UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Informatik und Mathematik
Arbeitsgruppe Codes und Kryptographie

Title

Bachelor's Thesis

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science

by
Niklas Isermann

submitted to:
Prof. Dr. Johannes Blömer
and
???

Paderborn, August 7, 2022

Eidesstattliche Versicherung

Nachname: ______________________ Vorname: ______________________

Matrikelnr.: __________ Studiengang: ______________________________

☐ Bachelorarbeit ☐ Masterarbeit

Titel der Arbeit: Title

☐ Die elektronische Fassung ist der Abschlussarbeit beigefügt.

☐ Die elektronische Fassung sende ich an die/den erste/n Prüfenden bzw. habe ich an die/den erste/n Prüfenden gesendet.

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit (Ausarbeitung inkl. Tabellen, Zeichnungen, etc.) selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Abschlussarbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Die elektronische Fassung entspricht der gedruckten und gebundenen Fassung.

Belehrung

Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist die Vizepräsidentin / der Vizepräsident für Wirtschafts- und Personalverwaltung der Universität Paderborn. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz NRW in der aktuellen Fassung).

Die Universität Paderborn wird ggf. eine elektronische Überprüfung der Abschlussarbeit durchführen, um eine Täuschung festzustellen.

Ich habe die oben genannten Belehrungen gelesen und verstanden und bestätige dieses mit meiner Unterschrift.

Ort: ______________________ Datum: __________

Unterschrift: ______________________________

Datenschutzhinweis

Die o.g. Daten werden aufgrund der geltenden Prüfungsordnung (Paragraph zur Abschlussarbeit) i.V.m. § 63 Abs. 5 Hochschulgesetz NRW erhoben. Auf Grundlage der übermittelten Daten (Name, Vorname, Matrikelnummer, Studiengang, Art und Thema der Abschlussarbeit) wird bei Plagiaten bzw. Täuschung der/die Prüfende und der Prüfungsausschuss Ihres Studienganges über Konsequenzen gemäß Prüfungsordnung i.V.m. Hochschulgesetz NRW entscheiden. Die Daten werden nach Abschluss des Prüfungsverfahrens gelöscht. Eine Weiterleitung der Daten kann an die/den Prüfende/n und den Prüfungsausschuss erfolgen. Falls der Prüfungsausschuss entscheidet, eine Geldbuße zu verhängen, werden die Daten an die Vizepräsidentin für Wirtschafts- und Personalverwaltung weitergeleitet. Verantwortlich für die Verarbeitung im regulären Verfahren ist der Prüfungsausschuss Ihres Studiengangs der Universität Paderborn, für die Verfolgung und Ahndung der Geldbuße ist die Vizepräsidentin für Wirtschafts- und Personalverwaltung.

Contents

1 Abstract

2 Introduction

2.1 MPC and Databases

A famous problem in the context of MPC is Yao's millionaire's problem. In Yao's millionaire's problem there are two millionaires Alice and Bob. We will call Alice's wealth x and Bob's wealth y. Alice and Bob want to know who of them is has more money. i.e. they want to compute the function F(x,y) := $\begin{cases} Alice\, is\, richer & y \leq x \\ Bob\, is\, richer & y > x \end{cases}$. Yet neither of them is willing to trust the other and tell him how much money he has. Yao's millionaire's problem can be generalised into the general MPC problem. Instead of Bob and Alice, we now consider n parties $p_0, \ldots, p_{n-1}$ and each party i holds an arbitrary input x_i for an arbitrary function F$(x_0, \ldots, x_{n-1})$, that all parties have agreed upon. A MPC protocol π is protocol, that allows $p_0, \ldots, p_{n-1}$ to compute F$(x_0, \ldots, x_{n-1})$ without revelling any information about $x_0, \ldots, x_{n-1}$.

Andrew Yao proposed a solution for Yao's millionaire's problem in 1982 []. It has also been shown that MPC is Turing-complete[]. This means that for any function f that can be computed with a Turing machine. There exists a MPC protocol π that can compute f. -Databases ...

