A-SIT Technology
E-Government Innovation Center

5 March 2024 Overview · Privacy & Crypto · 25 min read

Evaluation of MPC's State of the Art

Karl W. Koch



Authors Karl W. Koch
Version 2023-11-30
Source https://technology2.a-sit.at/en/publications/evaluation-of-mpcs-state-of-the-art/
Note Machine translation.

Key findings

- MPC computations run via Boolean or Arithmetic Circuits; Marbled Circuits switch between them and can be more efficient depending on the computation.
- Stronger attacker models mean computationally more expensive protocols; models differ by majority, type of corruption and timing.
- MPC engines are either generic (e.g. MP-SPDZ, MOTION, JIFF in the browser) or aimed at specific numbers of parties and PPML (e.g. ABY-3, CrypTen).
- Use cases include the Danish sugar beet auction in 2008, distributed backups via secret sharing and privacy-preserving federated learning.
- Besides MPC, TEEs, HE and ZKPs are three further methods for computing on encrypted data, and they can be combined.

Data holds the enormous potential to learn a lot from it and thus improve practically all areas of life. Especially when we combine data from different sources (e.g. organizations or individuals). One of the biggest challenges in such scenarios is to protect the privacy of individuals.

„(Secure) Multi-Party Computation (MPC) is a technology that can be used, among other things, to perform data analyses from various sources while maintaining the privacy of the individual data providers. On the one hand, MPC is being increasingly used in practice, mainly due to relatively-mature MPC engines and frameworks such as MP-SPDZ. On the other hand, the relatively-large development of the theoretical and practical aspects of MPC over the last two decades has led to numerous choices and requirements when using MPC in real-world applications.

For example, there exist several threat models in MPC. Further, also different MPC engines/frameworks and MPC protocols/computation models to choose from exist; depending on the use case and general requirements on MPC. This diversity can be an initial hurdle on the way to a smooth use of MPC in real scenarios. Therefore, this report looks at the state of the art of MPC. The focus is placed on the areas of computation paradigms and environmental parameters, MPC engines/frameworks, threat models, use cases, and MPC within as well as offered by companies. Finally, the general status of computation on encrypted data is considered. Further resources on MPC and secure computations on encrypted data in general are shown in the appendix.

1. Introduction

Data holds the enormous potential to learn a lot from it and thus improve practically all areas of life. Especially when we combine data from different sources (e.g. organizations or individuals). One of the biggest challenges in such scenarios is to protect the privacy of individuals.

„(Secure) Multi-Party Computation“ (MPC) is a technique that can be used, among other things, to perform data analyses from various sources while maintaining the privacy of the individual data providers. On the one hand, MPC is being increasingly used in practice, mainly due to relatively mature MPC engines and frameworks (e.g. MP-SPDZ [1]). On the other hand, the relatively large development of the theoretical and practical aspects of MPC over the last two decades has led to a „jungle“ of choices and requirements when using MPC in real-world applications. The field of MPC – like the sub-area „computation on encrypted data“ in cryptography in general – has developed relatively quickly over the last ~10 years, especially in its practical aspects. Depending on the use case and environmental parameters, different computation paradigms have emerged.

For example, there are several MPC threat models. Further, depending on the use case and general requirements on MPC, different MPC engines and protocols/computation models are available. This „MPC jungle“ is an initial hurdle on the way to a smooth use of MPC in real scenarios.

To shed more light on this „MPC jungle“, this project looks at the state of the art of MPC with regard to computation paradigms and parameters, threat models, MPC engines/frameworks, use cases, and MPC within as well as offered by companies. For example, the subcategory of MPC engines shows common engines that offer different trade-offs in, e.g., different programming languages. There are, e.g., some engines with C++ as the computation basis, but also at least one in JavaScript to run MPC directly in the

browser; and some engines for running arbitrary MPC programs with a (in principle) arbitrary number of computing parties, and on the other hand engines with a focus on privacy-preserving machine learning (PPML) and/or a specific number of computing parties. That is, depending on the use case (programming language/PPML/etc.) and environmental parameters (number of desired or required parties/etc.), certain MPC engines are potentially interesting (and the others are not).

In addition, the general state of computing on encrypted data (COED) is examined, with a focus on potential further work. Since space is limited on the one hand and, on the other hand, only a snapshot can be given here, further resources on MPC are also listed in the appendix; they offer the possibility to dive even deeper into MPC.

2. Computation Paradigms and Parameters

This section presents computation paradigms and parameters of MPC.

In principle, MPC distinguishes between computations via „Garbled/Boolean Circuits“ [2] and „Arithmetic Circuits“. Boolean Circuits are based on logic gates, such as AND and XOR, and MPC programs are represented, so to speak, as a Boolean circuit. Arithmetic Circuits are based on arithmetic operations, such as addition and multiplication, and are usually realized with the method of secret sharing. There are again different approaches to secret sharing, but mostly a variation of Shamir's Secret Sharing [3] is used, such as the protocol of Maurer [4]. Arithmetic Circuits generally work more efficiently for operations modulo a prime number, whereas Boolean Circuits are generally more efficient for binary operations, such as comparing whether two numbers are equal. Because of this efficiency difference based on the underlying operations, there are also „Marbled Circuits“ (mixed arithmetic & Boolean), which can switch between an Arithmetic and a Boolean Circuit during the execution of an MPC program. Depending on the computation, the overhead of switching circuits in Marbled Circuits can be more efficient than realizing the computation only via a Boolean or an Arithmetic Circuit.

