TECHNISCHE
UNIVERSITÄT
DARMSTADT

Thesis Type

This is the Title

Student Name
March 16, 2022

ENCRYPTO
CRYPTOGRAPHY AND
PRIVACY ENGINEERING

Cryptography and Privacy Engineering Group
Department of Computer Science
Technische Universität Darmstadt

Supervisors: M.Sc. Helen Möllering
M.Sc. Oleksandr Tkachenko
Prof. Dr.-Ing. Thomas Schneider

Erklärung zur Abschlussarbeit gemäß §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Student Name, die vorliegende Thesis Type ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Thesis Statement pursuant to §23 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Student Name, have written the submitted Thesis Type independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.

In the submitted thesis the written copies and the electronic version for archiving are identical in content.

Darmstadt, March 16, 2022

Student Name

Abstract

Nowadays, the world has turned into an information-driven society where the distribution and processing of information is the most significant economic activity. Nonetheless, this global trend also brings a severe privacy risk because the increased amount of interconnected devices and services may reveal users' sensitive information to untrusted third-party service providers. Secure multi-party computation (MPC) was introduced in the 1980s and enabled secure computations between two or more parties, such that nothing beyond what can be inferred from the output is revealed. However, it can be possible that an adversary is able to determine if a particular data record was used in the computation of a concrete computation result. Such a so-called membership inference attack raises privacy concerns. In 2006, the concept of differential privacy (DP) was introduced, which guarantees the result of a group to be similar independent of whether an individual is in the queried database or not. One research direction for privacy protection is to combine both MPC and DP.

Although works that combine MPC and DP exist, they ignore the theoretical assumption of DP in the practical implementation. Specifically, DP assumes precise noise sampling and computation under real numbers. The major obstacle is guaranteeing DP and sample noise efficiently under MPC with finite precision.

The main goal of this thesis is to design new techniques that combine MPC and DP to guarantee privacy under floating-point arithmetic. We convert existing secure differentially private mechanisms for floating-point numbers and integer sampling methods into MPC protocols. In addition, we implement our MPC protocols and evaluate the computation and communication cost of different optimization techniques.

Acknowledgments

If you like, you can add acknowledgments here.

Contents

1 Fixed-Point and Floating-Point Operation **1**

1.1 Correctness of $\Pi^{Wrap}\left(\langle x\rangle_0^A, \ldots, \langle x\rangle_{(N-1)}^A\right)$ 2

List of Figures **5**

List of Tables **6**

List of Abbreviations **7**

Bibliography **8**

1 Fixed-Point and Floating-Point Operation

Let $LT(a, b)$ be defined as follows:

$$LT(a,b) = \begin{cases} 1 \text{ if } a < b \\ 0 \text{ otherwise} \end{cases} \tag{1.1}$$

Let $[x]_{2^\ell}$ denotes $x \mod 2^\ell$ and omit the subscript 2^ℓ when not affects the understanding.

For $a, b \in \mathbb{Z}_{2^\ell}$ (following arithmetic share is under $\mathbb{Z}_{2^\ell}$), we have $[a] = a$, $[b] = b$ and [MRVW21]:

$$\begin{aligned} &[a+b] = [a] + [b] - 2^\ell \cdot LT([a+b], [b]) = [a] + [b] - 2^\ell \cdot LT([a+b], [a]) \\ &[a-b] = [a] - [b] + 2^\ell \cdot LT([a], [b]) \end{aligned} \tag{1.2}$$

To extend SecFloat [RBS+22] to multiparty settings, we design following building blocks:

Protocol: $\Pi^{Wrap}\left(\langle x\rangle_0^A,\ldots,\langle x\rangle_{(N-1)}^A\right)$

Input: Secret shared value x**, such that parties hold** $\langle x\rangle_0^A,\ldots,\langle x\rangle_{(N-1)}^A$**, where** $x=\sum_{i=0}^{N-1}\langle x\rangle_i^A \mod 2^\ell$

