T3E: A Practical Solution to Trusted Time in Secure Enclaves

Gilang Mentari Hamidy, Pieter Philippaerts, Wouter Joosen

Presented at NSS'23 (University of Kent, 14-16 August 2023)

KU LEUVEN DistriNet

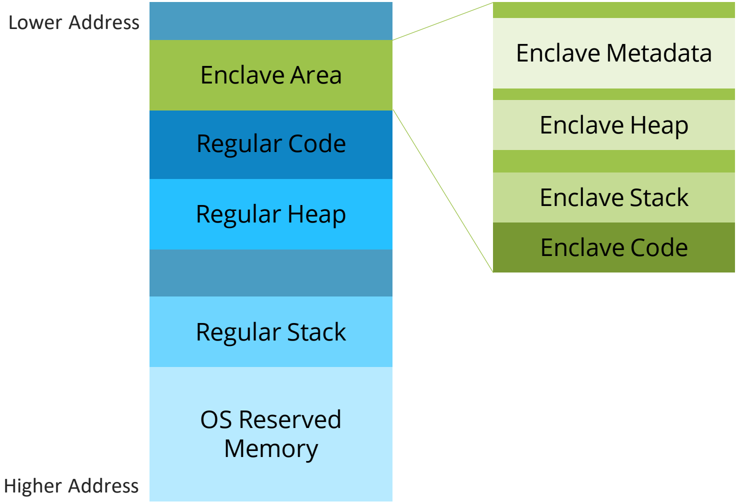
Outline

- Crash Course on TEE and Secure Enclaves
- Use Case of Trusted Time
- Background and Current Situation
- Related Works
- TPM at a Glance
- Approach
- Security Analysis
- Performance
- Conclusion

Crash Course on Trusted Execution

- **Trusted Execution Environment (TEE)**
  - **Secure and isolated environment** that provides critical functionality, particularly that requires trusts or security assurance. **Typically enforced by hardware**
  - Intel SGX is an example of a TEE
- **TEE provides Confidentiality, Integrity, and Authenticity in program execution**
- **Two Domains in TEE**
  - Untrusted World a.k.a. Rich Execution
  - Trusted World a.k.a. Trusted Execution

- **Intel SGX Enclave**
  - Protected memory region where only the trusted code can access
  - **Enclave size is limited, so memory footprint is important**



Use Case of Trusted Time

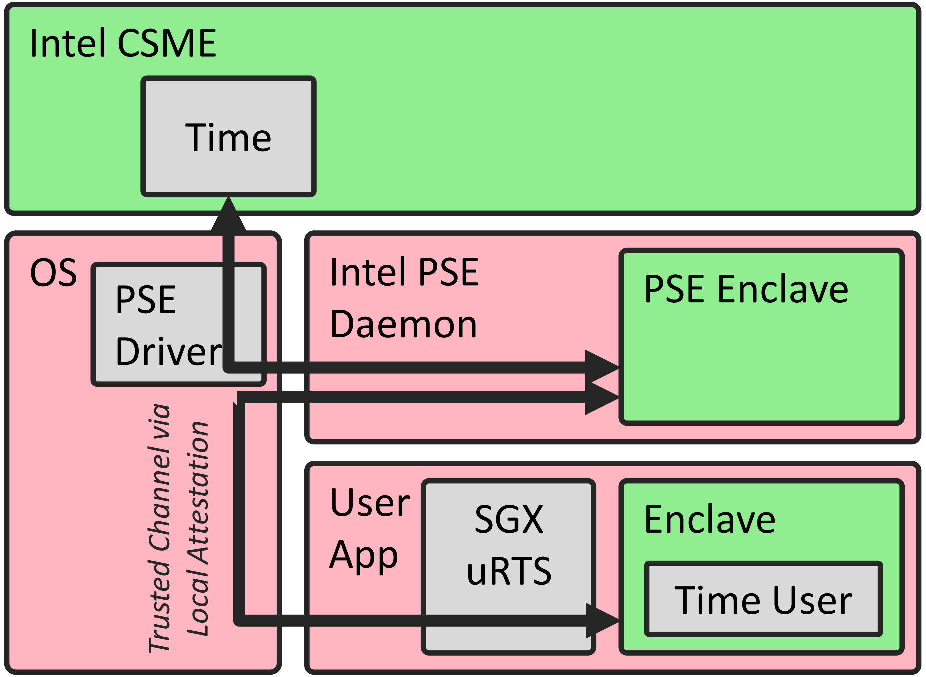
- **Two notion of time:**
  - Time point: exact moment of an event in time relative to an epoch
  - Duration: elapsed time between two events
- **Time is used as various security properties**
  - Information freshness
  - Non-repudiation of an event
  - Time-based access control
  - Automatic event control
- **Therefore, it is important that time value is sourced from a trustworthy source**

