Secure Multiparty Computation: Background and Use Cases

João Santos

Universidade de Évora, Portugal
d47238@alunos.uevora.pt
https://www.uevora.pt/

Abstract. This paper presents an overview of how Secure Multiparty Computation (MPC) has been used and how applications for this technology have evolved through the years. MPC was introduced in the 1980s with the goal of creating methods for parties to jointly compute a function over their inputs while keeping those inputs private. These methods use cryptographic and distributed system techniques to achieve the goal of sharing computational effort while maintaining some data private and unknown to other participating parties or even unrelated eavesdropping third parties. There are many applicability domains such as Healthcare and voting. Recently, MPC has been used to allow private data support in a blockchain platform. The adoption for these techniques has been slow but great progress is shown to have been made.

Keywords: Privacy · Distributed Systems · Cryptography.

1 Introduction

In nowadays increasingly digital world, cryptography is used as the backbone of many modern services. Cloud Computing is readily available and this means that it is possible to scale computing resources easily, according to predicted usage. Distributed computing can be used to improve the performance of a computation using connected computing devices. However, as lives become more digital, more of our sensitive data may be at risk.[11]

Secure Multiparty Computation (MPC), enables parties to carry out such distributed computing tasks with at least two important requirements. These requirements are privacy and correctness. The privacy requirement states that nothing should be learned beyond what is absolutely necessary. That is to say, parties should only learn the output of a calculation, with inputs remaining private. The correctness requirement states that each party should receive its correct output.[4]

In this paper, an introduction to secure multiparty computation and its applicability is presented. In Section 2 a brief historical perspective on cryptography is presented, and its main concepts are introduced. Distributed systems and concepts are introduced in Section 3. Section 4 presents MPC and its use cases. Finally there is a brief discussion about MPC applicability in modern society.

2 Classical and Modern Criptography

Cryptography has a long history. Classical cryptography concerns itself with historical use cases of these techniques. It is important to note that most of these have no real use case with today's vast computing capabilities. Classical cryptography uses transposition and substitution techniques. [8]

2.1 Substitution Ciphers

A Substitution cipher uses the simple concept of replacing one or more characters in a message. Caesar's cipher is one of the first known encryption techniques where Julius Caesar sent messages to his generals where each alphabetical character is replaced with the respective character three positions ahead. In this case, the key is embedded in the algorithm. This means that for the encryption to remain effective, the algorithm has to remain a secret. This is an instance of security through obscurity.

2.2 Transposition Ciphers

A Transposition cipher encrypts messages by shifting plain text according to a predetermined pattern. The encrypted message is derived from reordering the plain text. Many of these techniques use several encryption iterations to further disrupt decryption attempts and increase cryptographic strength. Many modern cryptography techniques also use several iterations.

2.3 Evolving Cryptography Study

Katz et. al (2007) define modern cryptography as the scientific study of techniques for securing digital information, transactions and distributed computations. While classical cryptography was widely regarded as an art, modern cryptography is regarded as a science. The application of cryptography also evolved after the late 20th century. Traditionally cryptography was mainly used by military and intelligence organizations, while nowadays cryptography is used for securing computing systems and services that modern society relies on.[8]

2.4 Kerckhoffs' Principle

Auguste Kerckhoffs, in the late 19th century stated: *the cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.* This statement is now simply known as the Kerckhoffs principle. In other words, it states that any cryptographic algorithm must be made public as it allows stronger scrutiny and standardization. The principle stands in stark contrast with how security was achieved in earlier years of cryptography, as previously discussed in subsection 2.1. This principle is widely accepted and embraced in modern cryptography.