2.2 related work

2.2.1 From Keys to Databases—Real-World Applications of Secure Multi-Party Computation

2.3 goals

In this section we describe the goals of our work.

2.4 structure

In this section we outline the structure this document.

3 preliminary

3.1 MPC definition

In the an MPC scenario there are n parties $p_0, \ldots, p_{n-1}$ that want to compute an agreed upon functionality $\mathrm{F}(x_0, \ldots, x_{n-1})$. A functionality is a function that is allowed to have internal randomness, so its not function in the strict mathematically sense of the word. Each party p_i holds an input value x_i and does not want to reveal anything about it. Therefore they jointly execute an MPC protocol π that enables them to compute $\mathrm{F}(x_0, \ldots, x_{n-1}.)$ without revelling their inputs. The attacker or adversary in this scenario has the ability to corrupt one or more party's. Once a party is corrupted the adversary get full information about every message the party send our receives, this also includes the messages from the time before the party had been corrupted.

3.1.1 passive adversary vs active adversary

There are multiple categorizations of adversary's and their capability's. On such categorization is the distinction between passive and active adversary's. A passive adversary can not force a corrupted party to deviate from the protocol in an any way. A active adversary has the power to force a corrupted party to deviate from the protocol in an arbitrary way. So if for example the protocol would at some point require that each party choses an integer between 1 and n uniformly at random. Then a passive adversary would have no choice but to choose the integer between 1 and n uniformly at random. On the contrary an active adversary would be able to force a corrupted party to chose the value 42 or any other value that the adversary considers to be advantageous for him.

3.1.2 general adversary vs threshold adversary

In the threshold MPC setting the adversary can choose to corrupt any party. The threshold adversary is only limited in the way that can at most corrupt t parties where t is set to be $0 < \mathrm{t} < \mathrm{n}$. A common setting for t is $\mathrm{t} = \lfloor \frac{n}{2} \rfloor$, which called the honest majority. For example and for n=3 the presence of an honest majority means, that it is assumed that the threshold adversary can corrupt at most 1 party. Threshold MPC best fits scenarios that feature a very homogenous group of parties. A general adversary is limited in his choice which party he corrupts by an adversary structure $Z = \{Z_1, \ldots, Z_l\}$. Where Z_i can be any set of parties. The general adversary must corrupt a set of parties P such that there exists an $x \in Z$ that holds $P \subset x$. This allows for a flexible way to formalise assumptions. If for example in protocol there are two parties that hold a very vital role and one want to assume that no adversary can corrupt both of these parties.

That can be formalised by using a general adversary an defining Z so that no element in Z contains both of these two parties.

3.1.3 static adversary vs dynamic adversary

Another important distinction is the distinction between static and adaptive adversary's. An static adversary is bound to chose which parties he wants to corrupt before the execution of π starts. An adaptive adversary can corrupt a party during the execution of the protocol. This makes the adaptive adversary much more powerful. As he can try to identify "weak links" based on the information he gets during the execution of the protocol and then choose corrupt those.

3.2 real world vs ideal world

For semi-honest adversary's we can now formalise the question if a protocol π archives our security goal of "not revelling $x_0, \ldots, x_{n-1}$.", this is done by describing an ideal world where there exits a perfect solution for the MPC problem and them comparing the execution of π to this ideal world. In an ideal world there exists an incorruptible third party P that all parties trust. In the real world there is no such incorruptible third party. Instead the parties execute the protocol π be exchanging messages. In the ideal world evaluating $\mathrm{F}(x_0, \ldots, x_{n-1}.)$ can be done in two steps. In a first round of communication party p_i send x_i to P. This gives P all the information required for computing $\mathrm{F}(x_0, \ldots, x_{n-1})$. In a second round of communication P send each party $\mathrm{F}(x_0, \ldots, x_{n-1})$. If $\mathrm{F}(x_0, \ldots, x_{n-1})$ is computed this way the ability of the adversary to gain new knowledge about $x_0, \ldots, x_{n-1}$ is minimized. We say that π is secure against adversary A if A cannot learn more information by attacking π in the real world then by attack the ideal world. This can be shown using simulation based proof. The View of a party consist of its input, the state of its memory which includes its internal randomness i and all messages it received. We say π is secure against A, if their exists an probabilistic polynomial-time simulator S that given the ideal world views of all parties A corrupts can compute the corresponding views in the real world. We require that the output of S has an identical distribution of values as the views that A would see when A attacks π in the real world. Given such S exists, A could instead of attacking the real world, simply attack the ideal world and then run S to get views that are identical distributed as the views A would have obtained by attacking the real world. So their is no advantage for A in attacking the real world compared to attacking the ideal world. Furthermore π is secure ,if π is secure against all A.

