Secure Multi-Party Computation with SCALE-MAMBA

Jonas Rülfing

Seminar: Privacy and Big Data - RWTH Aachen

Abstract. This thesis takes a look at SMPC especally using SCALE-MAMBA. For that it introduces first what SMPC is, how SPDZ the precessor of SCALE-MAMBA works and what are the differences. It discusses the functionalities of SCALE-MAMBA, especially the split into online and offline phase, how the beaver tripels are used for sacrificing and in which complexity classes the calculations are done. Finally it compares scale mamba shortly to other competitors in the market and gives an outlook what next technologies are coming right now and are coming in the near future in SMPC.

1 Motivation and Introduction

a) Motivation: - There are a lot of computations that have to be done in a distributed environment in todays technologies and in the future - Not possible to have all the information in one place anymore - Sometimes hard to know who to trust - Problem: want to compute something together but we don't want to share our information - A lot of use cases: number of distributed environments is climbing rapidly; for example see crypto currencies (distributed block chain; no one can do something alone, everyone relies on everyone else (or at least on a majority of the other participants) - A lot of research. Right now more theoretical than practical works; a lot of computational overhead - Scale-Mamba: o Developed by a research team of the KU Leuven o Implements a lot of the theoretical foundations o Crypto suit, end to end o Possible to be safe against corruption with abort o Good (compared to others) execution times o Improvement of before existing technologies (SPDZ especially, and others)

b) Structure: This thesis will try to give an overview over Secure Multiparty Computation, especially focusing on Scale-Mamba and all the technologies it builds upon. Therefore, it will explain the basic technologies first in an abstracted manner and later go more into detail.

2 Cryptography Basics

a) Message authentication code (MAC) b) Oblivious transfer (OT) c) Homomorphic Encryption (HE) d) Somewhat Homomorphic Encryption (SHE) //e) Access structures? //f) Monotone Span Programs (MSP) ? g) FHE – fully homomorphic encryption? h) Different adversary types a. Malicious b. Honest but corious

2.1 Adversary types

Cryptography is the science of protecting information or communication from an adversary. This adversary is normally a third party that wants to get information for his own gain or want to sabotage something because of his own interest. In the are of SMPC we we normally divide between two types of adversaries. In SMPC we expect that everyone but ourselves can be controlled by an adversary. That means the adversary can control between one and all of the other parties with which we are communicating. The difference to the normal cryptographic adversary model is, that the adversary is one of the parties we are communicating with but not a third party that tries to intercept communication.

Honest but curious This type of adversary tries to gain as much information as possible. But this adversary is obeying to the rules of our protocol, that means it is impossible to know that this party is controlled by a third entity. This adversary tries to retrieve some of the secret information from the other parties, by using all the information he gets during the execution of the protocol.

2.2 Malicious adversary

While this adversary also wants to gain all the information, he is not obeying to the rules of the protocol. This type of adverary can send different inputs to manipulate the results he receives. Furthermore, he wants to interrupt the whole protocol. That could mean making the computation impossible, but as well modifying his own input or his own calculations, so the final result is changed and might lead to different implications

It is important to decide between these two types of adversary, because it shows the different securities SMPC has to guarantee. It has to guarantee that no information is unwillingly shared with an illegal party and at the same time verify that the final result has a correct value.

Maybe move this behind the basic introduction of SMPC? Some parts like access structures

3 SMPC Basics and History

3.1 Beginnings

As we saw in the introduction, there are a lot current and much more upcoming usecases of calculating the output of a function in a distributed environment with multiple parties while at the same time keeping all the individual inputs secret. The first explorations in these fields started in the 1970s, with theoretical problems like the mental poker problem(how can two players play poker while not being in the same physical location and making sure noone is cheating) and Yaos millionaire problem(how can can two millionaires decide who is richer than the other while not revealing their wealth).

These explorations lead to a group of basic protocols which established the area of Secure Multi-Party Computation(SMPC).

Now we will take a short look on the two most famous of these protocols, which both were the basis for a whole family of algorithms established in the years to follow until today.

3.2 Yaos Garbled Circuit

Most MPC protocols fall into one of two broad categories: garbled circuits, and linear-secret-sharing-scheme-based (LSSS-based) MPC. The garbled-circuit approach, which began with the work of Yao [38], involves some collection of parties garblingä circuit to conceal the internal circuit evaluations, and then later a single party or a collection of parties jointly evaluating the garbled circuit.