2.5 Symmetric Cryptography

Algorithms that use the same key to perform encryption and decryption operations are symmetric. These algorithms may use either stream or block ciphers. Stream ciphers encrypt a message by generating a stream of pseudo-random bits of variable length and performing a XOR operation against the plain text. Block ciphers take in a fixed number of bits, called a block, and encrypt those bits as a single unit.[8,10]

2.6 Asymmetric Cryptography

Algorithms that use a pair of keys, a public and a private key, are classified as asymmetric cryptography algorithms. The public key can be made widely available, while the private key must be known only to the owner. Public key algorithms are fundamental security constructs in modern digital services.[9]

Asymmetric algorithms are slower when compared to symmetric algorithms. Some security schemes use these algorithms as a way to provide key distribution in an initial stage of communication and then use symmetric cryptography algorithms with the exchanged key in the following communications. Asymmetric cryptography algorithms are also used to provide digital signatures. [8]

3 Distributed Computing

Steen et. al (2017) define a distributed system as a collection of autonomous computing elements that appears to its users as a single coherent system. Nowadays many essential services share these characteristics, especially with the advent of the internet and widespread availability of mass communications. Sometimes these computing elements may be owned by different competing parties.

3.1 Fault Tolerance and Byzantine Failures

A distributed system is subject to failures. While there are many failure types, this paper focuses on a specific failure type called Byzantine failures. Byzantine failures are non deterministic errors caused by one or more Byzantine faults. In this situation, one or more components of a distributed systems act inconsistently. While some components of the system might be able to perceive the error, others might not detect it. When dealing with distributed systems, measures and preparations must be taken, so that there is resistance to these types of failures. [2,11]

4 Secure Multiparty Computation

As discussed, distributed computing often deals with questions of computing under the possibility of different failures. While byzantine failures can be caused by malicious behaviour from any entity it is not the focus of distributed computing

literature. On the other hand, secure multiparty computation is concerned with the possibility of deliberately malicious behaviour by a participating or external party. Adversarial behavior models must therefore be defined.[11]

4.1 Definition of Security

In this domain of research it is imperative that the definition of security in regards to adversarial behaviour is defined, as it allows further conclusions. From now on, this paper assumes a semi-honest adversary model of security, unless specified otherwise.

The semi-honest adversary model is when some corrupt parties run the protocol honestly but try to learn as much as possible from a protocol execution. Such an adversary can also be called "honest-but-curious" or "passive". [4]

4.2 Building blocks for MPC

There are two building blocks for MPC that are important to understand the foundations of MPC protocols. The first is the oblivious transfer (OT) protocol. This is a type of protocol in which a sender transfers one of potentially many pieces of information to a receiver. In this case, the sender does not know what piece of information was transferred or if any was transferred at all. The receiver knows nothing about the other possible pieces of information.

The second is a zero-knowledge proof (ZKP) or zero-knowledge (ZK) protocol. This is a type of protocol where a prover can convince a verifier that it knows some truth without revealing anything other than the fact that he is able to know this fact. In this case, a prover is an entity that allows the verifier to prove a truth.[12]

4.3 Two Party Computation

Secure two-party computation (2PC) is sub-problem of MPC. Yao (1982) introduced what is now known as Yao's Millionares' problem. In this problem, two millionaires, Alice and Bob, wish to know which one is richer without revealing any information about their wealth to each other. In other words, the goal is to compute the result of $x1 \leq x2$, where $x1$ is the Alices's private input and $x2$ is Bob's private input. In this case, $x1$ is Alice's wealth and $x2$ is Bob's wealth. The solution presented assumes a finite solution space and uses asymmetric cryptography techniques and oblivious transfer techniques to transfer sensitive data in an encrypted manner.

4.4 Protocol Implications

This protocol is known as Yao's Garbled Circuit and is the basis of many MPC protocols. Yao (1982) also proved that no eavesdropping third party could interfere. However the protocol is proven to not be secure against actively malicious

participating parties. It is however considered secure, by the author, in a semi-honest adversary model.