Background and Current Situation (1)

- **Current Intel SGX architecture does not provide a secure and trustworthy time information**
  - x86 system does not provide a stable real-time clock
- **What about TSC (Timestamp Counter)?**
  - TSC depends on the processor core clock, **differs in each CPU**, system software must adapt accordingly
  - Intel SGX (at least prior to SGX 2) **restrict access** to read TSC register
  - Although SGX 2 allows RDTSC instruction, TSC register is **suspect to untrusted write from the OS** (through writing TSC Offset register)
  - Hence, from the SGX security design perspective, **TSC is untrusted**

Background and Current Situation (2)

- **So, Intel SGX enclave must source trusted time information from external sources**
  - First era of SGX uses Platform Service Enclave (PSE) which communicates with Intel Management Engine (ME) to provide trusted time and trusted monotonic counter
  - Exposes the API: `sgx_get_trusted_time` in the SGX SDK



Background and Current Situation (3)

- **Since 2020, the `sgx_get_trusted_time` API is removed from the Linux SGX SDK**
  - No official statement for the reason of its removal
  - Some discussions noted about the removal of Intel ME driver in Linux due to licensing
  - May also due to SGX being segmented towards server machine where Intel ME is not used (Intel SPS is used instead)

Related Works

IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 15, 2020 1589

Establishing Trusted I/O Paths for SGX Client Systems With Aurora

Hongliang Liang, *Member, IEEE*, Mingyu Li, Yixiu Chen, Lin Jiang, Zhuosi Xie, and Tianqi Yang

Bring the Missing Jigsaw Back: TrustedClock for SGX Enclaves

Hongliang Liang
Beijing University of Posts and Telecommunications
Beijing, China
hliang@bupt.edu.cn

Mingyu Li
Beijing University of Posts and Telecommunications
Beijing, China
maxul@bupt.edu.cn

ABSTRACT

Intel SGX provisions shielded executions for security-sensitive computation, but has to rely on untrusted system services, such as clock, network and filesystem. This makes enclaves vulnerable to Iago

Although Intel offers trusted time service for enclaves, its time value is course-grained (second-resolution) and not absolute [11]. For enclave programs that request timestamp information tens or hundreds of times per second, this time service is far from be-

Intrusive and Impractical for Compatibility: Using hardware modification which requires deploying a custom firmware to build the trusted IO path

Applications and Challenges in Securing Time

Fatima M. Anwar
UCLA

Mani Srivastava
UCLA

Abstract

In this paper, we establish the importance of trusted time for the safe and correct operation of various applications. There are, however, challenges in securing time against hardware timer manipulation,

A malicious host may degrade system performance and user experience by increasing user perceived delays [17].

