

Mihael Liskij
ETH Zurich

Xuhua Ding
Singapore Management University

Gene Tsudik
UC Irvine

David Basin
ETH Zurich

Abstract

A computing device typically identifies itself by exhibiting unique measurable behavior or by proving its knowledge of a secret. In both cases, the identifying device must reveal information to a verifier. Considerable research has focused on protecting identifying entities (provers) and reducing the amount of leaked data. However, little has been done to conceal the fact that the verification occurred.

We show how this problem naturally arises in the context of *digital emblems*, which were recently proposed by the International Committee of the Red Cross to protect digital resources during cyber-conflicts. To address this new and important open problem, we define a new primitive, called an Oblivious Digital Token (ODT) that can be verified obliviously. Verifiers can use this procedure to check whether a device has an ODT without revealing to any other parties (including the device itself) that this check occurred. We demonstrate the feasibility of ODTs and present a concrete construction that provably meets the ODT security requirements, even if the prover device's software is fully compromised. We also implement a prototype of the proposed construction and evaluate its performance, thereby confirming its practicality.

1 Introduction

We begin with the problem that motivates this research. Given the rapid growth of cyber armed forces in numerous states and the increasing likelihood of cyber-warfare, the International Committee of the Red Cross (ICRC) introduced a digital red cross emblem. According to ICRC, this emblem should [26]:

> "(...) convey a simple message: in times of armed conflict, those who wear them, or facilities and objects marked with them, must be protected against harm."

Similarly to prominently displayed red cross emblems physically painted on (or attached to) protected buildings, facilities, or vehicles in war zones, the digital emblem is intended to flag digital assets, e.g., computing devices, that are off-limits to cyber-attacks under international humanitarian law. A possible use-case [26] described by the ICRC is as follows: *During an armed conflict with State A, State B's malware spreads automatically to affect computers managing A's military supplies. State B's reconnaissance indicates that some systems are marked by digital emblems and belong to a hospital. State B therefore amends its attack program to avoid harming such systems.* Among the various technical and operational issues listed by the ICRC, we believe that any digital emblem scheme must satisfy three important properties for the parties involved in a cyber-conflict (e.g, States A and B) to respect it[1]:

- **Verification Obliviousness** Emblem verification must not put the aggressor (State B) at risk of being exposed.
- **Binding Integrity** Emblems should not be misused by the administrator (State A) "to falsely claim protection, e.g., by routing operations through facilities showing the emblem." [26]
- **Security Preservation** The security of protected assets should not be weakened by emblems, i.e., State B should be unable to leverage information returned by emblems to facilitate or ease its attacks on State A, even if it decides to do this contrary to the law.

Since the original call to action on digital emblems by the ICRC, there is increasing interest in this topic, including within the Internet Engineering Task Force (IETF) [33]. Thus far, two realizations of a digital emblem [26, 32] have been proposed and more may be on the way. Both proposals assume that the administrator (State A) is trusted. One technique embeds information in visible system artifacts [26] (e.g., file names, IP addresses, or domain names) where the names or associated metadata indicate that the entity is protected. The other technique is the Authenticated Digital Emblem (ADEM) scheme [32], which builds a PKI-like infrastructure for verifying digital emblems inserted into DNS, TLS, and other communication.

In practice, we believe that the security of either technique

[1]The fact that some parties may not respect international humanitarian law neither lessens nor obviates the ICRC's need for digital emblems.

is uncertain. A dishonest administrator can easily break verification obliviousness by monitoring accesses to a file emblem. It can also compromise binding integrity by cloning a file emblem to unauthorized devices or by using a device protected by a digital emblem as a network proxy for its own (not protected) devices' communications. These administrator attacks are possible due to the inherent tension between binding integrity and (either or both) verification obliviousness and security preservation. We elaborate on this in the next section.

In this paper, we tackle these limitations and propose a novel digital emblem scheme that satisfies three properties listed above without trusting the system software and the network infrastructure on the administrator's side. The proposed digital emblem is a dynamically generated proof that a device is protected under international humanitarian law, in contrast with prior schemes where the emblem is static. To highlight this distinction and the different trust model, we call our emblems *Oblivious Digital Tokens* (ODTs).

At a high level, we devise a way for a device to insert an ODT into every outgoing TLS handshake. ODT generation takes place inside a pre-installed Trusted Execution Environment (TEE) instantiated using Intel SGX or Arm TrustZone (currently the most popular commodity TEEs), combining system security and cryptographic techniques. Disguised as a regular TLS server, an aggressor expects TLS connections from compromised devices and verifies the ODTs (if any) received over these connections. ODT verification is made oblivious by ensuring that messages sent by the aggressor are indistinguishable from a standard TLS message flow.

This work makes the following contributions.

- We propose the notion of ODT, which formulates the ICRC's digital emblem concept by considering both sides of a cyber-conflict as adversaries with respect to ODT security.
- We present a concrete ODT scheme constructed using two components: a TEE-based system to generate evidence for binding integrity and a Privacy-Preserving Equality Test (PPET) protocol for evidence verification.
- We prototype the ODT scheme based on OpenSSL and SGX and evaluate its practicability and performance. Our results show that ODT generation costs about 144ms per TLS handshake. For the security evaluation, we use the Tamarin protocol verifier to prove binding integrity and conduct a statistical and system analysis for security preservation and verification obliviousness, respectively.

Generally speaking, the ODT scheme can be viewed as a special type of a remote attestation technique. However, to the best of our knowledge, it is the first-of-its-kind scheme that considers both the prover and the verifier to be malicious and is thus designed from the stance of a neutral third-party.

ORGANIZATION. Section 2 formulates the ODT problem as an extension of digital emblems. Section 3 overviews our ODT construction, components of which are described in Sections 4 and 5. The complete scheme is presented in Section 6. Our security analysis is given in Section 7, followed by discussion and related work in Section 8.

2 Problem Formulation

We now review the ICRC digital emblem notion and formulate it as an ODT under an adversary model that is more realistic and stronger than those of prior schemes [26, 32].

2.1 ICRC Emblems

Figure 1 depicts the digital emblem setting considered by the ICRC. There are three stakeholders:

(1) The neutral third party ($\mathcal{NTP}$) represents an organization, such as the ICRC or Doctors Without Borders, that is protected under international humanitarian law. $\mathcal{NTP}$ issues and installs emblems on its devices: desktops, servers, laptops, and smartphones. These devices are said to be $\mathcal{NTP}$*-protected* to reflect their special legal status. Analogous to physical emblems, digital emblems just signal this status and *do not prevent* attacks. Device D in Figure 1 illustrates one such example.

(2) Once deployed, i.e., placed in the field, D is managed by an entity on one side of the cyber-conflict, called the administrator ($\mathcal{A}dm$). $\mathcal{A}dm$ also manages other non-protected devices without emblems, e.g., D' in Figure 1.

(3) The offensive party in the conflict is the aggressor ($\mathcal{A}ggr$), who mounts cyber-attacks on $\mathcal{A}dm$'s devices. According to international humanitarian law, $\mathcal{A}ggr$ is expected to check for the presence of an $\mathcal{NTP}$-issued emblem on the targeted or compromised device, before causing any damage. Hereafter, we use the terms $\mathcal{A}ggr$ and *Verifier* interchangeably.

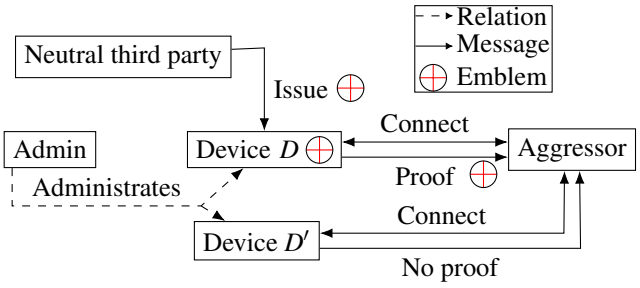

Figure 1: Overview of ICRC digital emblem use-case. D is $\mathcal{NTP}$-protected while D' is not.

2.2 Incentives and Models

To simplify the description, we only consider two devices, D and D', as in Figure 1, throughout the rest of the paper. The proposed technique is easily generalized to settings with additional devices.

$\mathcal{A}dm$ and $\mathcal{A}ggr$ are hostile to each other and both attempt to exploit the digital emblem scheme to their maximum advantage, in different ways.

- $\mathcal{A}dm$ has incentives to break binding integrity. It attempts to deceive $\mathcal{A}ggr$ into believing that D' is $\mathcal{NTP}$-protected.
- $\mathcal{A}dm$ also has incentives to break verification obliviousness. It attempts to detect the existence of emblem verification activities in D and D', e.g., to determine whether a file read or a network connection is being used for emblem verification.
- $\mathcal{A}ggr$ has incentives to verify emblems in order to comply with international humanitarian law. It also aims to break security preservation by abusing emblem verification to gather information about the target devices' software and configuration, which is otherwise unavailable.

Attack Capabilities We assume that $\mathcal{A}ggr$ can stealthily hack into both D and D', and inject code using techniques independent of the digital emblem. $\mathcal{A}dm$ fully controls the network and its infrastructure. This is a more powerful model than the classic Dolev-Yao adversary [19] that only controls communications. $\mathcal{A}dm$ also controls all system software, i.e., operating systems and hypervisors (if any) in both D and D', and has access to all system resources, e.g., memory and I/O ports. $\mathcal{A}dm$ monitors all relevant events, such as system calls, network, and disk I/O, as well as scans all memory for specific patterns.

Trust Model We assume that D is equipped with a hardware-based publicly identifiable TEE, e.g., Intel SGX or Arm TrustZone. We assume that no adversary against binding integrity can break the security assurances of the TEE, including data confidentiality and control flow integrity. However, we do *not* assume that executions inside the TEE reveal no side-channel information to the adversary that attacks verification obliviousness, since such assurance is outside the TEE's own design scope. Thus, the TEE is not part of the Trusted Computing Base (TCB) of verification obliviousness, which is realized using cryptography.

We do not consider $\mathcal{A}ggr$'s false claims of emblem absence on $\mathcal{NTP}$-protected devices. Likewise, emblem-unrelated attacks launched from $\mathcal{NTP}$-protected devices, e.g., a DDoS attack against $\mathcal{A}ggr$, are out of scope of this work.

CAVEAT. It is well-known that secret keys maintained within TEEs are subject to various side-channel attacks [45, 12, 41]. The compromise of such keys directly breaks binding integrity since the attacker can then clone the emblem. However, it does not affect verification obliviousness in our model and is orthogonal to security preservation.

2.3 Oblivious Digital Token (ODT)

The ODT definition embodies the ICRC's notion of digital emblems in the setting of Figure 1 with security in the adversarial model described above.

Definition 1. *Given any $\mathcal{NTP}$-protected device D and any unprotected device D', an ODT is an unforgeable and verifiable digital emblem created by an emblem generation functionality, such that, if verified successfully, the ODT proves to $\mathcal{A}ggr$ that D is $\mathcal{NTP}$-protected and satisfies three properties: (a) it is infeasible to produce an emblem proving that D' is $\mathcal{NTP}$-protected (**binding integrity**); (b) no entity (including $\mathcal{A}dm$ and $\mathcal{NTP}$) can detect the existence of $\mathcal{A}ggr$'s verification (**verification obliviousness**); and (c) no information about D is leaked to $\mathcal{A}ggr$ (**security preservation**).* □

Formal definitions of the properties (a)–(c) are deferred to Section 7, which also contains our security analysis. Note that an invalid ODT does not mean that the device is not $\mathcal{NTP}$-protected, because this could be the result of tampering with ODT generation or transmission. Also note that security preservation requires that an adversary cannot deduce non-public information about D, for example, library versions, running processes or secret keys, by engaging in ODT verification. Security preservation is not concerned with the anonymity of the device, since protected devices belong to public institutions.

