A Sample *ACM* SIG Proceedings Paper in LaTeX Format[*]

Ke Zhang
3030058805
Department of Computer Science
University of Hong Kong
Pokfulam Road, Hong Kong
kzhang2@cs.hku.hk

Tianxiang Shen
3030058776
Department of Computer Science
University of Hong Kong
Pokfulam Road, Hong Kong
txshen2@cs.hku.hk

ABSTRACT
This paper provides a sample of a LaTeX document which conforms to the formatting guidelines for ACM SIG Proceedings. It complements the document *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2$_\epsilon$ and BibTeX*. This source file has been written with the intention of being compiled under LaTeX2$_\epsilon$ and BibTeX.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgments, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file.

1. INTRODUCTION
This is introduction.

2. BACKGROUND
2.1 Secure Enclave
A secure enclave is a set of software and hardware features that together provide an isolated execution environment to enable a set of strong security guarantees for applications running inside the enclave. Enclave allows user-level as well as Operating System (OS) code to define private regions of memory, whose contents are protected and unable to be either read or saved by any process outside the enclave itself, including processes running at higher privilege levels [].

Primally, secure enclaves can provide confidentiality, integrity, and attestation. Confidentiality guarantees that an adversary outside of the enclave cannot inspect the state of execution inside the enclave, even if they compromise the operating system or correctness of the computation running inside the enclave even if the operating system has been compromised or a user attempts to subvert the execution of the program inside the enclave. Finally, hardware-based attestation provides an unforgeable proof that enables a remote party to verify what has run inside the enclave even if they don't have physical access to the machine. A secure enclave thus provides a powerful cornerstone for secure computing and development of secure systems in general.

Intuitively, secure enclave fundamentally ensures the correctness and isolation in executing given process. The confirmation of input data freshness is hard to achieve, especially when the enclave encounters crash or restart. There are several wildly used secure enclave services [], one of the most popular security architectures is Intel Software Guard Extensions (SGX) []. However, a mature secure enclave designation as SGX still shows unsatisfied performance towards rollback attacks. In this report, we focus on the SGX architecture and its existing promotions in proposing protection against rollback attacks.

2.2 Rollback Attack
Rollback attacks remain a potential secure problem in secure enclave. In a rollback attack, attackers replace the latest data with an older version without being identified by the system.

Data integrity violation through rollback attacks can have severe implications. Consider, for example, a financial application implemented as an enclave. The enclave repeatedly processes incoming transactions at high speed and maintains an account balance for each user or a history of all transactions in the system. If the adversary manages to revert the enclave to its previous state, the maintained account balance or the queried transaction history does not match the executed transactions.

In reality, enclaves cannot easily detect this replay, because the processor is unable to maintain persistent state across enclave executions that may include reboots or crash. Another way to carry out rollback attacks in secure enclaves is to create multiple instances of a same process and route update requests to one instance and read requests to the other. Due to the characteristic of secure enclave, the instances are indistinguishable to remote clients or OS.

[*](Produces the permission block, copyright information and page numbering). For use with ACM_PROC_ARTICLE-SP.CLS V2.6SP. Supported by ACM.

CSIS 8101 Surveys 2012, CS Dept, HKU

To avoid rollback attacks, most commonly considered direction is to record the time related information for every state change. In this paper, we mainly discuss three designations in rollback attacks protection built on the SGX architecture.

3. PROBLEM

In this section, we introduce the SGX architecture with three main operations directly related to our rollback protection designations. Then we describe the adversary model targets in rollback attacks by detailing their assumptions and capabilities.

3.1 SGX Architecture

In a standard SGX as specified in [], apart from the confidentiality and integrity nature of SGX, there are fundamentally three operations we concern in this report, *i.e.*, the enclave creation, the sealing, and the attestation.

- **Enclave creation.** An enclave is created by the user client. In enclave creation, the client specifies the code to be processed in SGX. Security mechanisms in the processors create a data structure called SGX Enclave Control Structure (SECS) that is stored in a protected memory area. Enclaves' code created by the client cannot contain sensitive data. The start of the enclave is recorded by the processor, reflecting the content of the enclave code as well as the loading a sequence of instructions. The recording of an enclave start is called measurement and it can be used for later attestation. Once an enclave is no longer needed, the OS can terminate it and thus erase its memory structure from the protected memory.
- **Sealing** Enclaves can save confidential data across executions. Sealing is the process to encrypt and authenticate enclave data for persistent storage []. All local persistent storage (*e.g.* disk) is controlled by the untrusted OS. For each enclave, the SGX architecture provides a sealing key that is private to the executing platform and the enclave. The sealing key is derived from a Fuse Key (unique to the platform, not known to Intel) and an Identity Key that can be either the Enclave Identity or Signing Identity. The Enclave Identity is a cryptographic hash of the enclave measurement and uniquely identifies the enclave. If data is sealed with Enclave Identity, it is only available to this particular enclave version. The Signing Identity is provided by an authority that signs the enclave prior to its distribution. Data sealed with Signing Identity can be shared among all enclave versions that have been signed with the same Signing Identity.
- **Attestation** Attestation is the process of verifying that certain enclave code has been properly initialized. In local attestation a prover enclave can request a statement that contains measurements of its initialization sequence, enclave code and the issuer key. Another enclave on the same platform can verify this statement using a shared key created by the processor. In remote attestation the verifier may reside on another platform. A system service called Quoting Enclave signs the local attestation statement for remote verification. The verifier checks the attestation signature with the help of an online attestation service that is run by Intel. Each verifier must obtain a key from Intel to authenticate to the attestation service. The signing key used by the Quoting Enclave is based on a group signature scheme called EPID (Enhanced Privacy ID) which supports two modes of attestation: fully anonymous and linkable attestation using pseudonyms []. The pseudonyms remain invariant across reboot cycles (for the same verifier). Once an enclave has been attested, the verifier can establish a secure channel to it using an authenticated key exchange mechanism.