A time stack provides timekeeping and timestamping capabilities to all systems, and consists of three major components: hardware timers that count oscillations

2019 IEEE Real-Time Systems Symposium (RTSS)

Securing Time in Untrusted Operating Systems with TimeSeal

Fatima M. Anwar, *UMass Amherst*, fanwar@umass.edu
Luis Garcia, *UCLA*, garcialuis@ucla.edu
Xi Han, *UCLA*, xihan94@ucla.edu
Mani Srivastava, *UCLA*, mbs@ucla.edu

for securing applications in shielded execution environments such as Haven [8], SCONE [9], Panoply [10], and Graphene-SGX [11] have no access to secure time. Instead, they rely on untrusted operating system (OS) time. A compromised OS may lie about the time or signal early timeouts, and although traditional cryptographic techniques, trusted execution technologies, and network

VI. IMPLEMENTATION AND EVALUATION

We provide a scalable TIMESEAL implementation on a SGX enabled computer with an i7-6700K processor and 16 GiB memory running Ubuntu Linux 16.04.3, kernel version 4.10.32. The *sgx_get_trusted_time* API provides time from the hardware management engine. An application that wishes to acquire secure time instantiates TIMESEAL within its own process to limit OS

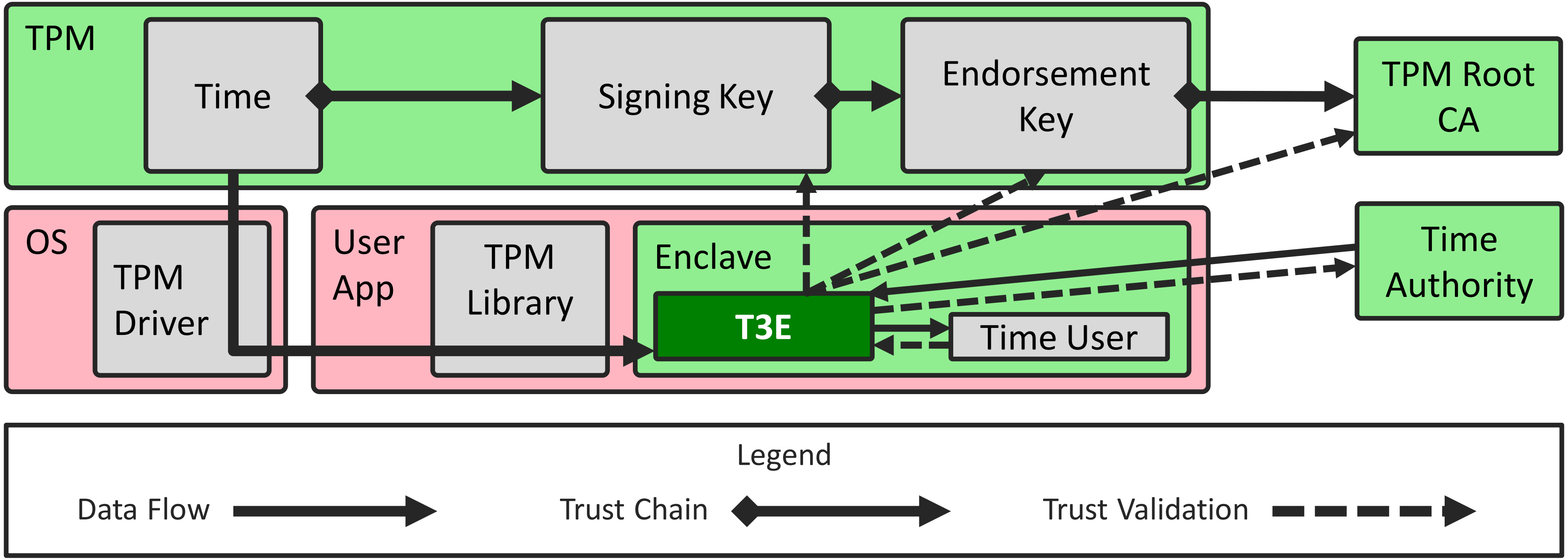
Obsolete

KU LEUVEN DistriNet

TPM at a Glance

- **A TEE device that enables establishing trust**
  - Performs essential cryptographic operations, particularly that involves asymmetric keys
  - TPM stores or derives a private key that is not leaked to outside system including its host system
- **TPM 2.0 also provides a monotonic clock**
  - Typically for timestamping purpose
- **Form-factor:**
  - **Hardware TPM** – dedicated chip
  - **Firmware TPM (fTPM)** – part of firmware of the host system hardware
  - **Virtual TPM** – Emulated by host system

