# Master Thesis
## Secure Two-Party Computation: ABY versus Intel SGX

TECHNISCHE UNIVERSITÄT DARMSTADT

The *Engineering Cryptographic Protocols Group* (ENCRYPTO) is offering a master thesis that will be supervised by Christian Weinert (christian.weinert@crisp-da.de) and Prof. Dr.-Ing. Thomas Schneider.

### Motivation

Secure two-party computation allows two parties to jointly compute a publicly known function without disclosing their respective inputs to each other. This can be realized using well-known cryptographic protocols such as the ones provided by Yao (Yao's Garbled Circuits) and Goldreich, Micali, and Wigderson (GMW). These protocols both obliviously evaluate a virtual *boolean circuit* that represents the desired functionality. In case the functionality consists only of additions and multiplications, there is also the possibility to use a protocol that obliviously evaluates a virtual *arithmetic circuit*.

The ABY framework (`https://github.com/encryptogroup/ABY`) provides state-of-the-art implementations of all these protocols. It furthermore allows to select different protocols for different parts of the functionality in order to optimize the overall performance. The circuits representing the functionality can either be defined manually within the framework or are automatically generated from a high-level language with the help of a compiler like CBMC-GC (`http://forsyte.at/software/cbmc-gc/`). Nevertheless, the computational and communication overhead incurred by secure two-party computation is at least an order of magnitude higher compared to unprotected computation. Thus, while practicality in several application scenarios can already be claimed, the situation is far from satisfying.

Intel Software Guard Extensions (SGX) seems to be a promising alternative. This technology provides a trusted execution environment within processors s.t. code can be executed securely on a remote machine and is isolated from all other processes running on the same machine. Thereby it is possible for two parties to submit their inputs and (attested) code to an SGX-enabled machine and retrieve the computation result without significant overhead and without disclosing their inputs. What is especially appealing about SGX is its ubiquitous availability as it comes for free in all recent Intel CPUs.

Unfortunately, researchers have found many side-channel attacks that can compromise the security of SGX protected data. For example, a malicious operating system could monitor memory page accesses to infer information about the control flow, which in turn often depends on the data that is supposed to be protected.

### Goal

Boolean and arithmetic circuits as used in cryptographic secure two-party computation protocols are inherently secure against a wide range of side-channel attacks. This is because they always execute all possible paths in a program and thus do not reveal secrets in case the control flow is observed by some malicious entity.

Therefore, the main goal of this thesis is to enable the execution of large virtual boolean and arithmetic circuits (in the same format that can be processed by ABY) within SGX environments. The scalability and the performance of the resulting solution should be compared to ABY and to the execution of unprotected code within SGX.

A special focus of the performance evaluation lies on so-called *universal circuits*. This kind of circuit can be configured by input data to emulate arbitrary circuits, thereby enabling so-called *private function evaluation* where not only the inputs but also the computed function should be kept private. Possible application scenarios include smart buildings (energy suppliers calculate individual electricity rates) and smart cars (insurance companies determine custom rates).

Furthermore, the security of the solution should be assessed (with respect to different side-channel attacks) and the performance should be compared to various existing side-channel protection mechanisms.

The final aspect of this thesis deals with circuit optimization. For cryptographic secure two-party protocols it is important to optimize circuits with respect to either the total number of AND / multiplication gates or the maximum depth of paths containing AND / multiplication gates, depending on the protocol chosen for execution. The question is which kind of optimization strategy yields the most efficient execution within the SGX environment and whether there is a significant difference measurable between optimized and unoptimized circuits.

### Requirements

- Good programming skills in C/C++ (or Rust)
- At least basic knowledge of cryptography
- High motivation + ability to work independently