In this report, protocols described in Section 4 primally utilize these three operations in SGX to provide rollback attack protections.

3.2 Adversary Model

In this report, we consider rollback attacks [], where the adversary may carry out their attacks in two ways. One is replay, *i.e.*, arbitrarily shut down the system, and replay from a stale state. The other direction is forking attacks [], where the adversary attempt to fork the storage system, *e.g.*, by running multiple replicas of the storage system.

The goal of methods specified in Section 4 are to guarantee the data integrity, confidentiality, and freshness towards rollback attacks based on SGX architecture. Note that for different methods, different level of adversary's strength is considered, which is listed and compared in Section 4.

4. SOLUTIONS

In this section, we mainly introduce three state-of-art solutions in solving the problem we mention in Section 3. We separately describe their motivation, inner designations, and respective improving directions in detail.

4.1 SGX Counter

Intel has recently added support for monotonic counters (MC) [] as an optional SGX feature. The Monotonic Counter can be utilized by enclave developers for rollback attack protection.

4.1.1 Overview

SGX supports creating a limited number of MCs for each enclave. Monotonic counters are shared among enclaves that have the same code. An enclave can query availability of counters from the Platform Service Enclave (PSE). If supported, the enclave can create up to 256 counters. The default owner policy encompasses that only enclaves with the same signing key may access the counter. Counter creation operation returns an identifier that is a combination of the Counter ID and a nonce to distinguish counters created by different entities. On creating a MC, it gets written to the non-volatile memory in the platform. The enclave must store the counter identifier to access it later, as there is no API call to list existing counters. After a successful counter creation, an enclave can increment, read, and delete the counter. Because each enclave shares the same value of the monotonic counters, it guarantees the verification for data freshness. In other words, only when an enclave preserves the same counter value as the others in the platform, its reserving data are the latest. Also, when one enclave encounters crash or reboot, it can recover data with the help of monotonic counters shared in the platform.

According to the SGX API documentation [5], counter operations involve writing to a non-volatile memory. Repeated write operations can cause the memory to wear out, and thus the counter increment operations may be rate limited. Based on Intel developer forums [51], the counter service is provided by the Management Engine on the Platform Control Hub (PCH).

4.1.2 Limitations

Though being as a selective feature in SGX architecture, it has strict memory constraints and performs slow during experimental tests [].

The SGX Monotonic Counter updates take 80-250 ms and reads 60-140 ms. When an enclave needs to persistently store an updated state, it can increment a counter, include the counter value and identifier to the sealed data, and verify integrity of the stored data based on counter value at the time of unsealing. However, such approach may wear out the used non-volatile memory. Assuming a system that updates one of the enclaves on the same platform once every 250 ms, the non-volatile memory used to implement the counter wears out after approximately one million writes, making the counter functionality unusable after a couple of days of continuous use. Even with a modest update rate of one increment per minute, the counters are exhausted in two years. Thus, SGX counters are unsuitable for systems where state updates are frequent and continuous. Additionally, since the non-volatile memory used to store the counters resides outside the processor package, the mechanism is likely vulnerable to bus tapping and flash mirroring attacks [].

Note that SGX also provides the SGX trusted time feature for checking the timestamp of one stored data record. However, including a timestamp to each sealed data version only allows an enclave to distinguish which out of two seals is more recent, enclaves cannot identify if the sealed data provided by the OS is fresh and latest.

4.2 ROTE

To overcome the slowness of SGX Monotonic Counters and provide stable persistent rollback attack protections, ROTE [] is proposed as a distributed trusted counter service based on a consensus protocol.

4.2.1 Overview

4.2.2 Limitations

4.3 Speicher

5. RUNTIME

This is runtime.

6. EVALUATION PLAN

This is evaluation part.

7. CONCLUSION

This is conclusion.

8. ACKNOWLEDGMENTS

This section is optional.

9. REFERENCES