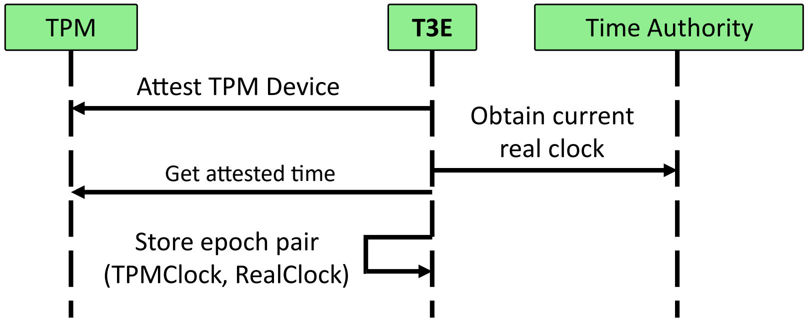
Approach



TPM
Time
Signing Key
Endorsement Key
TPM Root CA
OS
TPM Driver
User App
TPM Library
Enclave
T3E
Time User
Time Authority
Legend
Data Flow
Trust Chain
Trust Validation

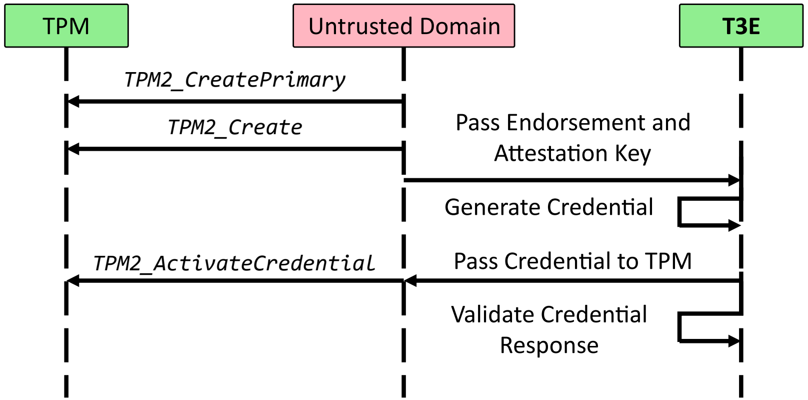
Provisioning the T3E

- **Provisioning procedure**



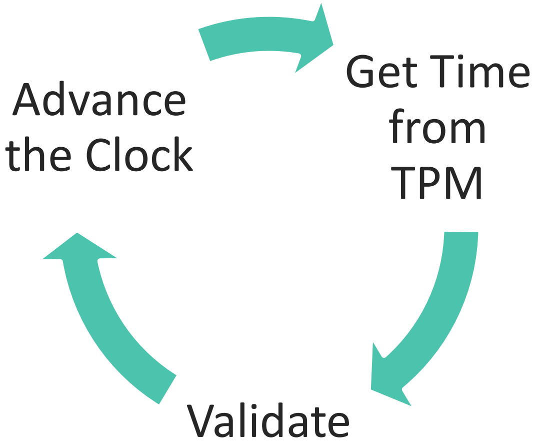
  - Initialize T3E with the trusted time information obtained from external trusted Time Authority and TPM to determine the **time epoch**

