Vulnerabilities in **Trusted Execution Environments (TEEs)** require that TEE applications must be designed to **recover** from and **limit** the damage caused by a compromise.

# A Formal Foundation for Recoverability

Paul Crews
ptcrews@google.com
*Google*

## Introduction

Trusted Execution Environments (TEEs) such as Intel SGX aim to provide confidentiality and integrity for programs inside the environment. However, in practice vulnerabilities undermine these guarantees, and thus TEE applications must be designed to handle a compromise. Using trace properties, this poster proposes a new threat model and new definitions to formally model how to design an application that can recover from and mitigate such a compromise.

## Background

Most Intel SGX threat models assume that both SGX and the enclave application are secure. However, recent vulnerabilities in SGX (Table 1) and the possibility of bugs in the application itself undermine this assumption. As a result, only applications that can handle such a compromise are suitable for running in a TEE.

| Attack Category | Security Property Compromised | | | SGX Keys Compromised | | Examples |
|---|---|---|---|---|---|---|
| | Confidentiality | Integrity | Attestation | EPID Key | CPU Fuse Key | |
| User Enclave Read | ● | | | | | Page Fault Attacks [3, 4] |
| User Enclave Code Execution | ● | ● | ● | | | Plundervolt [2] |
| Enclave Read | ● | | ● | ● | | Foreshadow [1] SGAxe [5] |
| Enclave Code Execution | ● | ● | ● | ● | | Plundervolt [2] |
| Microcode Code Execution | ● | ● | ● | ● | ● | ? |

**Table 1.** Types of attacks against Intel SGX, and what security properties each attack type compromises.

1. Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In *27th USENIX Security Symposium (USENIX Security 18)*.
2. Kit Murdock, David Oswald, Flavio D Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. 2020. Plundervolt: Software-based fault injection attacks against Intel SGX. In *2020 IEEE S&P*.
3. Shweta Shinde, Zheng Leong Chua, Viswesh Narayanan, and Prateek Saxena. 2016. Preventing page faults from telling your secrets. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*.
4. Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. 2017. Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In *26th USENIX Security Symposium (USENIX Security 17)*.
5. Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom. 2020. SGAxe: How SGX fails in practice.

## Definitions

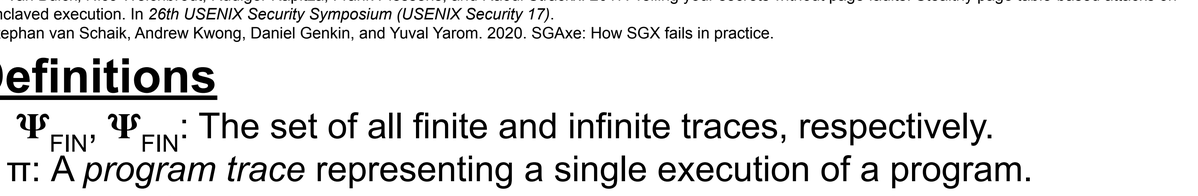
- $\Psi_{FIN}$, $\Psi_{FIN}$: The set of all finite and infinite traces, respectively.
- π: A *program trace* representing a single execution of a program.
- Π: A set of traces representing all possible executions of a program.
- P ∈ Prop: A *trace property* is defined by a set of traces. A program satisfies P if Π ⊆ P. Prop is the set of all trace properties.
- **P** ∈ **HP**: A *hyperproperty* is defined by a set of sets of traces, describing properties between program traces. **HP** is the set of all hyperproperties.

