Chapter 3

Overview

As already indicated, the starting point of this thesis was **not** zero. The DPBoost paper ([31], 2020) provided the theory behind DP-GBDT learning. Moreover, Théo Giovanna built a first Python implementation [19] of said algorithm during the course of his master's thesis (submitted Feb. 2021). Starting from this point, the high level objectives were in short: Porting the code to C++, putting it into an enclave, harden it against side-channels, come up with a realistic strategy for deployment in practice.

To acquire an understanding of the problem we will first give a high level overview of the desired solution for the insurance use case. Subsequently, the adversary model is presented. At last, we outline the goals and requirements our system should meet.

3.1 Solution overview

A visual representation of the setup is given in figure 3.1. Prior to actual operation, the insurance has deployed the DP-GBDT enclave on its servers. The enclave must be reachable by external insurance customers. From there on, the insurance starts selling their cyber insurance policies to companies. In return, customers send back filled out questionnaires about their cyber security policies. They can use a simple client application for this. The client application conducts the survey and sends the encrypted answers directly to the enclave. The questionnaires arrive at the enclave one at a time. Upon reception the enclave saves the acquired data to disk. This continues until enough samples are collected to get a meaningful result from running the DP-GBDT algorithm. At this point the insurance notifies the enclave to initiate the first training. Upon completion the enclave returns the output model to the insurance. The enclave continues collecting samples from customers. And again, once enough samples are collected the insurance can instruct the enclave to train on the newly received data. The corresponding output can be used to improve the previous model. Further, if at any point some part of the model is lost, the insurance can instruct the enclave to recreate the missing part of the model.

3.2 Threat model

Next, we discuss the threat model, which is relevant for side-channel hardening and secure deployment of the system (chapters 4 and 6). We use **two different threat models** to cover the two main security goals. To ensure customer data privacy, which is our main priority, a very powerful adversary was chosen. To address certain types of denial of service attacks by

1. attest
2. questionnaire
3. store
4. train model
5. output...

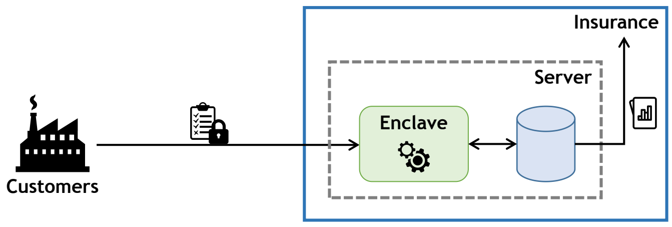

Figure 3.1: Insurance use case overview

external customers, we use a weaker adversary model. It is clear that denial of service attacks can occur at virtually any point of an enclave setup. Most of them (e.g. network flooding) are out of scope. However, some cases would inevitably need to be handled in practice. In particular, a malicious external customer must not be able to disrupt the whole setup by crafting and submitting malicious messages to the enclave.

TM1, Goal: Customers want privacy even if the insurance is malicious TM1 represents a malicious service provider. This adversary has full control over both the software (including OS or hypervisor) and the hardware of his system. Thus he is able to start or stop processes at any point, set their affinity and modify them at runtime. He can also read and write to any memory region except the enclave memory, map any virtual page to any physical one and dynamically re-map pages. He can run multiple instances of an enclave in parallel Moreover he has various network control capabilities: He may tamper with, delete, delay, reorder, or replay any network packets exchanged between the server and the participating parties. We also expect him to closely monitor the program execution for side-channels. Lastly, he has full access to state-of-the-art enclave attack frameworks like SGX-Step [41]. The goal of the malicious service provider is to gather as much information as possible about customers participating in the training process. Note that, from a customers perspective, service provider and insurance might very well be the same entity. Or similarly, the two might collude to steal secrets from the insurance customers.

TM2, Goal: Insurance wants good results even if customers are malicious TM2 assumes existence of a legitimate external customer with malicious intentions. We assume that such a customer does not have access to the servers where the enclave resides. Put differently, The enclave is located in a secure area at the insurance company. However, the malicious customer does have full control over his questionnaire data that is sent to the enclave. Further capabilities include: initiating multiple session with the DP-GBDT enclave or sending his questionnaire multiple times (but not to the extend where the sheer number of messages leads to denial of service). The objective of a malicious customer is to decreases the quality of the overall result, or even wipe out previous progress of the learning process entirely.

Out of scope The following attack vectors are considered out of scope for both TM1 and TM2: We assume that the underlying hardware of the enclave setup is not compromised. Further, all known SGX vulnerabilities with a fix available are patched. The TEE hardware must work as expected and provide an isolated execution environment with remote attestation capabilities for any code running inside. Enclave-provisioned secrets are not accessible from untrusted code, etc. As previously indicated, traditional denial of service attacks are also out of scope. Thus, concerns like the network connection being cut, or the OS not scheduling our code are ignored.

3.3 Goals

Efficient algorithm The new C++ implementation has to meet multiple requirements. First of all and contrary to it's Python predecessor it should be free of bugs. From this point, it should still be able to produce usable results for reasonably small privacy budgets. In other words, the noise added through differential privacy should not completely erase the output model's expressive power. Additionally, the algorithm runtime should be reasonable. Even though training will likely not be performed very frequently.

Side-channel hardening As SGX enclaves are vulnerable to side-channels, the implementation must be adequately hardened. But the hardening process differs from traditional hardening approaches to some extent: First of all, the whole algorithm is quite large and complex. Usually hardening is applied in a smaller frame, for instance to an AES encryption function in a cryptographic library. Second, we are not trying to eliminate every single bit of leakage. We eliminate just enough to achieve ε-differential privacy. For example, in our algorithm the adversary is allowed to learn the final model. For information that must be kept secret in turn, we eliminate all leakage through *digital side-channels*. As already introduced earlier (ref TODO), this sums up all side-channels that carry information over discrete bits. Examples are address traces, cache usage, and data size. In terms of secret dependant memory/cache accesses we decided against specifying a fixed acceptable leakage granularity (e.g. cache line). Admittedly, such an assumption would make the hardening process easier and execution times faster. However, the result would be prone to be broken again once the next better attack vector is discovered. To put it differently and to give an example: In order to retrieve a value at a secret dependent index from an array, every single element should be touched.

Secure deployment Even the most highly performing and privacy-preserving algorithm is useless if it can't be applied to a real-world system. Apart from side-channels, there are several other attack vectors that must be considered for secure deployment. Enclave state rollback or fork attacks, for instance. Additionally, we should be able to deal with potential technical failures due to either human negligence or machine errors. Our goal is therefore to highlight the key problems and present viable new or existing solutions. This includes:

- Data collection over an extended time period
- Training on newly acquired customer data to improve the existing model output[1]
- Training on old data after output model loss[1]
- Enclave replication and migration for maintaining persistent state

[1]This is not trivial whilst not violating differential privacy