Output: Compute the arithmetic secret share $\langle t\rangle^A$**, such that** $\sum_{i=0}^{N-1}\langle x\rangle_i^A = x+t\cdot 2^\ell$**.**

```
1 : Parties generate edaBit and each party holds $\left(\langle r\rangle_i^A,\langle r_0\rangle_i^B,\ldots,\langle r_{\ell-1}\rangle_i^B\right)$, where $r=\sum_{j=0}^{\ell-1}2^j\cdot r_j$.
2 : Each party locally compute $t_i^{(1)}=LT\left(\left[\langle x\rangle_i^A+\langle r\rangle_i^A\right],\langle r\rangle_i^A\right)$ by taking $\left[\langle x\rangle_i^A+\langle r\rangle_i^A\right]$ and $\langle r\rangle_i^A$ as plaintext values.
3 : Each party locally compute $\left[\langle x\rangle_i^A+\langle r\rangle_i^A\right]$ and broadcast it to other parties.
4 : Each party locally compute public known $t^{(2)}$, where $\sum_{i=0}^{N-1}\left[\langle x\rangle_i^A+\langle r\rangle_i^A\right]=[x+r]+t^{(2)}\cdot 2^\ell$.
5 : Parties compute $\left\langle t^{(3)}\right\rangle^A$ using $\Pi_{BitAdder}$, where $\sum_{i=0}^{N-1}\langle r\rangle_i^A=r+t^{(3)}\cdot 2^\ell$.
6 : Parties compute $\left\langle t^{(4)}\right\rangle^A=\Pi_{LT}([x+r],r)$.
7 : Each party locally compute $\langle t\rangle_i^A=t^{(1)}+t_i^{(2)}-\left\langle t^{(3)}\right\rangle_i^A-\left\langle t^{(4)}\right\rangle_i^A$ as an arithmetic share of $t$.
```

Protocol 1.1: Protocol for wrap operation.

1.1 Correctness of $\Pi^{Wrap}\left(\langle x\rangle_0^A,\ldots,\langle x\rangle_{(N-1)}^A\right)$

For $i\in[0,N-1]$, suppose that each party i holds an arithmetic share $\langle x\rangle_i^A$ of a secret x (i.e., $x=\left[\sum_{i=0}^{N-1}\langle x\rangle_i^A\right]_{2^\ell}$ and $\langle x\rangle_i^A\in\mathbb{Z}_{2^\ell}$), and edaBit [EGK+20] $\langle r\rangle_i^A,\langle r_0\rangle_i^B,\ldots,\langle r_{\ell-1}\rangle_i^B$ (i.e., $r=\sum_{j=0}^{\ell-1}2^j\cdot r_j$ and $r\in\mathbb{Z}_{2^\ell}$), we want to compute an arithmetic share $\langle t\rangle^A$ of t, where

$$x+t\cdot 2^\ell=\langle x\rangle_0^A+\ldots+\langle x\rangle_{(N-1)}^A \tag{1.3}$$

In following, we show that $t=\sum_{i=0}^{N-1}t_i^{(1)}+t^{(2)}-t^{(3)}-t^{(4)}$ can be computed in four steps.

1. Compute $t_i^{(1)}=LT\left(\left[\langle x\rangle_i^A+\langle r\rangle_i^A\right],\langle r\rangle_i^A\right)$

First, as $[a+b]=[a]+[b]-2^\ell\cdot LT([a+b],[b])$ (cf. Eq. (1.2)), we replace a with $\langle x\rangle_i^A$ and b with $\langle r\rangle_i^A$ and get follows:

$$\left[\langle x\rangle_i^A+\langle r\rangle_i^A\right]=\langle x\rangle_i^A+\langle r\rangle_i^A-2^\ell\cdot LT\left(\left[\langle x\rangle_i^A+\langle r\rangle_i^A\right],\langle r\rangle_i^A\right) \tag{1.4}$$

which can be reformulated as follows:

$$\langle x\rangle_i^A = \left[\langle x\rangle_i^A + \langle r\rangle_i^A\right] - \langle r\rangle_i^A + 2^\ell \cdot LT\left(\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right], \langle r\rangle_i^A\right) \tag{1.5}$$