2.4 To Invade or Not To Invade

$\mathcal{A}ggr$ may wish to verify the ODT of a device just via network communication because, by being non-invasive, it can easily remain oblivious to $\mathcal{A}dm$. However, our threat model reflects that $\mathcal{A}dm$ is capable of system manipulation and we argue that this prevents a non-intrusive $\mathcal{A}ggr$ from verifying emblems with binding integrity. To see why this is the case, we describe attacks on ADEM and some hypothetical schemes before delving deeper into the problem.

2.4.1 Attacks on ADEM and Its Variants

Consider a DNS-based ADEM as a warm-up example. $\mathcal{A}dm$ can simply assign D' the IP address ostensibly for D. As a result, $\mathcal{A}ggr$ would mistakenly believe that D' is $\mathcal{NTP}$-protected. Similar attacks work against other types of ADEM emblems.

One might try to strengthen ADEM with a hardware-based TEE. For example, $\mathcal{NTP}$ could install a TEE on D to host a device-specific signing key, with an associated certificate. For every TLS connection, the TEE then returns to its peer a dynamically generated emblem, i.e., an attestation using the current TLS handshake secret and the device state. This scheme would prevent $\mathcal{A}dm$ from cloning the emblem from D to D', since the TEE's key is bound to D's hardware. However, $\mathcal{A}dm$ can route D''s network traffic through D, similar to a NAT setting. This attack, which is explicitly noted by ICRC, breaks binding integrity since the remote party would mistakenly treat its communication peer D' as $\mathcal{NTP}$-protected.

Similarly, isolating the entire TLS layer or even the hardware interface using the TEE only makes $\mathcal{A}dm$'s routing attack more complex, rather than effectively stopping the attack.

2.4.2 The Core Issue

Under our adversary model, $\mathcal{A}dm$ can manipulate the system and network infrastructure so that *no reliable device namespace* is accessible to $\mathcal{A}ggr$. Keys, IP addresses, host names, etc. are all logical identifiers. By themselves, they neither enable $\mathcal{A}ggr$ to securely pinpoint a physical device nor determine the emblem's protection scope.

This issue is somewhat similar to cuckoo attacks [42, 55], a well-known problem in identifying the root of trust (RoT) in a remote device. Current solutions to cuckoo attacks, such as those using distance-bounding [17] and ambient properties [55], fail in the ODT context for two reasons: First, they require $\mathcal{A}ggr$ to have physical control over the relevant device, e.g., by being in close physical proximity. Second, they require the prover's explicit and interactive cooperation with the verifier. Both are impractical in a cyber-conflict. They also conflict with the verification obliviousness requirement.

2.4.3 Necessity of Compromise-then-Verify

Recall the use case in ICRC's report [26] that suggests a compromise-then-verify paradigm in which $\mathcal{A}ggr$ invades a target device and injects its *agent*, an executable program for reconnaissance purpose, to check the existence of a file emblem or a process emblem among an "active process list". This paradigm is consistent with the so-called "kill-chain model" used in the Advanced Persistent Threat (APT) attacks as reported by Hutchins et al. [25], as well as ransomware, and supply chain attacks. These sophisticated attacks, with several US government organizations (including the DoD) on their victim list, establish a persistent and stealthy command-and-control channel between a compromised device and a remote server on the Internet.

We thus propose for $\mathcal{A}ggr$ to build the namespace for its targets by leveraging its capability of invading them. The intuition is that the reconnaissance agent helps $\mathcal{A}ggr$ to identify the target. Namely, $\mathcal{A}ggr$ checks whether the program's execution and ODT generation take place on the same device.

Stealthy Compromise v.s. ODT Properties Clearly, $\mathcal{A}ggr$'s stealthy compromise is a prerequisite for its ODT verification to achieve binding integrity. If the agent is detected, $\mathcal{A}dm$ can run it on D' and enlarge its execution scope to enclose D, thus breaking binding integrity. The relation between stealthy compromise and verification obliviousness is more intricate and subtle. It is neither in our scope nor is it $\mathcal{NTP}$'s responsibility to *ensure* that the program remains undetected. Nonetheless, it is in our scope to ensure that *ODT verification does not induce additional risks of exposing $\mathcal{A}ggr$'s reconnaissance agent*.

3 ODT Scheme: Overview

The proposed scheme follows the compromise-then-verify paradigm. We assume that $\mathcal{A}ggr$ can stealthily inject its reconnaissance agent into target devices. In the following, we first describe how our scheme expects the agent to behave and then describe the scheme's workflow, followed by the technical approach.

3.1 Agent Description

The agent either runs as an independent process or resides within a victim process. The ODT scheme only expects the agent to use its hosting device's default library to establish TLS connections with a remote server, presumably under $\mathcal{A}ggr$'s control. We chose TLS over other protocols since it is very widely used by applications in laptops and mobile phones, thus providing a better coverage than less popular protocols. We neither impose requirements on the bytes sent or received in the connections nor do we require the agent to perform any specific operation or have specific memory contents, such as a key or a secret pattern.

Since the proposed scheme only assumes that the agent makes TLS connections in the same way as other applications and malware do that, we show in Section 7.4 that the scheme does not make the agent more susceptible to detection than TLS-using malware.

3.2 High-Level Workflow

Figure 2 illustrates a system consisting of D' and D managed by $\mathcal{A}dm$. Processes in both devices make TLS connections to various servers. $\mathcal{NTP}$'s TEE (denoted *O-TEE*, for *oblivious TEE*) is installed on D. O-TEE holds a globally unique public key certified by $\mathcal{NTP}$ and inserts itself into *all* TLS handshakes initiated by local processes in D. $\mathcal{A}ggr$'s agent stealthily runs on D and D'. As O-TEE and $\mathcal{A}ggr$ operate without coordination, we sketch their operations separately.

Treating each incoming TLS server handshake as a challenge, O-TEE produces the corresponding *co-residence witness*, which serves as evidence that O-TEE is (or is not) on the same platform as the client process. After the handshake, O-TEE returns to the TLS server the ODT which consists of the witness and a signature. We stress that O-TEE behaves the same way in every TLS connection and cannot determine if its peer is $\mathcal{A}ggr$'s TLS server.

Upon receiving a TLS client handshake, $\mathcal{A}ggr$'s TLS server (i.e., the second server in Figure 2) sends an ODT verification challenge disguised as the server handshake, such that the resulting flow is indistinguishable from those sent by regular servers. After the handshake, it extracts the ODT (if any) from received messages. It then verifies the ODT offline by checking the signature and whether the witness matches the expected state of its agent.

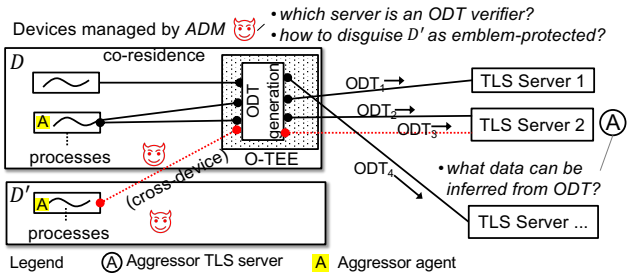

Figure 2: A system view of the ODT scheme. O-TEE mediates four TLS handshake sessions and produces distinct ODTs. Red lines indicate $\mathcal{A}dm$'s attack on binding integrity.

3.3 Our Approach

Our security rationale is as follows. We design a *Witness Generation System* to attain binding integrity, and a Privacy-Preserving Equality Test protocol (PPET) for witness verification. The latter prevents an ODT verifier from inferring information from the witness, thus contributing to security preservation. Moreover, the witness generation system and the PPET protocol are integrated with TLS messages, such that neither O-TEE nor $\mathcal{A}dm$ can detect whether an ODT verification is taking place, which ensures verification obliviousness. We now outline the three components of the ODT scheme whose details are given in subsequent sections using the notations in Table 1.

1. Witness Generation System: A *witness* is a hash digest that securely captures both the memory state of the TLS client thread and the thread's co-residence (or lack thereof) with O-TEE. Essentially, O-TEE runs this component to perform a memory measurement that attests to binding integrity without trusting the operating system. (See Section 4.)

2. Privacy-Preserving Equality Test (PPET) Protocol: The second component is an interactive cryptographic protocol. It has three message flows between Prover (i.e., O-TEE) holding a secret w and Verifier (i.e., $\mathcal{A}ggr$) that guesses Prover's secret as w'. At the end of the protocol, Verifier determines whether $w = w'$ is true, without being able to derive any other information about w. The distinguishing feature of this PPET protocol, compared with similar schemes [34, 36], is its *TLS-compatibility*, rather than stronger security or better performance. This feature means that: (i) protocol messages are compatible with the TLS standard; and (ii) Prover can successfully run the protocol without differentiating whether TLS handshake messages come from a normal TLS server or Verifier. This compatibility paves the way for the third component that makes $\mathcal{A}ggr$ indistinguishable from a normal TLS server, which yields verification obliviousness. (See Section 5.)

3. TLS Integration The third component integrates the Prover part of the PPET protocol with the TLS client handshake protocol run by O-TEE, and integrates the Verifier part of the PPET protocol with the TLS server handshake protocol run by $\mathcal{A}ggr$. This integration allows O-TEE to perform PPET as Prover in *every* TLS handshake, oblivious to whether its peer is an actual Verifier. Hence $\mathcal{A}ggr$ hides its PPET execution under the guise of a TLS server. (See Section 6.)

Table 1: Notation used throughout the paper.

Notation	Description
pk_D, sk_D	public/private key pair for O-TEE in D
σ_D	signature generated using sk_D
$\mathbb{B}^n$	set of all n-bit binary strings.
Ω	predefined set of address ranges, $\Omega \subset \mathbb{B}^{48}$
HS	TLS handshake secret i.e., the ephemeral Diffie-Hellman key
$\iota^{-1}(), \iota()$	Elligator [8] encoding and decoding functions.

4 Witness Generation

Recall that $\mathcal{A}dm$ is the adversary (working against binding integrity) that controls all system software in D and D'. Hence, the main problem for witness generation is to detect whether the data acquired by O-TEE is *not* staged by $\mathcal{A}dm$.

4.1 O-TEE Initialization

When D boots up, its hardware launches O-TEE provisioned by $\mathcal{NTP}$. O-TEE generates an asymmetric key-pair (pk_D, sk_D), the public key component of which (pk_D) is certified by the underlying hardware. $\mathcal{NTP}$ defines a set of 48-bit virtual address[2] segments to be used as the domain of memory measurement, denoted by $\Omega \subset \mathbb{B}^{48}$.

4.2 Memory Measurement Scheme

The *co-residence witness* is the outcome of the O-TEE-based memory measurement scheme. We define this scheme below and describe how a valid co-residence witness can only be produced for a process running on the same device. The measurement is executed by O-TEE after deriving the TLS handshake secret k. Hence, we define the measurement scheme with k as one of the inputs.

Suppose that process P in D is to be measured. Based on k, O-TEE selects one or more virtual memory locations of P using the function Select, and produces a witness w by running the procedure Measure.

- Select is a function that takes k as input and returns a vector of m challenge addresses $(c_1, \cdots, c_m)$ where: $c_i = \mathsf{M}(\mathcal{H}(i||k))$ (for $1 < i \leq m$). $\mathcal{H}$ is a cryptographic hash function, and M is a random function that uniformly maps a hash digest into Ω.

[2]Both x86-64 and ARM64 architectures use such addresses.

- Measure is a procedure that reads m memory words $(w_1, \cdots, w_m)$ from P's virtual memory at challenge addresses $(c_1, \cdots, c_m)$. It also sets a bit-flag B_{CO} to zero to indicate its co-residence with P. It returns a 256-bit w as the *witness*: $w = \mathcal{H}(B_{CO} \| w_1 \| w_2 \cdots \| w_m)$. The methods for determining B_{CO} depend on the O-TEE architecture. Two concrete ones are described in Section 4.3.