4 framework description

4.1 CipherCompute

On candidate for our study was Cipher Compute. With the CypherCompute framework it is possible to solve a huge range of MPC problems using Rust. These include SQL operations like joins that are of interest for us. Furthermore CypherCompute provides a rich documentation, consisting of a full quickstart guide and several well documented example projects. CypherCompute utilises the SCALE-MAMBA framework for its underlying MPC operations. SCALE-MAMBA itself has evolved out of the well-known SPDZ protocol. Unfortunately the early access version of CypherCompute is not functioning by the time we conducting this study. Therefore we have decided to not include CypherCompute in our study.

4.2 Prio+

Prio+ [AGJ+21] is the next generation of the highly influential Prio [CGB17]. Prio+ strives to maintain the same use and security as Prio, while significantly increasing performance compared to its predecessor. Prio Plus allows an arbitrary number of parties to jointly compute aggregated statistics, like SUM, MAX, MIN operators. Prio+ utilises a client server model. In which the (potentially many) input parties use a small number of servers to compute the statistics. Prio+ guarantees confidentiality of the input values if at least one server stays honest. Unlike CipherCompute or conclave Prio+ is not a framework for developing MPC solutions. Its rather an already complete system. This means that the use of Prio+ can not be extended beyond the usecases that have been originally implemented by the authors of Prio+. This leaves Prio+ with a relatively small range of usecases compared to frameworks like aby3 or conclave. Therefore we have decided to not include Prio+ in our study.

4.3 conclave

4.3.1 functionality

4.3.2 underlying MPC technology

- published in 2019,
- combining MPC operations and cleartext processing
- moving operations outside of MPC to maximise performance
- allows to explicitly annotate trust between parties

- utilises trust through hybrid operations for additional performance increase
- maintaining same end-to-end security as "pure" MPC
- relying on Obliv-C and Sharemind as MPC backend
- backend theoretic exchangeable through generic interface

- Query optimization aims to move as work outside of mpc
- contrary to conventional sql operations that aims to minimize the total amount of work e.g. filters before join

- secure against semi-honest adversary
- adversary is statically
- no secure channel setting
- inherits security guarantees of backend -> honest majority
- compares to "SMCQL most similar existing system"
- jiff dependency
- requires python 3.5
- ...

4.4 aby3

4.4.1 functionality

4.4.2 underlying MPC technology

In [MRR20] .. -3 Party setting
, - honest majority
- passiv adversary
- all protocols constant rounds of communication
- O(n) overhead in for join where n is number of rows
- supports inner left ,full join, set union,set minus, single table operation like where or aggregate. prototype - does feature a LAN VS WAN comparison in benchmarks

4.5 smcql

4.5.1 functionality

4.5.2 underlying MPC technology

- prototype for 2 parties can be expanded for more parties - honest adversary

5 implementation

6 evaluation

Bibliography

[Eti14] Y. Eti. On the Importance of Correct Stirring. *International Journal of Cookie Theory*, 13(1):1–247, 2014.

[MRR20] Payman Mohassel, Peter Rindal, and Mike Rosulek. *Fast Database Joins and PSI for Secret Shared Data*, page 1271–1287. Association for Computing Machinery, New York, NY, USA, 2020.