## Threat Model (One-Shot Adversary)

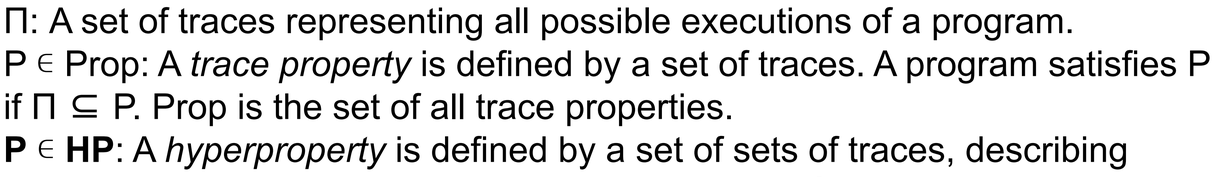
A *one-shot adversary* can exploit a TEE application precisely once. We model this by arbitrarily modifying a program trace π after the time of compromise. The set of all possible compromises for a trace π is defined as MUTATE(π):

$$\texttt{MUTATE}(\pi) = \{(m, i) | i \geq 0, m = [\pi_0, \pi_1, \ldots, \pi_{i-1}, m_i, m_{i+1}, \ldots]\}$$

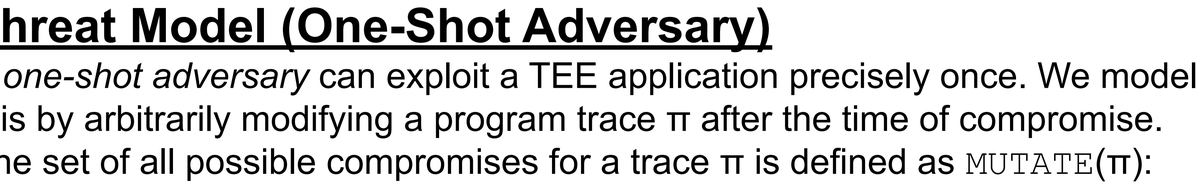
For simplicity, we also assume that the attacker does not compromise the TEE's ability to recover and perform secure remote attestation. In the case of SGX, this means the attacker does not compromise the CPU Fuse Keys.




(a): Recoverability




(b): Bounded Lookback




(c): Recoverability and Bounded Lookback

## Recoverability

Recoverability is a property of trace and hyperproperties. A property is recoverable if there exists a finite sequence of actions r such that if they occur after the time of compromise, the property still holds for the trace(s).

$$\texttt{RECV} \equiv \{\Pi \in \texttt{Prop} | \exists r \in \Psi_{FIN}, \forall \pi \in \Pi, (m, i) \in \texttt{MUTATE}(\pi) : (r \subset_E m[i+1:] \Rightarrow m \in \Pi)\}$$

$$\mathbf{RECV} \equiv \{\mathbf{\Pi} \in \mathbf{HP} | (\forall \Pi \in \mathbf{\Pi} : (\exists r \in \Psi_{FIN}, \forall \pi \in \Pi, (m, i) \in \texttt{MUTATE}(\pi) : (r \subset_E m[i+1:] \Rightarrow \Pi \setminus \pi \cup \{m\} \in \mathbf{\Pi})))\}$$

## Bounded Lookback

Bounded lookback is a property of trace and hyperproperties. A property has bounded lookback if there exists a finite sequence of actions b such that if they occur before the time of compromise, the property still holds for the trace(s).

$$\texttt{BL} \equiv \{\Pi \in \texttt{Prop} | \exists b \in \Psi_{FIN}, \forall \pi \in \Pi, (m, i) \in \texttt{MUTATE}(\pi) : (b \subset_E m[:i] \Rightarrow m \in \Pi)\}$$

$$\mathbf{BL} \equiv \{\mathbf{\Pi} \in \mathbf{HP} | (\forall \Pi \in \mathbf{\Pi} : (\exists b \in \Psi_{FIN}, \forall \pi \in \Pi, (m, i) \in \texttt{MUTATE}(\pi) : (b \subset_E m[:i] \Rightarrow \Pi \setminus \pi \cup \{m\} \in \mathbf{\Pi})))\}$$

## Observational Determinism

Observational Determinism (OD) is a hyperproperty that ensures that for any two traces with the same low-equivalent starting state, the entire trace is low-equivalent[6].