The measurement must be *secret*, *atomic* and *performed at native speed*. *Secret* means that the memory address to be read is unknown to $\mathcal{A}dm$ before the read operation, which prevents $\mathcal{A}dm$ from gaining any advantage by preparing the targeted data ahead of time. *Atomic* means that, if the memory read is interrupted, the interruption is faithfully factored into the measurement result. *Performed at native speed* means that it is not slower than the kernel.

The rationale for using a secret, atomic and native-speed memory measurement scheme is as follows. Consider a process $\hat{P}$ running on D'. $\mathcal{A}dm$ wants to deceive O-TEE about its co-residence with $\hat{P}$. Under our adversary model, $\mathcal{A}dm$ cannot tamper with O-TEE's execution. It also does not know any challenge addresses before the measurement starts. Hence, it either (i) replicates $\hat{P}$'s entire virtual memory to D, or (ii) tampers with O-TEE's measurement to learn the challenge addresses before making a targeted copy.

The former (i) is a brute-force attack that incurs a prohibitively high cost since the size of an application's virtual memory ranges from a few to hundreds of megabytes. It requires copying the memory words as well as building the same address mapping, which involves page table modifications. The cost is even higher considering that $\mathcal{A}dm$ does not know which process on D' is $\mathcal{A}ggr$'s agent. It must clone every process from D' to D in order to pass ODT verification with a 100% probability. This is equivalent to running them directly on D.

For attack (ii) to succeed, $\mathcal{A}dm$ must: (1) win the race with O-TEE (i.e., complete the forgery before O-TEE finishes the read), and (2) hide any trace of tampering. O-TEE always starts the measurement before $\mathcal{A}dm$ can attack, and O-TEE reads the memory as fast as $\mathcal{A}dm$. On one hand, without interrupting O-TEE, $\mathcal{A}dm$ cannot win because it must first observe and then copy. On the other hand, an interrupt can be detected by O-TEE. Moreover, since all memory locations are independently chosen $c_i = \mathsf{M}(\mathcal{H}(i||k))$, similar to shuffled measurements [11] used in device attestation, exposure of the locations measured thus far reveals nothing about subsequent ones.

In summary, if the O-TEE-based memory measurement scheme is secret, atomic and performed at native speed, it yields a valid co-residence witness with respect to the measured process P and O-TEE. The concrete schemes described in Section 4.3 satisfies the three requirements above. O-TEE uses the secret key k to determine the memory location(s) to measure. It reads the target memory at the native speed and in an atomic way. The hardware updates the O-TEE state if an interrupt or exception occurs when executing the memory read instruction.

4.3 Concrete Schemes

On laptops, desktops, and servers, we propose to instantiate O-TEE using the well-known Intel SGX[3]. On mobile and IoT devices, we instantiate O-TEE using Arm TrustZone on Cortex-M or Cortex-A processors. Confidential computing technologies, such as Intel TDX [27], Arm CCA, and AMD SEV, are not ideal for O-TEEs because it is difficult (if not impossible) for software running in those TEEs to atomically access virtual memory of external processes.

4.3.1 SGX-based O-TEE

Intel SGX isolates user space data and code demarcated by a virtual address range with an *enclave* comprising a set of Enclave Page Cache (EPC) pages. SGX provides a confidential computing environment: no software outside an enclave (e.g., kernel or BIOS) can access an enclave's EPC pages. Also, an SGX enclave is publicly verifiable.

O-TEE can take the form of an enclave mapped to all processes' virtual address spaces as part of the system library. Intel's Attestation Service (IAS) provides a hardware-based remote attestation facility that allows a remote verifier to authenticate an enclave's genuineness based on the hash digest of memory pages loaded during enclave creation. O-TEE encloses the hash of its public key pk_D into its local attestation to the Quoting Enclave, an enclave provisioned by Intel to facilitate remote attestation. The latter generates and signs the enclave quote for O-TEE using the platform's attestation key. The signed quote is released for public verification. Hence, $\mathcal{A}ggr$ can determine that there *exists* a legitimate O-TEE instance that owns pk_D.

Atomic Memory Access SGX does not provide a secure clock for the enclave code to measure time. When executing enclave code, the CPU accesses non-EPC pages the same way as when running normal code, i.e., via the MMU's address translation and the memory controller. A memory load instruction is either executed unobstructed or it encounters an exception. No software can intervene in its operation without triggering an exception. Any exception during enclave execution causes an immediate enclave exit. Before the CPU is trapped to the kernel, an `EXITINFO` object located in the enclave State Save Area (SSA) page is updated by hardware to reflect the exception cause. Since the SSA page is inside the enclave, no system software can read from, or write to, it. Our design uses this SGX feature to detect exceptions during memory measurement.

[3]While SGX is deprecated on future "client platform" processors, it continues to be available on Xeon processors for servers and cloud platforms.

Measurement and Witness Computation O-TEE enclave is pre-configured to enable the hardware to log page faults and general protection exceptions. (Other types of exceptions are logged by default.) It clears `EXITINFO` in the SSA page just before the measurements. It then sets B_{CO} to 0 after the measurement if `EXITINFO` indicates an exception and sets B_{CO} to 1 otherwise.

4.3.2 TrustZone-based O-TEE

Arm TrustZone partitions the platform into: *Normal World* that hosts the Rich OS plus regular applications and *Secure World* that hosts the Trusted OS and secure applications. O-TEE is realized as a kernel module of the Trusted OS. Unlike Intel SGX, there is no centralized trust hierarchy established on a processor and software in Secure World. Nonetheless, the Trusted OS can carry a credential certified by $\mathcal{NTP}$ or the manufacturer. The Trusted OS on D can further certify pk_D generated by O-TEE. Thus, the certificate of pk_D is publicly verifiable.

Atomic Memory Access As part of the Trusted OS, O-TEE runs at Exception Level 1 (EL1) of Secure World. Its code and data are located in the kernel virtual address range and are thus translated via the `TTBR1_EL1` register. When execution of a user process P is trapped to Secure World for requesting TLS connections, O-TEE clones the `TTBR0_EL1` register serving P. This way, O-TEE memory loading instructions can directly read P's virtual memory.

All exceptions raised in Secure World are delivered to the Trusted OS by the hardware. To detect an exception during measurement, O-TEE hooks the exception handler of the Trusted OS before the measurement starts. It sets B_{CO} to 0 if its handler is called; and to 1 otherwise.

5 Privacy-Preserving Equality Test (PPET)

Recall that PPET is a building block for the ODT scheme. It allows $\mathcal{A}ggr$ to check whether the witness w generated by O-TEE matches the value w' expected by $\mathcal{A}ggr$. We now describe cryptographic aspects of the protocol, showing how the privacy of w is protected against $\mathcal{A}ggr$ when $w \neq w'$ and how it is structurally compatible with TLS.

While prior private equality test (PET) [34, 36] and private set intersection (PSI) [50, 16, 24] protocols provide the privacy properties we need, they are *not* TLS-compatible. For instance, some protocols require the verifier to send two messages [34] or a zero-knowledge proof showing that its message is well-formed [16]. Since large messages and complex cryptographic data structures do not fit into TLS messages, $\mathcal{A}ggr$ cannot send them covertly. This leads us to construct a TLS-compatible PPET protocol.

5.1 The Protocol

Let p be a large prime number and $\mathbb{G}$ be a group of order p where the DDH assumption [9] holds. Let g be a generator in $\mathbb{G}$. The PPET protocol *PPET* consists of four steps run by Prover and Verifier as in Figure 3.

`PPET protocol between Prover holding` w `and Verifier holding` w'

1. (by Prover) Send a random group element $u \in_R \mathbb{G}$.
2. (by Verifier) Do the following:
   - Pick a random number $s \in_r \mathbb{Z}_p$ and compute $v = g^s u^{w'}$ as the commitment to its w'.
   - Send v.
3. (by Prover) If v is not in $\mathbb{G}$, send two random group elements $y, z \in_R \mathbb{G}$; otherwise do the following:
   - Pick a random number $t \in_r \mathbb{Z}_p$ and compute $y = g^t$ and $z = v^t u^{-wt}$.
   - Send (y, z).
4. (by Verifier) Assert $w = w'$ iff $z = y^s$.

Figure 3: PPET Protocol *PPET*.

If and only if $w' = w$, $z = g^{st} u^{(w'-w)t} = g^{st} = y^s$. Hence, if both parties execute the protocol faithfully, the protocol returns the correct outcome with respect to w and w'. The rationale behind *PPET* is that Verifier commits to its witness w' as part of sending v. Prover's response is computed from v such that a mismatching w' is not canceled out of the u^t term, which masks information about w. As a result, Verifier must run the protocol with Prover to verify its guessed value for w. If Verifier intends to learn w, it must perform *online* guessing attacks. By keeping w secret, we minimize information leakage about the memory locations used to derive it.

5.2 Privacy Preservation

If, after running *PPET*, Verifier learns that $w' = w$, information is leaked. However, if $w' \neq w$, Verifier learns nothing about w except that it differs from w'. Thus, a malicious Verifier can eliminate at most one value after each protocol run and cannot perform an offline guessing attack. We denote the resulting security property as *privacy preservation*.

Definition 2 (Privacy Preservation)**.** *Let $\mathbb{G}$ be a group of order p and g a group generator. Let $\mathbb{W}$ be the domain of possible witnesses. For all witness verification protocols PPET and all adversaries $\mathcal{A}dv$, we define the advantage function:*

$$\mathsf{adv}^{\mathsf{PP}}_{\mathcal{A}dv,PPET} = \Big|\frac{1}{2} - \mathsf{Pr}\big[\mathbb{G} \xrightarrow{\$} u; \mathbb{Z}_p \xrightarrow{\$} t; \mathbb{W} \xrightarrow{\$} w; \mathcal{A}dv(u) \xrightarrow{\$} v \in \mathbb{G}; z_0 = v^t u^{-wt}; \mathbb{G} \xrightarrow{\$} z_1; \{0,1\} \xrightarrow{\$} b; \mathcal{A}dv(g^t, z_b) \xrightarrow{\$} b' : b' == b\big]\Big|.$$

A PPET satisfies privacy preservation if, for all efficient algorithms $\mathcal{A}dv$, $\mathsf{adv}^{\mathsf{PP}}_{\mathcal{A}dv,PPET}$ is negligible when $\mathcal{A}dv$ incorrectly guesses the witness.

Theorem 1. *PPET satisfies privacy preservation.* [4]

5.3 Structure Compatibility with TLS

The message flows of the *PPET* protocol and a TLS handshake have a similar composition structure. (The distribution indistinguishability between corresponding data objects is addressed in the next section.) Figure 4 illustrates the client-server interaction in a TLS v1.3 handshake [46]. The first two messages perform an ephemeral Diffie-Hellman key exchange. After receiving ClientHello, the server computes the handshake secret $\mathsf{HS} = X^{y_s}$ before sending ServerHello.

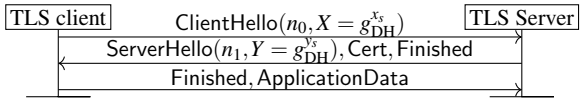

Figure 4: Message flows in the TLS v1.3 handshake.

Verifier Flow vs. TLS Server Flow *PPET* requires Verifier to send one random group element v, while in TLS a server sends a nonce. The congruence between the two flows allows us to make them indistinguishable using additional encoding techniques (see Section 6), which are needed for oblivious verification. Note that it is insecure to use TLS application data to send v since it is *not* a universal behavior for all TLS servers.

Prover Flows vs. TLS Client Flows The first flow in *PPET* from Prover is congruent to its counterpart in TLS ClientHello. Both include random numbers. However, TLS client's `FINISHED` message does not involve any random numbers. Hence, when *PPET* is integrated with TLS, y and z are sent in a TLS heartbeat message [46].

6 Complete ODT Scheme Over TLS

We now present the ODT scheme that integrates: (1) witness generation and Prover steps of *PPET* into TLS client functionality inside O-TEE on $\mathcal{NTP}$-protected devices, and (2) Verifier steps of *PPET* into $\mathcal{A}ggr$ acting as a TLS server. This integration ensures that the $\mathcal{A}ggr$'s message is indistinguishable from a ServerHello message. Hence, O-TEE runs consistently with standard TLS servers and $\mathcal{A}ggr$'s server.