- **Attestation procedure**



  - Establish trust chain between T3E and TPM that passes through untrusted domain

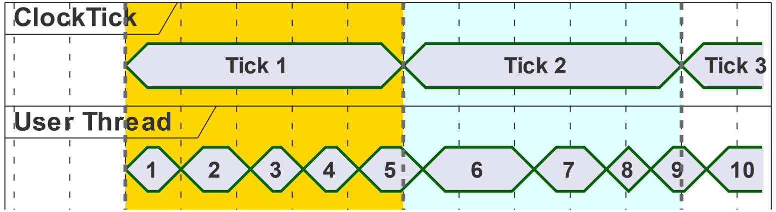
Advancing the Clock



- **T3E periodically requests and obtain time from TPM to advance its internal clock**
  - TPM acts as the tick source for T3E clock
  - The real wall clock is calculated by the offset between TPM time and the real clock time stored in the provisioning steps
- **T3E validates the time report from TPM via the established trust gained via the attestation**

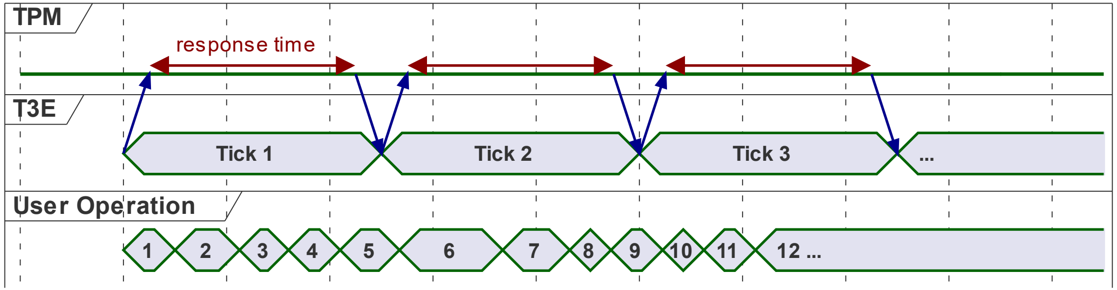
Challenges

- **TPM tick is not immediate**
  - TPM has processing delay between each tick request
- **T3E is not completely immune to adversarial delay**
  - Because the tick is still sourced through untrusted channel
  - The enclave cannot reliably determine if the execution has been delayed
  - Therefore T3E needs to have additional security properties
- **T3E strategy is to impose "Maximum Use-Count" to alleviate possible delay**
  - Within a single tick period, there may only be a limited *n* number of operations that requests a time

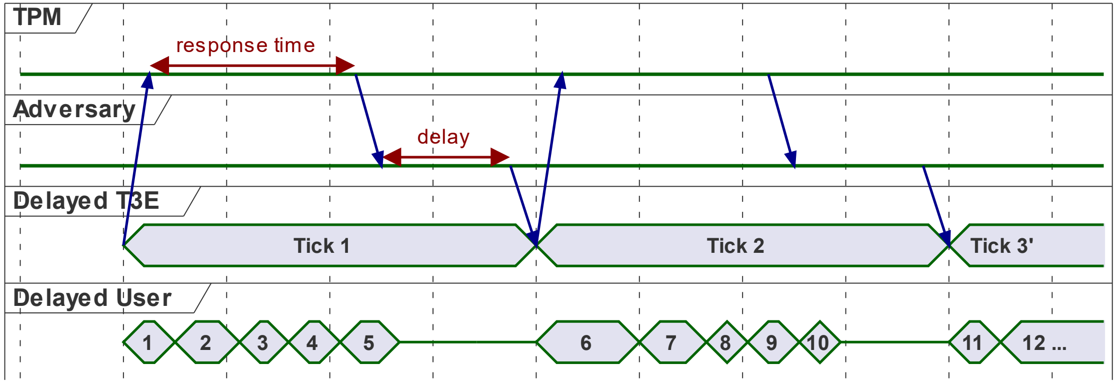


Impact of Use-Count on Timing

- **Regular Timing**
  - A single tick period may allow limited set of operations