$$\mathbf{OD} \equiv \{\Pi \in \texttt{Prop} | (\forall \pi, \pi' \in \Pi : \pi[0] =_L \pi'[0] \Rightarrow \pi \approx_L \pi')\}$$

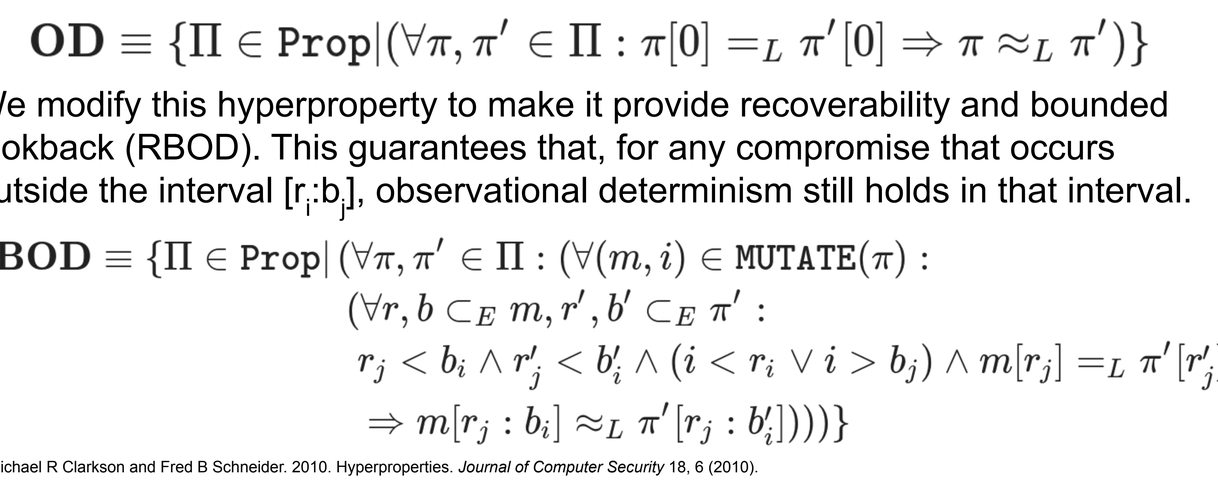
We modify this hyperproperty to make it provide recoverability and bounded lookback (RBOD). This guarantees that, for any compromise that occurs outside the interval $[r_i:b_j]$, observational determinism still holds in that interval.

$$\mathbf{RBOD} \equiv \{\Pi \in \texttt{Prop} | (\forall \pi, \pi' \in \Pi : (\forall (m, i) \in \texttt{MUTATE}(\pi) : (\forall r, b \subset_E m, r', b' \subset_E \pi' : r_j < b_i \wedge r'_j < b'_i \wedge (i < r_i \vee i > b_j) \wedge m[r_j] =_L \pi'[r'_j] \Rightarrow m[r_j : b_i] \approx_L \pi'[r_j : b'_i])))\}$$

6. Michael R Clarkson and Fred B Schneider. 2010. Hyperproperties. *Journal of Computer Security* 18, 6 (2010).

## Example SGX Application: EnclaveDB

EnclaveDB is a DBMS that runs inside an SGX enclave, providing confidentiality and integrity for stored data[7]. Since EnclaveDB uses a single global database key, it is not recoverable and does not provide bounded lookback. We implemented a small prototype extension to EnclaveDB that implements key rotation, providing bounded lookback and recoverability. The graph below shows the decryption latency overhead as a function of time and rotation frequency.




7. Christian Priebe, Kapil Vaswani, and Manuel Costa. 2018. EnclaveDB:A Secure Database Using SGX. In *2018 IEEE Symposium on Security and Privacy (SP)*.

## Future Work

- Formally model recoverability and bounded lookback.
- Implement an SGX library that provides recoverability and bounded lookback for cryptographic primitives.
- Prove that the SGX library is recoverable and has bounded lookback.
- Implement SGX applications that use this library and prove they are recoverable and have bounded lookback.