6.1 TLS for ODT Communication

While the *PPET* and TLS flows are structurally similar, we must fine-tune the data representation in order to achieve indistinguishability.

[4] For the proof, see the appendix.

6.1.1 TLS Placeholders

For ODT messages from O-TEE, we use the field ClientHello.random (i.e., client nonce n_0 in Figure 4) to send u and the TLS heartbeat request for sending y and z. We chose to send the message v from $\mathcal{A}ggr$ in the field ServerHello.random, i.e., server nonce n_1 in Figure 4. This chosen field is mandatory in the TLS handshake phase. Hence, its appearance in the flow does not violate obliviousness.

Both nonces are 256 bits long and, more importantly, both are randomly sampled values, so no subsequent TLS computations restrict their domain. By default, a TLS heartbeat request carries a payload of up to 2,048 bytes and requires a heartbeat response echoing it back to the sender. The table below summarizes the TLS objects involved in the scheme.

TLS obj	Type in TLS	ODT obj	Sent by
nonce n_0	random 256 bits	u	O-TEE
heartbeat	2048 bytes	y, z	O-TEE
nonce n_1	random 256 bits	v	Server /$\mathcal{A}ggr$

6.1.2 *PPET* Data in TLS Objects

To load *PPET* data into selected TLS objects, we must address both the size issue (to fit the limited binary size) and the binary representation issue (to match the distribution of the TLS objects). Since both nonces are 256 bits, we instantiate group $\mathbb{G}$ in PPET as an elliptic curve whose group elements can be represented by 256-bit strings. As a result, both u and v fit in their respective nonces. Since the heartbeat request can accommodate up to 2,048 bytes, it can easily hold y and z.

Group Elements vs. Random Numbers As described in Section 5, $\mathcal{A}ggr$'s commitment v is uniformly distributed in $\mathbb{G}$. Since $\mathbb{G}$ is instantiated as a subgroup of points on an elliptic curve, v must be converted into a 256-bit string. Hence the set of the binary strings representing all group elements is only a subset of the domain $\mathbb{B}^{256}$, which is used for nonce generation. Since the server nonce is not always convertible to an element in $\mathbb{G}$, directly using v as a nonce in ServerHello indicates that there is a non-negligible chance that the server is $\mathcal{A}ggr$.

To address this issue, we use Elligator [8] – a censorship-circumvention technique that transforms elliptic curve points into random-looking strings and vice versa – so that an adversary cannot differentiate between an elliptic curve point and a random string with non-negligible probability. We use Curve25519 [7] for $\mathbb{G}$ since it is supported by Elligator. To represent v with a 256-bit nonce, we first apply the Elligator decoding function which returns a 254-bit binary string, and then prepend the string with two random bits to get the nonce. To parse a 256-bit nonce to v, we discard the two most significant bits and apply the Elligator encoding function to the remaining bits.

CAVEAT. Note that $\mathbb{G}$ is used by both O-TEE and $\mathcal{A}ggr$ in the *PPET* protocol. It is independent of the group used in the Diffie-Hellman handshake in TLS. Moreover, we do not require O-TEE's message u in the *PPET* protocol to be encoded using Elligator, since O-TEE always sends out a random group element regardless of whether its peer is $\mathcal{A}ggr$ or not. There is therefore no risk to verification obliviousness.

6.2 A Complete Account of the ODT Scheme

Figure 5 shows the computational steps of the ODT scheme. We first describe how O-TEE runs as a TLS client and emits an ODT in every TLS handshake. Afterwards, we describe how $\mathcal{A}ggr$ runs as a TLS server and verifies an ODT.

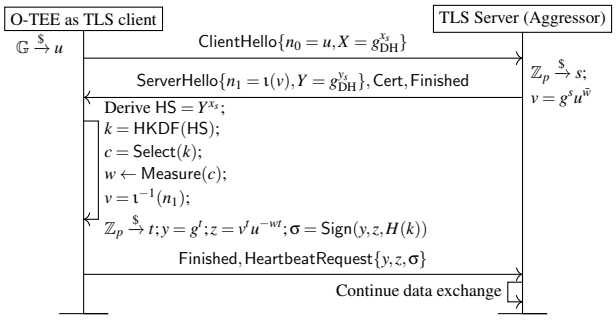

Figure 5: Process P's TLS handshake through O-TEE.

6.2.1 O-TEE as TLS client

Triggered by process P's request to establish a TLS connection, O-TEE executes the following steps.

STEP 1. It initiates a TLS handshake session by sending ClientHello. It sets nonce n_0 as the binary representation of a random group element $u \in_R \mathbb{G}$. This is the first message in *PPET*.

STEP 2. After receiving ServerHello with nonce n_1, it computes the handshake secret HS following the TLS specification and runs the witness generation procedure in Section 4 against P's virtual memory. It also treats n_1 as the Verifier message in *PPET* and decodes n_1 into $v \in \mathbb{G}$. It sets

$$k = \mathsf{HKDF}(\mathsf{HS}); c = \mathsf{Select}(k); w = \mathsf{Measure}(c); v = \iota^{-1}(n_1);$$

where HKDF is the key derivation function used in TLS and ι^{-1} is the Elligator encoding function used to extract an elliptic curve point from a binary string.

STEP 3. It uses witness w to compute (y, z) in the third flow of *PPET*. It signs (y, z) and $\mathcal{H}(k)$ with sk_D, and embeds both (y, z) and the resulting signature σ into a heartbeat request message immediately after the handshake's end.

STEP 4. It passes all session keys to an external TLS library and is not involved in the rest of the TLS connection in order to achieve better performance. □

Note that the triple (y, z, σ) returned in Step 3 is the ODT in our scheme. Issuing a heartbeat request is not a normal TLS client's behavior. However, it does not compromise obliviousness since O-TEE does this for all TLS connections, except for TLS servers that explicitly disable heartbeat messages. We discuss alternatives to heartbeat messages in Section 8.2.

Since the hash of HS is covered by σ, a successful signature verification by the TLS server implies that O-TEE, which owns pk_D, is also the TLS client holding k. This assures the TLS server that O-TEE *is* the communication endpoint with which it interacts and that the generated ODT is cryptographically linked to this TLS connection. Revealing all TLS session keys *after* signing $(y, z, \mathcal{H}(k))$ does not break the ODT-TLS link. Although it can intercept the heartbeat request, $\mathcal{A}dm$ cannot replace the ODT because it cannot force O-TEE to generate a TLS shared key equal to k. Note that, to verify if D *is* the device it intends to check and also contains O-TEE owning pk_D, $\mathcal{A}ggr$ must validate (y, z) using the verifier algorithm in *PPET*.

6.2.2 $\mathcal{A}ggr$ as a TLS Server

Note that $\mathcal{NTP}$ publishes all parameters used in witness generation and *PPET*, such as group $\mathbb{G}$ and generator g so that $\mathcal{A}ggr$ has all the needed parameters before starting any verification.

Suppose that $\mathcal{A}ggr$ sets up its own TLS server and waits for its agents' connection requests. There are several ways for $\mathcal{A}ggr$ to obtain the expected value of the witness. For instance, if $\mathcal{A}ggr$ trusts its command-and-control channel with the agent, it can use that channel to secretly extract needed memory data. If the agent's memory state is less influenced by external factors, $\mathcal{A}ggr$ can execute the agent on the system environment similar to the one in D. The agent runs until it requests a TLS connection to $\mathcal{A}ggr$'s TLS server for ODT verification. The resulting memory state mirrors the agent's state running on the remote devices. Note that address space layout randomization [43, 53] used in commodity operating systems only randomizes base addresses of code, stack, heap and libraries. Also note that these are only suggestions: our scheme does not prescribe a specific way for $\mathcal{A}ggr$ to obtain the witness value.

Upon receiving a TLS connection request, $\mathcal{A}ggr$ proceeds as follows:

STEP 1. Following the TLS standard, it generates its Diffie-Hellman key share, computes HS, and sets

$$k = \mathsf{HKDF}(\mathsf{HS}); c = \mathsf{Select}(k).$$

It obtains expected witness w' and follows the *PPET* protocol to compute the commitment v using the client nonce $n_0 = u$ if it is in $\mathbb{G}$. Otherwise, v is computed with a randomly chosen u. It then uses the Elligator decoding function ι to convert v into the 256-bit server nonce n_1 in its ServerHello. After this step, it behaves exactly according to the TLS standard.

STEP 2. If a heartbeat request message is received from the present TLS session, $\mathcal{A}ggr$ responds to the heartbeat request following the TLS standard and saves the ODT in the message for an offline verification in two steps described below.

(a) $\mathcal{A}ggr$ parses the binary string in the heartbeat request into (y,z) and signature σ. It then verifies σ against (y,z) and $\mathcal{H}(k)$ with the credential certified by $\mathcal{NTP}$.

(b) Following the *PPET* protocol, $\mathcal{A}ggr$ checks if $z \stackrel{?}{=} y^s$ returns True. If so, it asserts that the device where its agent resides is $\mathcal{NTP}$-protected. □

If $z \neq y^s$, $\mathcal{A}ggr$ cannot assert that the device is not $\mathcal{NTP}$-protected. There are various possible reasons for that outcome. For instance, the ODT scheme's execution is attacked by an adversary or the client process is not $\mathcal{A}ggr$'s agent. Note that the agent can repeat the verification by making a new TLS connection.

6.3 Experiments

We built a prototype of the proposed ODT scheme to assess its performance and practicality. All experiments were conducted on a laptop with Intel i5-10210U CPU and Manjaro Linux with the 5.15.155 kernel.

6.3.1 Implementation

To implement $\mathcal{A}ggr$'s TLS server, we integrate the operations specified in Section 6.2.2 with OpenSSL server version 3.2.0-alpha2. In addition to the OpenSSL's Curve25519 implementation, we use the Elligator functionality of the Monocypher[5] library to implement cryptographic parts of the protocol.

For the O-TEE, we use the Intel SGX SSL[6] library that embeds OpenSSL in an SGX enclave. We modify the TLS client functionality of this library for operations specified in Section 6.2.1. An API is added to send the heartbeat request containing the ODT following the Finished flow. Since the cryptographic setup of the ODT scheme is separate from the one for TLS, the scheme does not restrict the cryptographic configuration of existing TLS clients and servers.

6.3.2 Performance Results

On the experimental platform, we set up two TLS clients (a native client and one using O-TEE) and two TLS servers (a native server and an ODT server for verification). Since all TLS connections under measurement are local, the results are dominated by computation time rather than network delays, which more faithfully reflect the performance of the scheme. Both the ODT server and O-TEE sample the heap in five locations and try to perform a verification for all incoming and outgoing connections respectively.

[5]https://monocypher.org/

[6]https://github.com/intel/intel-sgx-ssl

We run two experiments, each repeated 1,000 times, to get the averaged data. The first assesses how the ODT server performs and how it differs from the native server. We use the curl tool to connect to both servers and measure the time between sending ClientHello and receiving ServerHello. Note that although curl uses the native TLS client, the ODT server still generates v using a randomly chosen group element u. Table 2 reports the results. The ODT server incurs a slowdown of less than 1 millisecond per request due to parsing the client nonce into u and calculating the commitment v. Among the addition time spent by the ODT server, the *PPET* commitment generation and Elligator decoding cost 0.33 *ms* and 0.17 *ms*, respectively. The performance difference with the native TLS server does not compromise verification obliviousness. A detailed analysis of side-channel attacks is presented in Section 8.3.

Table 2: Server response time measured using curl (in *ms*).