More recently, there have been variants of the protocol where the security model is relaxed, allowing malicious parties in some degree. This has allowed for further optimizations to the protocol. In the malicious adversary model, the corrupted parties will not follow protocol specification. Malicious adversaries can also be called "active".[12]

4.5 Building upon Initial Work

Further work was achieved by Goldreich et. al. (1987), where a general procedure for partial-information multi-party cryptographic computation problems was developed. The algorithm developed uses secret sharing of all the inputs and zero-knowledge proofs, as well as introducing fault tolerance procedures. Further work was achieved on an oblivious transfer (OT) protocol. 1–2 oblivious transfer was introduced and generalized to "1 out of n oblivious transfer".[12]

4.6 Use Cases

There are many use cases for MPC. Some are presented in this paper, but it is by no means an extensive list. Consider, for example, the need to compare a patient's medical data against a database of other patients medical data, with the goal of finding if the person belongs to a high risk group for a certain type of disease. MPC can be used to solve a wide variety of healthcare related problems, enabling the utilisation of sensitive data without compromising privacy.

Another example use case is electronic voting. The tallying of votes can be described as the simple computation of the addition function for each vote. There is a need to guarantee that each vote is not tampered with, as well as, the requirement that each vote must remain private. This aligns with the fundamental requirements of an MPC protocol.

Hyperledger Fabric, a permissioned blockchain also uses MPC to handle private data. In a blockchain, inherently all the participating peers must have the same view of the shared ledger since a blockchain is, generally speaking, a continuously growing list of records being written on a ledger. The ledger is a structure where all records are written and stored, which is constantly replicated across all participating parties.

It is important to support private data transactions as it enables new application in areas like healthcare where data is sensitive and in the face of increasing privacy related regulations. Hyperledger Fabric added support for private data using On-chain Secure-MPC to leverage the benefits that a permissioned distributed ledger provides, given that the underlying trust model in a permissioned distributed ledger is essentially the same as the one used in MPC protocols. In both cases, different often competing and distrustful parties must communicate to accomplish a common goal.[3,7]

5 Discussion

Cryptography and distributed systems techniques are common nowadays to secure the critical digital services that modern society relies on. MPC is a domain of research that provides protocols for securing these systems. While adoption of these techniques is slow, great progress is shown to have been made, as discussed in Subsection 4.6. Furthermore, if society as a whole becomes more aware of privacy related issues, the need for MPC based approaches can potentially justify the investment in the face of increasing privacy related regulations.

References

1. Lindell, Y., Pinkas, B.: Secure Multiparty Computation for Privacy-Preserving Data Mining. The Journal of Privacy and Confidentiality, pp. 59–98 (2009)
2. Driscoll, K., Hall, B., Sivencrona, H., Zumsteg, P.: Byzantine Fault Tolerance, from Theory to Reality. Computer Safety, Reliability, and Security, pp. 235–248 (2003)
3. Santos, J.: Identity Management in Healthcare using Blockchain Technology. Universidade de Évora (2018)
4. Lindell, Y.: Secure Multiparty Computation (MPC). Cryptology ePrint Archive, Report 2020/300 (2020)
5. Yao, A.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), pp. 160–164. IEEE, Chicago (1982)
6. Galil, Z., Haber, S., Yung, M.: Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model. In: Pomerance C. (eds) Advances in Cryptology (CRYPTO '87), pp. 135–155. Springer, Berlin (1988)
7. Benhamouda, F., Halevi, S., Halevi, T.: Supporting Private Data on Hyperledger Fabric with Secure Multiparty Computation. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 357–363. IEEE, Orlando (2018)
8. Boavida, F., Bernardes, M.: Introdução à Criptografia. 1st ed. FCA, Lisboa (2019)
9. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. 2nd ed. CRC Press, Florida (2015)
10. Stallings, W.: Cryptography and Network Security. 7th ed. Pearson Education Ltd., England (2017)
11. Steen, M., Tanenbaum, A.: Distributed Systems. 3rd ed. Pearson Education, Inc., England (2017)
12. Evans, D., Kolesnikov, V., Rosulek, M.: A Pragmatic Introduction to Secure Multi-Party Computation. 1st ed. Now Publishers Inc., Hanover (2018)