3.3 Shamirs Secret Sharing Scheme

LSSS - linear secret sharing scheme

By contrast, the LSSS-based approach involves using a so-called linear secret-sharing scheme, in which the parties: shareä secret into several shares which are distributed to different parties, perform computations on the shares, and then reconstruct the secret at the end by combining the shares to determine the output. LSSS-based MPC is traditionally presented in the context of information-theoretic security, although many modern practical protocols that realise LSSS-based MPC often make use of computationally-secure primitives such as somewhat-homomorphic encryption (SHE) [25] or oblivious transfer (OT) [32]. In this paper, we focus on computationally-secure LSSS-based MPC.

3.4 Terminology

Over the years different terminologies came up in the are of SMPC. Following will be a short introduction into some of them, so their definitions are clear.

Oblivious Transfer

Access structures There are a lot of different use cases for SMPC, so of course the adversary model changes depending on the system, the security needs, and the amount of parties involved. it is important to differentiate between these different use cases, because the different needs have big implications for security measurements that have to be taken and for runtime complexity. SMPC should run as fast as possible, while still providing all the neccesary security guarantees.

To describe which party has which rights, and how many parties can be corrupted so the security can still be guaranteed, access structures were introduced.

An access structure for a set of parties de

nes which subsets of parties are allowed to discover the secret if they pool their information. Such quorums of parties are often called quali

ed sets of parties. An access structure is called Q‘ (for ‘ 2 N) if the union of any set of ‘ unquali

ed sets of parties is missing at least one party. We discuss this in some detail later, but for now the reader can think of an (n; t)-threshold scheme where t < n=‘ which is where a subset of parties is quali

ed if and only if it is of size at least t + 1. Computationallysecure LSSS-based MPC has recently seen signi

cant, effcient instantiations for full-threshold access structures [7, 23, 25, 32], which is where the protocol is secure if at least one party is honest, even if the adversary causes the corrupt parties to run arbitrary code (though this behaviour may cause the protocol to abort rather than provide output to the parties). In the threshold case similar effcient instantiations are known, such as the older VIFF protocol [21] which uses (essentially) information-theoretic primitives only.

Monotone Span Programs a. Part of cryptography that was started in the 1970s and is getting much more relevance in the last years b. YAOs Millionaire problem c. Mental poker d. Yaos garbled circuit e. Shamirs secret sharing scheme i. Choose random polynomial out of finite field to a prime ii. Degree of polynomial t is the number of tolerated adversaries iii. Polynomial: f(0) = s iv. Send shares to each other player v. T+1 shares needed to reconstruct secret (with Lagrange interpolation in O(n hoch 2)) vi. These can be used for different operations; for example some calculations can be made on the secrets, and at the end the final result is revealed and calculated and compared; so a result can be calculated without revealing the own secret f. Other secret sharing schemes i. Replicated (KNF based) ii. Dx‘NF based b) SMPC basics - Access structures - Montone span programs? - Offline/online phase - Secret Sharing foundations (mathematical) c) Early adoptions of LSSR; MASCOT? d) SPDZ - Maybe not too many details, because a lot of things are similar to SCALE-MAMBA?

4 SCALE-MAMBA

a) Why we need it? Problems of SPDZ, other SMPC protocols b) Architecure - Written in C++ - Scale: Secure Computation Algorithms from LEuven o SMPC foundations o Code base based on different projects like SPDZ, HE. . . ?,. o Combination of offline online phase o We will focus more on this then on the Compiler part, its more relevant for this thesis o - Mamba: Multiparty AlgorithMs Basic Argot o Compiler; similar to python o Why necessary? Easily able to write own configurations for SMCP for SCALE-MAMBA c) Idea, Computational Logic - Online Offline phase - Offline phase: o Calculation of beaver triples o Precalculations like MAC? - Online phase: o Secret sharing o Sacrifice of triples to check calculation results o Check for adversaries (MAC?) o Abortion - Options/ Configurations: o It is possible to use different setups, depending on the taste of the user d) Performance, real world usage, problems, upcoming developments, - Upcoming: Calculating offline data before, and then compiling the Mamba code just in time (especially full threshold access structures need a lot of time)

5 Comparison

a. What else is out there? b. Different use cases – when which SMPC tool is more useful; what are the strengths of SCALE-MAMBA compared to other ones? c. Factual comparisons

6 Outlook and Conclusion

a. What brings the future in general in SMPC? b. SMPC is a technology that will shape the future (at least a bit)

a. SCALE-MAMBA is amazing b. We saw how it works in which cases it is more useful c. We saw why SMPC is necessary; we saw that we need it (or at least we should use it)

7 TODO LIST

Format: Your report should have a minimum length of 15 pages LNCS Style, excluding references. Your report has to include references, but they do not count towards your total page limit. The 'literature list' is the list of references.

Try not to write more than 20 pages unless you feel you have a very good reason.

Template: The LNCS is template is available for Latex [1] and Word [2] on the Springer page, but I suggest you use this quick start version [3].