ODT Server (overall cost)	8.25 ± 0.71
(a) PPET commitment	0.33 ± 0.26
(b) Elligator decoding	0.17 ± 0.11
OpenSSL Server	7.93 ± 0.73

The second experiment measures the performance impact of O-TEE. We measure the time for a complete handshake session in four combinations of client-server setups. The results are shown in Table 3. Compared with the native TLS client, it costs O-TEE around 144*ms* more per handshake. The overhead is due to cryptographic operations (including generating u, encoding v, and calculating the ODT), witness generation, and extra time for SGX `ocall` and `ecall`. While none of these costs can be saved, we can reduce the handshake time by letting O-TEE send the Finished message before ODT and witness generation. This optimization does not affect security of the scheme.

Table 3: Total handshake duration in four combinations of TLS connections (in *ms*)

	ODT server	OpenSSL Server
O-TEE	164.30 ± 2.81	164.10 ± 2.72
OpenSSL Client	20.40 ± 0.65	20.08 ± 0.63

7 Security Analysis

7.1 Background on the Tamarin Prover

Tamarin [52, 37] is a tool for the mechanized analysis of security protocols. Tamarin works with a formal model of the security protocol, its desired properties, and the adversary. It is used to construct either: (1) a proof that the protocol is secure, i.e., its properties hold in the specified adversarial model (even when run with an unbounded number of protocol

participants), or (2) a counter-example, i.e., an attack on the protocol. Tamarin has been used for formal verification of a wide range of large-scale real-world protocols [6], including 5G authenticated key agreement [5], TLS v1.3 [14], and the electronic payment standard EMV [4].

Tamarin reasons using a symbolic model of cryptography, going back to the seminal work of Dolev and Yao [19]. Protocols are represented as labeled transition systems augmented with equational theories formalizing common cryptographic operators. This kind of model abstracts away from a low-level implementation of cryptography, focusing instead on the properties of idealized cryptographic operators.

7.2 Protocol Model in Tamarin

To prove binding integrity, we first model the protocol.

Protocol roles We specify five protocol roles: (1) an aggressor role that can create agents, (2) an agent role ($\mathcal{A}gt$) created by an aggressor and that can run on a device, (3) a device role that can run agents and be assigned an O-TEE, (4) a neutral party ($\mathcal{NTP}$) role that can instantiate O-TEE bound to a device, and (5) an O-TEE role.

Adversary model The Tamarin adversary represents the dishonest $\mathcal{A}dm$. This adversary has all the capabilities of the standard (Dolev-Yao) adversary used in symbolic models: it can read, create, modify, and block any messages created or sent over the network. Moreover, it can compromise any device and gain its capabilities, though it cannot compromise $\mathcal{NTP}$. On compromised devices, it can interrupt and resume the execution of an attached O-TEE and read, create, modify, and block all messages the device sends or receives over the network or from O-TEE. If it interrupts an O-TEE, the adversary can modify the data O-TEE reads. As explained in Section 4.2, the O-TEE measurement scheme ensures that the co-residence witness generated by O-TEE after an interrupt is invalid. We also assume that the adversary can mirror only a small subset of a running process's memory between devices instead of the whole memory. We justify this using the argument presented in Section 4.2.

Measurements As described in Section 4, the core of our protocol is the witness generation scheme that has two functions: Select and Measure. While the protocol measures the agent's memory, we model an abstraction of this and assume O-TEE measures *properties* of a process. Note that memory measurement is one way to achieve binding integrity and that other measurements, such as cache or control flow measurements, are possible, provided they identify the process.

Given $\mathcal{A}gt$ with a set of properties $\mathcal{P}(\mathcal{A}gt)$, O-TEE measures a subset of these properties. To select the subset, O-TEE uses the $\mathsf{Select}(k)$ function that takes as arguments a key k, and returns a set $\mathcal{S}$ of positions that are measured. We do not model subsets explicitly and instead leave them in their symbolic form $\mathcal{S} = \mathsf{Select}(k)$, where k represents the shared secret between O-TEE and the aggressor.

```
lemma binding_integrity:
 "All AGR O-TEE AGT prop #i.
 AcceptedVerificationWithAgent(AGR, O-TEE, AGT, prop) @i
 ==>  (Ex NP D #j #k #l #m #n.
     OTEERegisteredOnDevice(NP, O-TEE, D) @j &
     OTEEHonestPropertyRead(O-TEE, D, AGT, prop) @k &
     AggressorCreateAgent(AGR, AGT, prop) @l &
     AggrDHKey(AGR, DHE) @m & OTEEDHKey(O-TEE, DHE) @n)"
```

Figure 6: Binding integrity property as specified in Tamarin.

We model measurement as a function $\mathsf{Measure}(\mathcal{S}, \mathcal{P}(\mathcal{A}gt))$ that returns a witness w given the set of indices $\mathcal{S}$ and the properties $\mathcal{P}(\mathcal{A}gt)$ of an agent process. As already mentioned, the adversary can interrupt O-TEE and recover the indices $\mathcal{S}$ or modify the values O-TEE reads.

Verification We model verification in the same way we specify it in our scheme. The aggressor first verifies that signature σ is valid and belongs to a registered O-TEE, and then verifies if the witness response is correct. If both checks succeed, we consider the verification to be successful.

In summary, given public O-TEE measurement functions Select and Measure, we define $w = \mathsf{Measure}(\mathsf{Select}(k), \mathcal{P}(\mathcal{A}gt))$ to be the witness extracted from an agent process $\mathcal{A}gt$ with properties $\mathcal{P}(\mathcal{A}gt)$, given a key k shared between the aggressor and O-TEE. O-TEE's measurement and signature ensure that the aggressor can establish a binding between its agent process, O-TEE, and the TLS connection.

7.3 Binding Integrity

We formalize binding integrity from Section 2.3 as a trace property in Tamarin as shown in Figure 6. We prove that a successful verification by an aggressor AGR that believes it is talking with an O-TEE and that created its own agent process AGT exhibiting properties prop, implies: 1) that the O-TEE is a genuine O-TEE registered on device D by a neutral party NP, 2) that the O-TEE measured the agent process AGT residing on device D that exhibited the properties prop, 3) the aggressor AGR created the agent AGT with properties prop, and 4) AGR and O-TEE derived the same Diffie-Hellman secret DHE.

In Section 2.3 we note that the intrinsic challenge of ODT verification is to determine how entities to be authenticated are identified. In our ODT scheme, the aggressor uses distinct injected agents to build a namespace of devices. Since an agent's identity is ambiguous, i.e. there is no canonical way to refer to a piece of code, we use the properties that the code itself exhibits as the agent's identity. As explained in Section 4.2, our approach to witness generation guarantees co-residence: when the measurement succeeds, the process memory that was measured is on the same device as O-TEE. This, combined with the fact the aggressor connects to an

agent over a TLS channel, enables our ODT scheme to achieve binding integrity as proven in Tamarin.

7.4 Verification Obliviousness

We formalize verification obliviousness as the adversary's advantage over a random guess in differentiating a given process's two TLS handshakes: with a regular server and with $\mathcal{A}ggr$ performing ODT verification. Specifically, we define it as a game where the adversary ($\mathcal{A}dm$), given an $\mathcal{NTP}$-protected device D and all secrets in its O-TEE, is challenged to determine if ODT verification took place with an oracle that simulates either a normal TLS server or $\mathcal{A}ggr$ depending on a random bit. Note that verification obliviousness is not dependent on O-TEE's security. The formal definition is below.

Definition 3 (Verification Obliviousness)**.** *Let D be a device with O-TEE. Let $\mathcal{A}ggr$ be an aggressor and $p_{\mathcal{A}}$ be $\mathcal{A}ggr$'s agent on device D. Let $O(b)$ be an oracle that, with a random input bit b, simulates a standard TLS server for $b=0$ or $\mathcal{A}ggr$ otherwise. For all protocols ODT and all adversaries $\mathcal{A}dv$, we define the advantage function:*

$$\mathsf{adv}^{\mathsf{VO}}_{\mathcal{A}dv,ODT} = \left| \frac{1}{2} - \mathsf{Pr}\Big[\mathcal{A}ggr \xrightarrow{\$} p_{\mathcal{A}}; \{0,1\} \xrightarrow{\$} b; \mathcal{A}dv^{O(b)}(D, p_{\mathcal{A}}) \xrightarrow{\$} b' : b' == b\Big] \right|.$$

An ODT protocol satisfies verification obliviousness if, for all efficient algorithms $\mathcal{A}dv$ that have complete control over the device and its O-TEE, $\mathsf{adv}^{\mathsf{VO}}_{\mathcal{A}dv,ODT}$ is negligible.

To show our scheme satisfies Definition 3, we first prove that the verifier's message v of the *PPET* protocol in Figure 3 is random in $\mathbb{G}$. This is a necessary condition for Elligator to encode elliptic curve points into uniformly random looking strings, a fact we use in our verification obliviousness proof. We call this intermediate security property: *verifier message uniformity*.

Definition 4 (Verifier Message Uniformity)**.** *Let $\mathbb{G}$ be a group of order p and g a generator of the group. Let $\mathbb{W}$ be the domain of possible witnesses. For all PPET protocols PPET, all adversaries $\mathcal{A}dv$, we define the advantage function:*

$$\mathsf{adv}^{\mathsf{VM\_UNI}}_{\mathcal{A}dv,PPET} = \left| \frac{1}{2} - \mathsf{Pr}\Big[\mathcal{A}dv \xrightarrow{\$} u \in \mathbb{G}; \mathbb{Z}_p \xrightarrow{\$} s; \mathbb{W} \xrightarrow{\$} w'; v_0 = g^s u^{w'}; \mathbb{G} \xrightarrow{\$} v_1; \{0,1\} \xrightarrow{\$} b; \mathcal{A}dv(v_b) \xrightarrow{\$} b' : b' == b\Big] \right|.$$

A witness verification protocol PPET satisfies verifier message uniformity if, for all efficient algorithms $\mathcal{A}dv$, $\mathsf{adv}^{\mathsf{VM\_UNI}}_{\mathcal{A}dv,PPET}$ is negligible.

Lemma 1. *The Privacy-Preserving Equality Test Protocol PPET in Figure 3 satisfies verifier message uniformity.*

Proof. Observe that $u^{w'}$ is an element of $\mathbb{G}$. For a given uniformly random group element g^s, the probability that $g^s u^{w'}$ is equal to a given group element is $1/p$. Since this holds for any possible value of $u^{w'}$, v_0 is uniformly distributed in $\mathbb{G}$. Since v_1 is also uniformly distributed in $\mathbb{G}$, the statistical difference between v_0 and v_1 is zero. Hence, $\mathsf{adv}^{\mathsf{VM\_UNI}}_{\mathcal{A}dv,PPET}$ is negligible. □

Theorem 2. *Our ODT scheme satisfies Definition 3.*

Proof. In order to determine whether a verification took place, the adversary can run $p_{\mathcal{A}}$ and other processes of its choice on device D and look at: (1) the oracle's response v, and (2) the behavior of O-TEE. We discuss timing attacks and other side-channels in Section 8.

For (1), the only difference the adversary can detect between an honest TLS server and the aggressor is in how the ServerHello nonce is generated. When the aggressor sends $\iota^{-1}(v)$, the nonce's distribution might be different from a uniformly random string that is sent by a normal TLS server. However, according to Lemma 1, we know that v is uniformly random in $\mathbb{G}$. After v is Elligator encoded, we are guaranteed that an adversary has a negligible probability to differentiate the Elligator-encoded v from a uniformly random looking string. For (2), the adversary cannot use O-TEE as an oracle to test if a given process is performing verification, as O-TEE always behaves in the same way for all processes that invoke a TLS connection. Therefore, we conclude that $\mathsf{adv}^{\mathsf{VO}}_{\mathcal{A}dv,ODT}$ is negligible. □

The proof shows that $\mathcal{A}dm$ cannot use our ODT protocol to detect a verification attempt. It does not capture the probability that $\mathcal{A}dm$ can detect the agent. The likelihood of agent detection ultimately depends on the arms race between $\mathcal{A}ggr$ and $\mathcal{A}dm$, whose outcome is not prescribed by our design. We show below that detecting an agent for ODT verification is as hard as detecting TLS-using malware.