Note that $t_i^{(1)} = LT\left(\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right], \langle r\rangle_i^A\right)$ is computed by each party i locally, i.e., takes the values of $\langle x\rangle_i^A$, $\langle r\rangle_i^A$, calculate the modulo $\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right]$ and compare it with $\langle r\rangle$ in plaintext. Note that $t_i^{(1)}$ is an integer and only known to party i.

Then, we can rewrite Eq. (1.3) with Eq. (1.5) as follows:

$$\begin{aligned} x + t\cdot 2^\ell &= \langle x\rangle_0^A + \ldots + \langle x\rangle_{(N-1)}^A \\ &= \left[\langle x\rangle_0^A + \langle r\rangle_0^A\right] - \langle r\rangle_0^A + 2^\ell \cdot LT\left(\left[\langle x\rangle_0^A + \langle r\rangle_0^A\right], \langle r\rangle_0^A\right) + \ldots \\ &= \sum_{i=0}^{N-1}\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right] - \sum_{i=0}^{N-1}\langle r\rangle_i^A + 2^\ell \cdot \sum_{i=0}^{N-1} LT\left(\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right], \langle r\rangle_i^A\right) \\ &= \sum_{i=0}^{N-1}\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right] - \sum_{i=0}^{N-1}\langle r\rangle_i^A + 2^\ell \cdot \sum_{i=0}^{N-1} t_i^{(1)} \end{aligned} \tag{1.6}$$

Next, we explain how to compute the first two terms $\sum_{i=0}^{N-1}\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right]$, $\sum_{i=0}^{N-1}\langle r\rangle_i^A$.

2. Compute $t^{(2)}$, where $\sum_{i=0}^{N-1}\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right] = [x+r] + t^{(2)}\cdot 2^\ell$

Each party can locally compute $\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right]$ by taking $\langle x\rangle_i^A$ and $\langle r\rangle_i^A$ as plaintext values, compute the addition and modulo operation, and broadcast it to other parties. Then all parties use the received values to compute $\sum_{i=0}^{N-1}\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right] = [x+r] + t^{(2)}\cdot 2^\ell$ in plaintext, where $t^{(2)}$ is an integer. Note that both $[x+r]$ and $t^{(2)}$ are public known, but as r is a random value, $[x+r]$ reveals no information about x.

3. Compute $\left\langle t^{(3)}\right\rangle^A$, where $\sum_{i=0}^{N-1}\langle r_i\rangle^A = r + t^{(3)}\cdot 2^\ell$

For $\sum_{i=0}^{N-1}\langle r_i\rangle^A$, the parties can deploy an adder circuit. More specifically, each party i with private input $\langle r_i\rangle_0^B, \ldots, \langle r_i\rangle_{N-1}^B$ compute $\left\langle t^{(3)}\right\rangle^A$ together, where $\sum_{i=0}^{N-1}\langle r_i\rangle^A = r + t^{(3)}\cdot 2^\ell$ and $\left\langle t^{(3)}\right\rangle^A$ are computed by converting the corresponding output bits of the adder circuit without revealing r.