How to correctly include references in your literature list:

Please refer to e.g. the Chicago manual of style [5].

Any valid reference style is accepted, as there are many styles, and as they basically contain the same information.

Next steps: I will send you instructions about the peer review process closer to the next deadline.

Best regards, Benjamin Heitmann.

- Use Latex - Find examples/ empirical results - Gmw protocol – scapi library LSSR based, passive as comparison - Reusing results: maybe check in night for adversaries? - Could be nice to include reuse -

8 Bibliography

9 Abbreviations

List of abbrviations LSSR SMPC SPDZ OT MAC HE SHE SCALE MAMBA

10 Introduction

The remainder of the paper starts with a presentation of related work (Sect. 11). It is followed by a presentation of hints on LaTeX (Sect. 12). Finally, a conclusion is drawn and outlook on future work is made (Sect. 13).

11 Related Work

Winery [2] is a graphical modeling tool. The whole idea of TOSCA is explained by Binz et al. [1].

12 LaTeX Hints

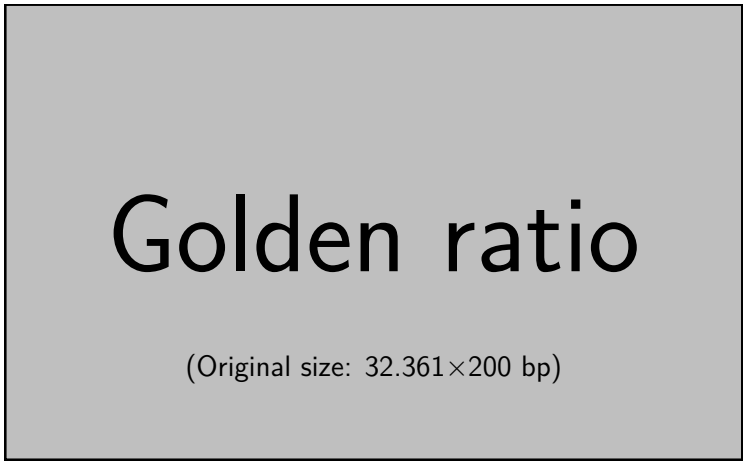

Fig. 1. Simple Figure. [based on 3]

Table 1. Simple Table

Heading1	Heading2
One	Two
Thee	Four

Listings 1 and 2 show listings typeset using the `lstlisting` environment.
cref Demonstration: Cref at beginning of sentence, cref in all other cases.
Figure 1 shows a simple fact, although Fig. 1 could also show something else.
Table 1 shows a simple fact, although Table 1 could also show something else.
Section 10 shows a simple fact, although Sect. 10 could also show something else.

```
public class Hello {
    public static void main (String[] args) {
        System.out.println("Hello␣World!");
    }
}
```

List. 1. Example Java Listing

```
<example attr="demo">
  text content
</example>
```

List. 2. Example XML Listing

Brackets work as designed: <test> One can also input backquotes in verbatim text: `test`.

The symbol for powerset is now correct: $\wp$ and not a Weierstrass p ($\wp$).

1. All these items... 2. ...appear in one line 3. This is enabled by the paralist package.

Please use the “qq command” or the “enquote command” to quote something. “something in quotes” using plain tex syntax also works.

You can now write words containing hyphens which are hyphenated (application-specific) at other places. This is enabled by an additional configuration of the babel package. In case you write “application-specific”, then the word will only be hyphenated at the dash. You can also write application-specific, but this is much more effort.

The words “workflow” and “dwarflike” can be copied from the PDF and pasted to a text file.

Numbers can written plain text (such as 100), by using the siunitx package like that: 100 $\frac{\mathrm{km}}{\mathrm{h}}$, or by using plain LaTeX (and math mode): $100\frac{km}{h}$.

13 Conclusion and Outlook

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Acknowledgments ...

In the bibliography, use `\textsuperscript` for "st", "nd", ...: E.g., "The 2nd conference on examples". When you use JabRef, you can use the clean up command to achieve that. See `https://help.jabref.org/en/CleanupEntries` for an overview of the cleanup functionality.

References

1. Binz, T., Breiter, G., Leymann, F., Spatzier, T.: Portable Cloud Services Using TOSCA. IEEE Internet Computing 16(03), 80–85 (May 2012)
2. Kopp, O., et al.: Winery – A Modeling Tool for TOSCA-based Cloud Applications. In: Proceedings of 11th International Conference on Service-Oriented Computing (ICSOC'13). LNCS, vol. 8274, pp. 700–704. Springer Berlin Heidelberg (2013)
3. Scharrer, M.: The `mwe` Package (2017), `http://texdoc.net/mwe`

All links were last followed on October 5, 2017.