Discussion on Agent Detection According to several surveys [49, 40], it is popular for malware to use TLS to communicate with its backend server due to the widely used HTTPS protocol. For instance, the TLS-based Cobalt Strike Command & Control channel was widely used by ransomware as reported by CISCO [38] and was also later found in the SolarWinds attack [20]. Accompanying this trend is the evolution of malware detection techniques based on TLS anomalies, benefiting from advances in machine learning. Traffic data often used in anomaly analysis includes port numbers, the amount of transported data, connection duration, and handshake parameters [2]. The potential anomaly exhibited by malware is due to the involved task, such as downloading a payload or uploading stolen data. Since the ODT verification agent does not have such tasks, it can avoid detection by making ordinary TLS connections.

Our ODT scheme imposes no requirements on an agent's memory contents. Thus, memory contents involved in O-TEE's measurement do not have a specific pattern for $\mathcal{A}dm$ to

fingerprint an agent. $\mathcal{A}ggr$ can utilize existing anti-detection techniques [1], undocumented evasion techniques, and zero-day exploits to conceal its agent's existence. The mere existence of a TLS connection to a rarely-accessed server could appear suspicious in certain scenarios. $\mathcal{A}ggr$ may evade such detection by covertly compromising existing processes that use TLS and connecting them to its servers providing legitimate Internet services (e.g., DNS or NTS).

Lastly, we acknowledge that not all aggressors can invade target devices with a stealthy agent performing TLS connections. However, it is feasible for a state-backed aggressor, which is the primary subject of the ICRC's emblem application and international humanitarian law, to have such a capability. We leave it to future work to investigate ODT schema with less demanding requirements.

7.5 Security Preservation

Security preservation concerns a TLS server that extracts information about D from its TLS connections. We formulate it as the adversary's winning probability in the game where it guesses the value stored at a specific memory location. We quantify the probability that the adversary guesses correctly in relation to the memory's size, the number of memory locations O-TEE measures, the number of values each position can have, and the number of adversary queries. In the proof, we show that this probability is negligible.

Definition 5 (Security preservation)**.** *Let D be a device with O-TEE. Let X be the set of values that can be stored at a memory location. Let p be a process on D with memory contents $(x_i)_{i\in I}$, where I is the index set of memory locations and $x_i \in X$ is the value stored at location i. Let O_p be an oracle that, for the process p, simulates the execution of O-TEE on device D that measures C locations up to q times. For all protocols* ODT*, all adversaries $\mathcal{A}dv$ that output a set $\{x'_{j_1}, x'_{j_2}, \ldots, x'_{j_C}\}$ where $x'_j \in X$, we define the advantage function:*

$$\mathsf{adv}^{\mathsf{SP}}_{\mathcal{A}dv,ODT} = \mathsf{Pr}[I_t = \{i_1, i_2, \ldots, i_C\}, \forall i \in I; \\ \mathcal{A}dv^{O_p}(I_t) \xrightarrow{\$} \{x'_{i_1}, \ldots, x'_{i_C}\} : \forall i \in I_t . x'_i == x_i].$$

An ODT protocol satisfies security preservation if, for all efficient algorithms $\mathcal{A}dv$, $\mathsf{adv}^{\mathsf{SP}}_{\mathcal{A}dv,ODT}$ is negligible.

Theorem 3. *Our ODT scheme satisfies Definition 5.* □

Our proof (in the appendix) considers two different attack scenarios: extracting information from a low-entropy process, and extracting information about a 256-bit secret key with the knowledge of contents at all other memory locations. For the first scenario, we show that $\mathcal{A}dv$ only has a negligible success probability given that C is sufficiently large. We can lower $\mathcal{A}dv$'s success chance by measuring more locations. However, this demands $\mathcal{A}ggr$ to have a more complete and accurate view of the agent's memory.

For the second scenario, assuming that O-TEE measures one 64-bit word, we show that $\mathcal{A}dv$ learns at least one of the four words of the 256-bit secret key with a small (2^{-59}) probability after making 10^6 guesses for a process with one megabyte of memory. The relatively high probability is not an issue considering $\mathcal{A}dv$ relies on the device to initiate a TLS connection with its server. This scenario also represents an ideal case for $\mathcal{A}dv$ where it knows all other memory locations, which is generally infeasible in practice. Using more locations to measure also lowers the probability.

8 Discussions and Related Work

8.1 Alternatives to TLS v1.3

Among the protocols in the TLS family, the ODT scheme is compatible with DTLS v1.3 [48], a variant of TLS v1.3 for secure UDP communications. However, TLS v1.2 [47] (and its DTLS variant) do not fit the ODT scheme well. Servers using TLS v1.2 typically use public-key encryption-based handshakes where ServerHello is sent before the handshake secret is generated. As a result, $\mathcal{A}ggr$ cannot send its commitment v as a server nonce and instead has to send it in TLS records that carry application data. This undermines verification obliviousness since there is no uniform pattern for TLS servers' application data sending and O-TEE has to respond differently. It is thus seems infeasible to prove that $\mathcal{A}ggr$'s message is indistinguishable from those sent by regular TLS servers.

Although there are other protocols involving random data exchanges and shared keys, none of them is more suitable for the ODT scheme than TLS. Application layer protocols (such as Tor [18], IRC, and SSH) are not generic enough for all applications. Network layer protocols (such as the Internet Key Exchange protocol [29] for IPSec) are not application specific; hence, it is difficult to link the connection to a particular process.

8.2 Compatibility with TLS Implementation

$\mathcal{A}ggr$'s server is fully compatible with the TLS specification and provides services to all TLS clients. However, we discovered that the heartbeat request sent by O-TEE in our prototype disrupts TLS connections with servers that use OpenSSL. Although TLS v1.3 [46] defines the heartbeat extension, OpenSSL maintainers removed it from the code due to lack of real-world use-cases and concerns over implementation bugs.[7] We believe that the ICRC emblems are an important real-world use-case that would warrant the re-introduction of heartbeats into OpenSSL.

To avoid the compatibility issue, an O-TEE implementation can instead use an ICMP Echo-Request messages to deliver ODTs to TLS servers. This alternative does not compromise

[7] https://github.com/openssl/openssl/issues/4856

binding integrity, since our scheme does not depend on the security of the TLS connection to protect this message.

8.3 Side-Channel Attacks on Verification Obliviousness

We briefly consider the risks of compromising verification obliviousness via side-channel attacks on $\mathcal{A}ggr$'s messages and O-TEE. Since $\mathcal{A}ggr$ is remote from $\mathcal{A}dm$, the only side-channel observable to $\mathcal{A}dm$ is based on time intervals between sending ClientHello and arrival of ServerHello. However, $\mathcal{A}ggr$ can introduce an artificial delay in its non-ODT computations to mask the difference. Unless $\mathcal{A}dm$ knows the hardware that $\mathcal{A}ggr$'s server runs, it cannot detect this delay.

On the client side, the proposed ODT scheme eliminates the device's side-channel leakage about ODT verification. O-TEEs behave in the same fashion regardless whether the TLS peer is a regular server or $\mathcal{A}ggr$. Furthermore, the scheme does not require $\mathcal{A}ggr$'s agent to carry out special operations for ODT verification. For example, it does not perform a local attestation against an enclave O-TEE. While we acknowledge that there are side-channel attacks against secrets of various TEEs, coping with them is orthogonal to this work.

8.4 Limitations of Our ODT Scheme

Our scheme cannot be deployed on devices without a hardware-based TEE or virtual machines in a cloud even if confidential VM techniques such as Intel TDX are in place. It also cannot help aggressors without an agent on the target devices. Note that such aggressors can resort to ADEM [32] to perform preliminary checks. Also, $\mathcal{NTP}$ can deploy ADEM and ODT side by side.

The ODT schemes guarantees that if an ODT passes $\mathcal{A}ggr$'s verification, the device is $\mathcal{NTP}$-protected. However, the converse is not guaranteed. Thus, $\mathcal{A}ggr$ cannot conclude that the device is not $\mathcal{NTP}$-protected if verification fails. While it seems impossible to prove an ODT's non-existence, it is an open problem how to use the ODT verification transcript as an unforgeable proof for $\mathcal{A}ggr$'s due diligence.

Another limitation is the dependency on TLS v1.3, especially for the Diffie-Hellman handshake. It may require changes to work with future versions of TLS or a successor protocol suite. However, if the handshake algorithm is unchanged, we are optimistic that the adaptation will only involve message encoding, rather than changes of core algorithms.

8.5 Related Work

As mentioned earlier, ADEM [32] proposed an emblem based on digital signatures for use in network communications. While it ensures verification obliviousness, the resulting emblem-device binding is insecure against a malicious kernel.

The functionality of ODT is related to RoT attestation whereby a prover device attests its RoT attributes to a trusted verifier. These attributes include RoT genuineness [10, 51], RoT presence [55], and the distance [15, 17] to the verifier. More specifically, our protocol is related to remotely attested TLS protocols [23, 54, 31, 39]. These protocols combine TLS with remote attestation to create *trusted channels*, i.e. secure channels where one or both endpoint(s) is/are attested to the other. However, they are incompatible with verification obliviousness.

In terms of verification obliviousness, the ODT scheme is related to privacy protection techniques for hiding sensitive information that would otherwise be exposed to observers monitoring system or network actions. Related work includes oblivious RAM (ORAM) [21] and Private Information Retrieval (PIR) [13], which prevent an entity's interest in particular data items from leaking due to its actions on the data set. Anonymous communications techniques such as Tor [18] and mix networks [28], prevent sender or receiver information leakage from network activities.

The proposed PPET protocol provides another type of privacy protection, which primarily considers the privacy of data used in multi-party computation. Similar techniques include oblivious transfer [30], zero-knowledge proofs [22], secret handshakes [3] and private set intersection [50, 44, 24, 16].

9 Conclusions

The need for digital emblems arises in the context of cyber-conflicts and international humanitarian law. Developing effective solutions triggers novel requirements for labeling devices in a way that provides binding integrity, while ensuring obliviousness for the verifier and security preservation for the provers. In this paper, we defined this problem, fleshed out its requirements, and provided the first solution, along with a prototype implementation. Future work includes experiments with the proposed scheme to further assess its practicality and exploring open problems, such as ascertaining the absence of oblivious digital tokens.

Acknowledgments

Mihael Liskij's and David Basin's research was funded by the Werner Siemens-Stiftung (WSS) as part of the Centre for Cyber Trust (CECYT). We thank the WSS for their contribution.

Gene Tsudik's work was supported in part by NSF Award SATC1956393 and NSA Award H98230-22-1-0308.

This research was also partially supported by the Lee Kong Chian Fellowship awarded to Xuhua Ding by Singapore Management University.

Ethics Considerations

This research is directly motivated by the ICRC's initiative to introduce digital emblems in cyberspace. Such emblems should provide a way for protected parties to signal their right to protection in cyberspace, analogous to the use of a red cross, crescent, or crystal to mark, e.g., medical relief workers and medical facilities/objects in the physical world. This, in turn, provides a basis for extending international humanitarian law to cyberspace and making the world safer, in wartime. Our research focused on the technical foundations for digital emblems: their requirements, design, implementation, evaluation, and security proofs. There are no ethical concerns, since this work neither involved human subjects nor attacked systems or infrastructure of any kind. Hence, there was no need for approval by the authors' institutions' ethical review boards.

Open Science

The Tamarin model, Tamarin proofs, the ODT prototype, and all raw measurement results are available for download at Zenodo [35].

References

[1] Amir Afianian et al. "Malware Dynamic Analysis Evasion Techniques: A Survey". In: *ACM Comput. Surv.* 52.6 (Nov. 2019). ISSN: 0360-0300. DOI: 10.1145/3365001.

[2] Blake Anderson and David McGrew. "Identifying Encrypted Malware Traffic with Contextual Flow Data". In: *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*. AISec '16. New York, NY, USA: Association for Computing Machinery, Oct. 28, 2016, pp. 35–46. DOI: 10.1145/2996758.2996768.