- **Delayed Timing**
  - Delay may caused the subsequent operation to be postponed until tick has been received
  - The delay will be “accumulated” in the subsequent tick



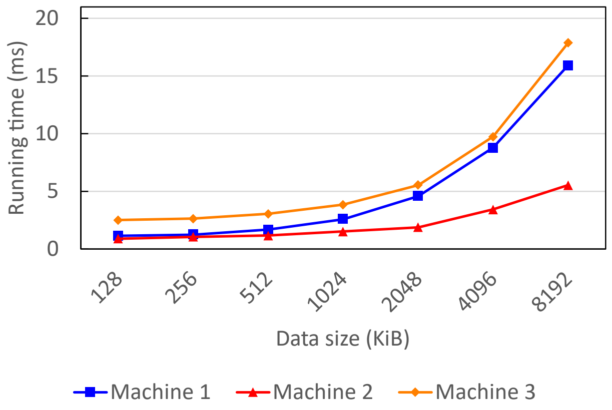
Security Evaluation

R1: Authentic time source	• T3E sources the epoch from authentic source (TPM and external/supervised trusted time source) • T3E advances the clock using authentic source (Attested TPM)
R2: Time cannot be replayed	• TPM clock is monotonically increasing • T3E enforces its internal clock to be monotonically increasing and protected against external adversary • T3E enforces nonces for the communication to the TPM
R3: Time cannot be sped up	• T3E only advances its clock using the tick from TPM • T3E may sense unusual tick duration (e.g., elapsed tick is slower than usual) and force for resynchronization
R4: Time cannot be paused or slowed down	• T3E prevents paused time tick by enforcing use-count • T3E may sense multiple use-count expiration to be a signal of attack and force for resynchronization

Performance Analysis

- **We analysed tick duration in three different machines with TPM**
  - 1 dedicated TPM, 2 fTPM
  - 3 signature schemes
- **We also measured a realistic use-case of trusted time to calculate its use-count**
  - Time-stamping Authority (TSA)

	RSASSA-PCKS1			RSASSA-PSS			ECDSA		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
Machine 1	396	391	401	402	398	415	280	278	297
Machine 2	83	82	92	85	84	95	30	29	37
Machine 3	231	227	291	231	229	254	138	136	190



use-count can then be computed using formula:

$$c_{max} = \frac{\bar{t}_{tpm\_interval}}{\bar{t}_{user\_time}}$$

KU LEUVEN DistriNet

Comparison Highlight

`sgx_get_trusted_time`	• **Deprecated** • No mitigation against delay attack
TimeSeal	• **Uses** `sgx_get_trusted_time` **(deprecated)** for baseline time keeping • Mitigating delay attack by using multiple counter thread (obfuscating the timer thread)
S-FaaS	• Uses Intel TSX to detect thread pause (**TSX itself is deprecated**) • Measure the duration, but cannot provide a time point because it is **unable to measure the pause duration** in the counter thread
TrustedClock	• Trusted channel via System Management Interrupt (SMI) handler • **Requires firmware modification** • **SMI throttles processor time** and not high-performing
T3E	• Uses TPM for time keeping • Mitigating delay attack by limiting the time usage, **assuming more operations to be done** in a single tick period

Conclusion

- **Providing trusted time in SGX enclave remains a challenge**
  - **Architectural changes is required** to enable an ideal trusted time information
  - RDTSC instruction (that is enabled) in SGX 2 **is not enough**
- **T3E allows the enclave to provide a practical means to obtain trusted time information in the absence of trusted time service in SGX**
  - **No hardware modification** required and not relying on deprecated APIs
  - **May be used in various high-performing use cases**, although **may not be ideal for low usage** due to the use-count upper bound limit
  - Further investigation to determine use-count dynamically depending on system load

Thank You

Gilang Mentari Hamidy

gilang.hamidy@kuleuven.be

KU LEUVEN DistriNet