Next, we have

$$\begin{aligned} x + t\cdot 2^\ell &= \sum_{i=0}^{N-1}\left[\langle x\rangle_i^A + \langle r\rangle_i^A\right] - \sum_{i=0}^{N-1}\langle r\rangle_i^A + 2^\ell \cdot \sum_{i=0}^{N-1} t_i^{(1)} \\ &= [x+r] + t^{(2)}\cdot 2^\ell - \left(r + t^{(3)}\cdot 2^\ell\right) + 2^\ell \cdot \sum_{i=0}^{N-1} t_i^{(1)} \\ &= [x+r] - r + 2^\ell \cdot \left(t^{(2)} - t^{(3)} + \sum_{i=0}^{N-1} t_i^{(1)}\right) \end{aligned} \tag{1.7}$$

As $[a-b] = [a] - [b] + 2^{\ell} \cdot LT([a],[b])$ (cf. Eq. (1.2)), when we replace a with $[x+r]$ and b with r, we get as follows:

$$[[x+r]-r] = [x+r] - r + 2^{\ell} \cdot LT([x+r], r) \tag{1.8}$$

which can be transformed into:

$$[x+r] - r = [[x+r]-r] - 2^{\ell} \cdot LT([x+r], r) \tag{1.9}$$

Next, we take above equation into Eq. (1.7) and get as follows:

$$\begin{aligned} x + t \cdot 2^{\ell} &= [x+r] - r + 2^{\ell} \cdot \left(t^{(2)} - t^{(3)} + \sum_{i=0}^{N-1} t_i^{(1)} \right) \\ &= [[x+r]-r] - 2^{\ell} \cdot LT([x+r], r) + 2^{\ell} \cdot \left(t^{(2)} - t^{(3)} + \sum_{i=0}^{N-1} t_i^{(1)} \right) \end{aligned} \tag{1.10}$$

4. Compute $\langle t^{(4)} \rangle^A$, where $t^{(4)} = LT([x+r], r)$

As $[[x+r]-r] = x$, **??** can be transformed as follows:

$$\begin{aligned} x + t \cdot 2^{\ell} &= [[x+r]-r] - 2^{\ell} \cdot LT([x+r], r) + 2^{\ell} \cdot \left(t^{(2)} - t^{(3)} + \sum_{i=0}^{N-1} t_i^{(1)} \right) \\ &= x - 2^{\ell} \cdot t^{(4)} + 2^{\ell} \cdot \left(t^{(2)} - t^{(3)} + \sum_{i=0}^{N-1} t_i^{(1)} \right) \\ &= x + 2^{\ell} \cdot \left(\sum_{i=0}^{N-1} t_i^{(1)} + t^{(2)} - t^{(3)} - t^{(4)} \right) \end{aligned} \tag{1.11}$$

The parties compute and obtain $\langle t^{(4)} \rangle^A$ together, where $t^{(4)} = LT([x+r], r)$. Note that $[x+r]$ is public known and r is secret shared as $\langle r \rangle_i^A$. Therefore, we use the Π^{LT} [MRVW21] to compare the public known $[x+r]$ with arithmetic share $\langle r \rangle_i^A$ and convert the comparison result into arithmetic share $\langle t^{(4)} \rangle^A$.

And each party i set $\langle t \rangle_i^A = t_i^{(1)} + t^{(2)} - \langle t^{(3)} \rangle_i^A - \langle t^{(4)} \rangle_i^A$ as an arithmetic share of t, where $t^{(2)}$ is public known and $t_i^{(1)}$ is a private value.

List of Figures

List of Tables

List of Abbreviations

Bibliography

[EGK+20] D. Escudero, S. Ghosh, M. Keller, R. Rachuri, P. Scholl. **"Improved primitives for MPC over mixed arithmetic-binary circuits"**. In: *Annual International Cryptology Conference*. Springer. 2020, pp. 823–852.

[MRVW21] E. Makri, D. Rotaru, F. Vercauteren, S. Wagh. **"Rabbit: Efficient Comparison for Secure Multi-Party Computation"**. Cryptology ePrint Archive, Report 2021/119. `https://ia.cr/2021/119`. 2021.

[RBS+22] D. Rathee, A. Bhattacharya, R. Sharma, D. Gupta, N. Chandran, A. Rastogi. **"SecFloat: Accurate Floating-Point meets Secure 2-Party Computation"**. Cryptology ePrint Archive, Report 2022/322. `https://ia.cr/2022/322`. 2022.