[3] D. Balfanz et al. "Secret Handshakes from Pairing-Based Key Agreements". In: *2003 Symposium on Security and Privacy, 2003.* 2003, pp. 180–196. DOI: 10.1109/SECPRI.2003.1199336.

[4] David Basin, Ralf Sasse, and Jorge Toro-Pozo. "The EMV Standard: Break, Fix, Verify". In: *2021 IEEE Symposium on Security and Privacy (SP)*. May 2021, pp. 1766–1781. DOI: 10.1109/SP40001.2021.00037.

[5] David Basin et al. "A Formal Analysis of 5G Authentication". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 1383–1396. DOI: 10.1145/3243734.3243846.

[6] David Basin et al. "Tamarin: Verification of Large-Scale, Real-World, Cryptographic Protocols". In: *IEEE Security & Privacy* 20.3 (2022), pp. 24–32. DOI: 10.1109/MSEC.2022.3154689.

[7] Daniel J. Bernstein. "Curve25519: New Diffie-Hellman Speed Records". In: *Public Key Cryptography - PKC 2006*. Vol. 3958. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 207–228. DOI: 10.1007/11745853_14.

[8] Daniel J. Bernstein et al. "Elligator: Elliptic-Curve Points Indistinguishable from Uniform Random Strings". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security - CCS '13*. Berlin, Germany: ACM Press, 2013, pp. 967–980. DOI: 10.1145/2508859.2516734.

[9] Dan Boneh. "The Decision Diffie-Hellman Problem". In: *Algorithmic Number Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 48–63. DOI: 10.1007/BFb0054851.

[10] Jan Camenisch et al. "One TPM to Bind Them All: Fixing TPM 2.0 for Provably Secure Anonymous Attestation". In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 901–920. DOI: 10.1109/SP.2017.22.

[11] Xavier Carpent, Norrathep Rattanavipanon, and Gene Tsudik. "Remote attestation of IoT devices via SMARM: Shuffled measurements against roving malware". In: *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 2018, pp. 9–16. DOI: 10.1109/HST.2018.8383885.

[12] Guoxing Chen et al. "SgxPectre: Stealing Intel Secrets from SGX Enclaves Via Speculative Execution". In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2019, pp. 142–157. DOI: 10.1109/MSEC.2019.2963021.

[13] Benny Chor et al. "Private Information Retrieval". In: *Journal of the ACM* 45.6 (Nov. 1, 1998), pp. 965–981. DOI: 10.1145/293347.293350.

[14] Cas Cremers et al. "A Comprehensive Symbolic Analysis of TLS 1.3". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Oct. 30, 2017, pp. 1773–1788. DOI: 10.1145/3133956.3134063.

[15] Cas Cremers et al. "Distance Hijacking Attacks on Distance Bounding Protocols". In: *2012 IEEE Symposium on Security and Privacy*. 2012, pp. 113–127. DOI: 10.1109/SP.2012.17.

[16] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. "Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model". In: *Advances in Cryptology - ASIACRYPT 2010*. Berlin, Heidelberg: Springer, 2010, pp. 213–231. DOI: 10.1007/978-3-642-17373-8_13.

[17] Aritra Dhar et al. "ProximiTEE: Hardened SGX Attestation by Proximity Verification". In: *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*. CODASPY '20. 2020, pp. 5–16. DOI: 10.1145/3374664.3375726.

[18] Roger Dingledine, Nick Mathewson, and Paul Syverson. "Tor: The Second-Generation Onion Router". In: *13th USENIX Security Symposium (USENIX Security 04)*. Vol. 4. USENIX Association, Aug. 2004, pp. 303–320. URL: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router (visited on 01/28/2025).

[19] D. Dolev and A. Yao. "On the security of public key protocols". In: *IEEE Transactions on Information Theory* 29.2 (1983), pp. 198–208. DOI: 10.1109/TIT.1983.1056650.

[20] FireEye. *A highly evasive attacker leverages a supply chain attack trojanizing SolarWinds Orion business software updates in order to distribute SUNBURST malware.* Google Cloud Blog. URL: https://cloud.google.com/blog/topics/threat-intelligence/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor (visited on 01/28/2025).

[21] O. Goldreich and R. Ostrovsky. "Software Protection and Simulation on Oblivious Ram". In: *Journal of the ACM* 43.3 (1996), pp. 431–473. DOI: 10.1145/233551.233553.

[22] Oded Goldreich and Yair Oren. "Definitions and properties of zero-knowledge proof systems". In: *Journal of Cryptology* 7.1 (1994), pp. 1–32. DOI: 10.1007/BF00195207.

[23] Gilang Mentari Hamidy et al. "TC4SE: A High-Performance Trusted Channel Mechanism for Secure Enclave-Based Trusted Execution Environments". In: *Information Security*. Vol. 14411. Cham: Springer Nature Switzerland, 2023, pp. 246–264. DOI: 10.1007/978-3-031-49187-0_13.

[24] Bernardo A. Huberman, Matt Franklin, and Tad Hogg. "Enhancing Privacy and Trust in Electronic Communities". In: *Proceedings of the 1st ACM Conference on Electronic Commerce*. Nov. 1999, pp. 78–86. DOI: 10.1145/336992.337012.

[25] Eric M Hutchins, Michael J Cloppert, Rohan M Amin, et al. "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains". In: *Leading Issues in Information Warfare & Security Research* 1.1 (2011), p. 80. URL: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf (visited on 01/28/2025).

[26] ICRC. *Digitalizing the Red Cross, Red Crescent, and Red Crystal emblems*. Geneva: ICRC, 2022. URL: https://www.icrc.org/sites/default/files/document_new/file_list/icrc_digitalizing_the_rcrc_emblem.pdf (visited on 01/28/2025).

[27] Intel. "Intel Trust Domain Extension". In: *White Paper* (2023). URL: https://cdrdv2-public.intel.com/690419/TDX-Whitepaper-February2022.pdf (visited on 01/28/2025).

[28] Markus Jakobsson and Ari Juels. "An optimally robust hybrid mix network". In: *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing*. 2001, pp. 284–292. DOI: 10.1145/383962.384046.

[29] Charlie Kaufman et al. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296. Internet Engineering Task Force, Oct. 2014. DOI: 10.17487/RFC7296.

[30] Joe Kilian. "Founding crytpography on oblivious transfer". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. 1988, pp. 20–31. DOI: 10.1145/62212.62215.

[31] Thomas Knauth et al. "Integrating Remote Attestation with Transport Layer Security". In: *CoRR* abs/1801.05863 (2018). arXiv: 1801.05863.

[32] Felix Linker and David Basin. "ADEM: An Authentic Digital EMblem". In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, Nov. 21, 2023, pp. 2815–2829. DOI: 10.1145/3576915.3616578.

[33] Felix Linker et al. *Problem Statement for a Digital Emblem*. Tech. rep. Work in Progress. Internet Engineering Task Force, 2024. 11 pp. URL: https://datatracker.ietf.org/doc/draft-linker-digital-emblem/01/ (visited on 01/28/2025).

[34] Helger Lipmaa. "Verifiable Homomorphic Oblivious Transfer and Private Equality Test". In: *Advances in Cryptology - ASIACRYPT 2003*. Vol. 2894. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 416–433. DOI: 10.1007/978-3-540-40061-5_27.

[35] Mihael Liskij et al. *Oblivious Digital Tokens*. Jan. 2025. DOI: 10.5281/ZENODO.14655914. (Visited on 01/29/2025).

[36] Catherine Meadows. "A More Efficient Cryptographic Matchmaking Protocol for Use in the Absence of a Continuously Available Third Party". In: *1986 IEEE Symposium on Security and Privacy*. Apr. 1986, pp. 134–134. DOI: 10.1109/SP.1986.10022.

[37] Simon Meier et al. "The TAMARIN Prover for the Symbolic Analysis of Security Protocols". In: *Computer Aided Verification*. Springer, 2013, pp. 696–701. DOI: 10.1007/978-3-642-39799-8_48.

[38] Jonathan Munshaw. *Quarterly Report: Incident Response Trends in Summer 2020*. Cisco Talos Blog. Sept. 1, 2020. URL: https://blog.talosintelligence.com/ctir-quarterly-trends-q4-2020/ (visited on 01/28/2025).

[39] Arto Niemi, Vasile Adrian Bogdan Pop, and Jan-Erik Ekberg. "Trusted Sockets Layer: A TLS 1.3 Based Trusted Channel Protocol". In: *Secure IT Systems*. Cham: Springer International Publishing, 2021, pp. 175–191. DOI: 10.1007/978-3-030-91625-1_10.

[40] Chaeyeon Oh, Joonseo Ha, and Heejun Roh. "A survey on TLS-encrypted malware network traffic analysis applicable to security operations centers". In: *Applied Sciences* 12.1 (2021), p. 155. DOI: 10.3390/app12010155.

[41] Oleksii Oleksenko et al. "Varys: Protecting SGX Enclaves from Practical Side-Channel Attacks". In: *2018 Usenix Annual Technical Conference (USENIX ATC 18)*. 2018, pp. 227–240. ISBN: ISBN 978-1-939133-01-4. URL: https://www.usenix.org/conference/atc18/presentation/oleksenko (visited on 01/28/2025).

[42] Bryan Parno, Jonathan M. McCune, and Adrian Perrig. *Bootstrapping Trust in Modern Computers*. Vol. 10. SpringerBriefs in Computer Science. New York, NY: Springer, 2011. DOI: 10.1007/978-1-4614-1460-5.

[43] Team PaX. *PaX address space layout randomization (ASLR)*. 2003. URL: http://pax.grsecurity.net/docs/aslr.txt (visited on 01/27/2025).

[44] Benny Pinkas, Thomas Schneider, and Michael Zohner. "Scalable Private Set Intersection Based on OT Extension". In: *ACM Transactions on Privacy and Security* 21.2 (Jan. 2018). DOI: 10.1145/3154794.

[45] Hany Ragab et al. "Crosstalk: Speculative data leaks across cores are real". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 1852–1867. DOI: 10.1109/SP40001.2021.00020.

[46] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Internet Engineering Task Force, Aug. 2018. DOI: 10.17487/RFC8446.

[47] Eric Rescorla and Tim Dierks. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. Internet Engineering Task Force, Aug. 2008. DOI: 10.17487/RFC5246.

[48] Eric Rescorla, Hannes Tschofenig, and Nagena Modadugu. *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3*. RFC 9147. Internet Engineering Task Force, Apr. 2022. DOI: 10.17487/RFC9147.

[49] Olivier Roques. "Detecting Malware in TLS Traffic". MA thesis. Imperial College London, 2019. URL: https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1819-pg-projects/Detecting-Malware-in-TLS-Traf%EF%AC%81c.pdf (visited on 01/28/2025).

[50] Mike Rosulek and Ni Trieu. "Compact and Malicious Private Set Intersection for Small Sets". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Nov. 12, 2021, pp. 1166–1181. DOI: 10.1145/3460120.3484778.

[51] Vinnie Scarlata et al. *Supporting third party attestation for Intel® SGX with Intel® data center attestation primitives*. 2018. URL: https://www.intel.com/content/dam/develop/external/us/en/documents/intel-sgx-support-for-third-party-attestation-801017.pdf (visited on 01/27/2025).

[52] Benedikt Schmidt et al. "Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties". In: *2012 IEEE 25th Computer Security Foundations Symposium*. June 2012, pp. 78–94. DOI: 10.1109/CSF.2012.25.

[53] Hovav Shacham et al. "On the effectiveness of address-space randomization". In: *Proceedings of the 11th ACM Conference on Computer and Communications Security*. 2004, pp. 298–307. DOI: 10.1145/1030083.1030124.

[54] Robert Walther, Carsten Weinhold, and Michael Roitzsch. "RATLS: Integrating Transport Layer Security with Remote Attestation". In: *Applied Cryptography and Network Security Workshops*. Cham: Springer International Publishing, 2022, pp. 361–379. DOI: 10.1007/978-3-031-16815-4_20.

[55] Zhangkai Zhang et al. "Presence Attestation: The Missing Link in Dynamic Trust Bootstrapping". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 89–102. DOI: 10.1145/3133956.3134094.

Appendix A: Theorem 1 Proof

Proof. We use a hybrid proof to show that the PPET protocol *PPET* from Figure 3 satisfies privacy preservation. We first define starting and end distributions:

$$\mathcal{D}_b = \{\mathbb{G} \xrightarrow{\$} u; \mathbb{Z}_p \xrightarrow{\$} t; \mathbb{W} \xrightarrow{\$} w; \mathcal{A}dv(u) \xrightarrow{\$} v \in \mathbb{G}; \\ z_0 = v^t u^{-wt}; \mathbb{G} \xrightarrow{\$} z_1 : (g^t, z_b)\}. \tag{1}$$

Note that $\mathcal{D}_0$ exactly matches the inputs given to the adversary in the privacy preservation game when $b = 0$ and $\mathcal{D}_1$ matches the inputs when $b = 1$. From this, we have $\mathsf{adv}^{\mathsf{PP}}_{\mathcal{A}dv,PPET} = \mathsf{adv}^{\Delta}_{\mathcal{A}dv,\mathcal{D}_0,\mathcal{D}_1}$. To prove that the protocol is secure, we show that $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable.

First hybrid We define $\mathcal{H}_1$ and show that it is statistically indistinguishable from $\mathcal{D}_0$:

$$\mathcal{H}_1 = \{\mathbb{G} \xrightarrow{\$} u; \mathbb{Z}_p \xrightarrow{\$} t; \mathbb{W} \xrightarrow{\$} w; \mathcal{A}dv(u) \xrightarrow{\$} u^x; \\ z = u^{xt-wt} : g^t, z\}. \tag{2}$$

By the protocol, v must be an element from $\mathbb{G}$. Therefore, there exists an x such that u^x is equal to v. Hence this is just a syntactic change and we have $\mathsf{adv}^{\Delta}_{\mathcal{A}dv,\mathcal{D}_0,\mathcal{H}_1} = 0$.

Second hybrid We define $\mathcal{H}_2$ and show that it is statistically indistinguishable from $\mathcal{H}_1$:

$$\mathcal{H}_2 = \{\mathbb{G} \xrightarrow{\$} u; \mathbb{Z}_p \xrightarrow{\$} t; \mathbb{W} \setminus x \xrightarrow{\$} w; \mathcal{A}dv(u) \xrightarrow{\$} u^x; \\ z = u^{xt-wt} : g^t, z\}. \tag{3}$$

The only change is that we are sampling w from a smaller set. The adversary can distinguish $\mathcal{H}_1$ and $\mathcal{H}_2$ with an advantage of at most the probability of outputting such an x. In the real world, this advantage represents the adversary's a priori knowledge of the distribution of the witness values. However, as stated in Definition 2, we perform this proof under the assumption that the adversary guesses the witness incorrectly. Therefore, we consider the advantage of the adversary in this step to be zero, i.e. $\mathsf{adv}^{\Delta}_{\mathcal{A}dv,\mathcal{H}_1,\mathcal{H}_2} = 0$, when the adversary guesses the witness incorrectly.

Third hybrid We define $\mathcal{H}_3$ and show that it is statistically indistinguishable from $\mathcal{H}_2$:

$$\mathcal{H}_3 = \{\mathbb{G} \xrightarrow{\$} u; \mathbb{Z}_p \xrightarrow{\$} t; \mathcal{A}dv(u) \xrightarrow{\$} u^x; \\ \mathbb{W} \setminus 0 \xrightarrow{\$} k; z = u^{kt} : g^t, z\}. \tag{4}$$

This is again just a syntactic change where we rewrite $k = x - w \in \mathbb{W} \setminus 0$, therefore $\mathsf{adv}^{\Delta}_{\mathcal{A}dv,\mathcal{H}_2,\mathcal{H}_3} = 0$. Note that the distribution of k depends on the adversary's distribution of witnesses embedded in x. Because the adversary might have some a priori knowledge on the distribution of witnesses, we cannot assume k is sampled uniformly.

Fourth hybrid We define $\mathcal{H}_4$ and show that it is computationally indistinguishable from $\mathcal{H}_3$:

$$\mathcal{H}_4 = \{\mathbb{G} \xrightarrow{\$} u; \mathbb{Z}_p \xrightarrow{\$} t; \mathcal{A}dv(u) \xrightarrow{\$} u^x; \\ \mathbb{W} \setminus 0 \xrightarrow{\$} k; \mathbb{Z}_p \xrightarrow{\$} c; z = g^{kc} : g^t, z\}. \tag{5}$$

We uniformly sample c and use it instead of t to compute z. We do a reduction to the Decisional Diffie-Hellman (DDH) problem and show that $\mathsf{adv}^{\Delta}_{\mathcal{A}dv,\mathcal{H}_3,\mathcal{H}_4}$ is negligible. We assume that there exists an algorithm $\mathcal{A}dv$ that can efficiently distinguish $\mathcal{H}_3$ and $\mathcal{H}_4$. We build an algorithm $\mathcal{R}$ that uses $\mathcal{A}dv$ to break DDH.

$\mathcal{R}$ receives from its DDH challenger $\alpha = g^a, \beta = g^b, \gamma = g^c$. It simulates $\mathcal{A}dv$'s game and uses α instead of u, β instead of g^t, and γ instead of u^t. When $b = 0$ in $\mathcal{R}$'s game, $\mathcal{R}$ outputs (β, g^{kab}) that exactly matches the distribution $\mathcal{H}_3$. When $b = 1$ in $\mathcal{R}$'s game, $\mathcal{R}$ outputs (β, g^{kc}) that exactly matches the distribution $\mathcal{H}_4$. If $\mathcal{A}dv$ outputs $b' = 0$ as its own guess $\mathcal{R}$ outputs 0, and if $\mathcal{A}dv$ outputs $b' = 1$ it outputs 1. Because $\mathcal{R}$ perfectly simulates $\mathcal{A}dv$'s distinguishing game, we conclude that $\mathsf{adv}^{\mathsf{DDH}}_{\mathcal{R}} = \mathsf{adv}^{\Delta}_{\mathcal{A}dv,\mathcal{H}_3,\mathcal{H}_4}$. However, this is a contradiction because DDH is difficult in the group $\mathbb{G}$. Hence $\mathsf{adv}^{\Delta}_{\mathcal{A}dv,\mathcal{H}_3,\mathcal{H}_4}$ is negligible.

Last step We show $\mathsf{adv}^{\Delta}_{\mathcal{A}dv,\mathcal{H}_4,\mathcal{D}_1} = 0$. For convenience, $\mathcal{D}_1$ is:

$$\mathcal{D}_1 = \{\mathbb{G} \xrightarrow{\$} u; \mathbb{Z}_p \xrightarrow{\$} t; \mathcal{A}dv(u) \xrightarrow{\$} v; \mathbb{G} \xrightarrow{\$} z_1 : (g^t, z_1)\}. \tag{6}$$

While the distribution of k is not uniform, combining it with a uniformly random variable c ensures that g^{kc} is uniform. To prove that we use the fact that the order of $\mathbb{G}$ is prime. For any value of k, g^k is a generator of the group. Because c takes all possible values from $\mathbb{Z}_p$, the probability that g^{kc} is equal to any group element is $1/p$. Since this holds for every k (no matter the distribution of k), we can conclude that the probability that g^{kc} is equal to a specific group element is also $1/p$. Therefore, the distributions $\mathcal{H}_4$ and $\mathcal{D}_1$ are statistically indistinguishable.

Conclusion We have shown for all distributions in our chain that they are either computationally or statistically indistinguishable from their neighbors. Using the hybrid argument, we conclude that $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable and therefore the advantage $\mathsf{adv}^{\mathsf{PP}}_{\mathcal{A}dv}$ of the adversary $\mathcal{A}dv$ in winning the PP game is negligible. □

Appendix B: Theorem 3 Proof

We argue this, considering two scenarios: (i) the adversary has partial knowledge of the memory and (ii) the adversary has full knowledge of the entire memory except for a 256

bit secret. The first scenario models a general brute force attack where $\mathcal{A}dv$ attempts to extract information from a system that has a relatively low entropy. We show that $\mathcal{A}dv$ has a negligible advantage given that the number of measured locations C is our security parameter. The second scenario models a targeted attack where the adversary is interested in extracting a cryptographic key from a process's memory. We use these cases to show that our scheme remains practically secure even against extremely powerful adversaries.

(i) Assume, for simplicity, that O-TEE measures 64 bit locations and that the adversary knows k out of 64 bits for every measurable location but not all of them. For the first measurement, the adversary has a probability of $(|X|-2^k)^{-C}$ to guess correctly. After l guesses over the same subset of measured values, the probability of the next guess being correct increases to $\left((|X|-2^k)^C-l\right)^{-1}$. On average, each subset is measured $q\binom{|I|}{C}^{-1}$ times. To simplify the calculations, we assume the adversary correctly guesses with the probability of the last guess $\left(\left(|X|-2^k\right)^C-q\binom{|I|}{C}^{-1}\right)^{-1}$, which provides an upper bound on the adversary's chance of success. The probability P that the adversary wins in the game by correctly guessing at least once using q queries is then

$$P = 1-\left(1-\left(\left(|X|-2^k\right)^C-q\binom{|I|}{C}^{-1}\right)^{-1}\right)^q.$$

Note that we interpret the adversary's output as an additional query and absorb it into q. This shifts the domain of q by one and ensures that q is greater than zero.

A function is defined to be negligible if $\lim_{n\to\infty} f(n)n^s = 0$ for all $s > 0$. We set $f(n) := P(C)$ and show that $\lim_{C\to\infty} P(C)C^s$ goes to zero when C is our security parameter. First, we simplify $P(C)$ to $\bar{P}(C) = 1-(1-(2^C-q)^{-1})^q$ by setting $\binom{|I|}{C}^{-1}$ to one and $|X|-2^k$ to two. Both of these changes do not diminish the adversary's winning probability, hence $P(C) \leq \bar{P}(C)$. Let $e = (2^C-q)^{-1}$, we calculate:

$$\bar{P}(C) = (1-(1-e)^q) = e\sum_{i=0}^{q-1}(1-e)^i < eq = q(2^C-q)^{-1}.$$

It is easy to see that $\lim_{C\to\infty} q(2^C-q)^{-1}C^s = 0$. Since $P(C)$ is bounded from above by $q(2^C-q)^{-1}$, we have $\lim_{C\to\infty} P(C)C^s = 0$ and therefore the probability the adversary wins in the security preservation game is negligible.

For the second scenario (ii), we assume that the 256 bit secret is aligned to the measured positions, i.e. four positions cover it fully, that O-TEE measures only one 64 bit position, and that the adversary knows all memory contents except the secret. We slightly modify the security game by fixing I_t to be the set of four locations containing the key. The adversary wins if it guesses correctly at least one of the four locations.

The probability of correctly guessing the value of the key after l guesses on the same position is $(|X|-l)^{-1}$. Each position will be measured $q/|I|$ times on average. Similarly to the previous scenario, we assume that the adversary is always guessing with the probability of the last guess giving us the formula $(|X|-q/|I|)^{-1}$. The probability of the adversary winning the game by guessing correctly at least one of the four key positions using q queries is then

$$1-\left(1-\left(|X|-q/|I|\right)^{-1}\right)^{4q/|I|}.$$

For a one megabyte program, the adversary has a probability of 2^{-59} of winning in the game assuming it can make 10^6 requests. While this probability is relatively high for cryptographic standards, we believe this is not an issue in practice. With the results from our first scenario, we can increase the security of our scheme by measuring more locations. Moreover, the adversary does not have full freedom of establishing requests and must rely on the victim process to initiate a connection. Finally, an adversary that has perfect knowledge of every memory location at every moment a TLS connection opens is highly unlikely.