In addition, the MPC protocol by „Smart-Pastro-Damgård-Zakarias“ (SPDZ protocol [5]) has contributed to more efficient MPC computations. The SPDZ protocol has a preparation phase („Offline Phase“) and a computation phase („Online Phase“). In the Offline Phase, expensive computations that are independent of the input data are performed, which are then used in the actual Online Phase, which depends on the input data. Mainly so-called „Beaver Triples“ [6] are generated in the Offline Phase, with which multiplications of numbers that have been secret shared can be realized; additions can be performed locally by the different computing participants.

Besides the underlying computation paradigms, the corresponding environmental parameters are also relevant for MPC computations. The most relevant general environmental parameters include (A) the number of participating computing parties (2, 3, 4, ..., 10, ..., etc.), (B) the intended or existing network environment (e.g. LAN with e.g. 0.1ms latency and 10Gbit/s bandwidth, WAN with 50ms latency and 100Mbit/s bandwidth, or various gradations between LAN and WAN), (C) the types of MPC parties (peer-to-peer, where every party that provides data takes part in the computation, or „outsourced“, where a few dedicated computing parties receive the input data, perform the computation, and then make the result available generally or to dedicated parties), (D) which MPC engine or framework, and (E) which underlying protocol and threat model is to be used.

3. Threat Models

As mentioned at the end of the previous section, MPC has several threat models that offer different trade-offs. In principle, a protocol that is secure against „stronger" adversaries is computationally more expensive than a protocol that is secure against „weaker" adversaries. This efficiency difference for protocols with more security results from more security checks, such as a MAC check for every exchange of secret-shared data. In principle, threat models distinguish between (A) the majority of trustworthy parties („Majority"), (B) the corruption by potential adversaries, and (C) the time of such a corruption.

With regard to the majority of the participating computing parties, there is a distinction between a trustworthy majority („Honest Majority") and no trustworthy majority („Dishonest Majority") . In an Honest Majority, at least t parties are trustworthy. Depending on the protocol, t is, for example, $n/2$ or $n/3$ (n = number of participating computing parties). In a Dishonest Majority, at least one computing party is (or remains) trustworthy; that is, all but one party can be corrupt.

With regard to the type of corruption by potential adversaries – how a potential adversary disrupts the protocol and tries to find out information from trustworthy participants – there are different variants and definitions. Passive or „Semi-Honest" adversaries follow the protocol or program flow, but try to learn as much as possible about the other computing participants. Active or „Malicious" adversaries can additionally deviate from the protocol or program flow. „Robust MPC" (Smart, 2023) means that even if adversaries deviate from the protocol, the trustworthy computing participants can still continue to compute the protocol or program. Robust MPC is the strongest security property among the types of corruption, and (so far) only works for an Honest Majority. „Security with Abort" means that if a computing participant deviates from the protocol or program flow, the other computing participants detect this and abort the MPC computation. „Covert" is by definition basically the same as Active or Malicious, plus that it is found out with a certain probability which of the computing parties has (become) an adversary. The Covert threat model is particularly interesting in scenarios where the computing parties build trust through successful MPC computations, for example like in a point system, and therefore have, so to speak, „a reputation to lose".

With regard to the time at which a potential corruption takes place, there is a distinction between Static and Adaptive. With static corruptions, it is fixed at the beginning of the MPC computation how many parties can become corrupt. With adaptive corruptions, any number of different computing parties can become corrupt in the course of the MPC computation; in the practical context, of course, provided that they are within the respective majority (Honest or Dishonest Majority).

4. MPC Engines and Frameworks

MPC engines or frameworks („MPC engines") are software systems that make the functionality of MPC possible in practice in the first place. Normally, each participating computing party operates its own engine. Essentially, there are two types of MPC engines: (1) specifically geared to the number of participating computing parties or (2) generic (in principle any number of participating parties). In addition, there are also MPC engines that focus on computing machine learning (ML) via MPC; so to speak privacy-preserving ML („PPML"). Moreover, the various MPC engines support different underlying (security) protocols and are based on different programming languages.

In the following, selected common MPC engines for generic computations, as well as for specific computations, are presented. Further MPC engines can be found, e.g., on the overview page of „@rdragos“ on GitHub (Rotaru, 2022).

4.1. Generic MPC Engines

This section lists one of the most common engines for generic MPC computations.

SCALE [7] („Secure Computation Algorithms from LEuven“) was developed by KU Leuven (Belgium), and is one of the two (official) successors of Bristol's „SPDZ-2“ [8], one of the first MPC engines that implemented the SPDZ protocol. The BackEnd of SCALE, which performs the actual computations, is based on C++. The FrontEnd, in which users define the MPC protocol, is based on „MAMBA“ (Multiparty AlgorithMs Basic Argot; file extension .mpc), which is a custom-developed Python-like programming language. There is also the option to use the programming language Rust for the FrontEnd. For the MPC computations, the FrontEnd code is compiled into a bytecode, which is then read in by the BackEnd. However, SCALE has not been developed further since about summer 2022. In principle, MP-SPDZ is very similar, though, and now offers considerably more functionality.

MP-SPDZ [9] („Multi-Protocol SPDZ“) is developed by Marcel Keller (Data61, Australia), and is the second (official) successor of Bristol's „SPDZ-2“. As with SCALE, the BackEnd is based on C++ and the FrontEnd on „MAMBA“ (.mpc).

MOTION [10], [11] („Framework for Mixed-Protocol Multi-Party Computation“) is developed by „encryptogroup“ (Lennart Braun, Daniel Demmler, Thomas Schneider, Oleksandr Tkachenko, et al.; TU Darmstadt, Germany). „An efficient, user-friendly, modular, and extensible framework for mixed-protocol secure multi-party computation with two or more parties“. With MOTION, the BackEnd and FrontEnd are based on C++.

FRESCO [12] („FRamework for Efficient and Secure COmputation“) is developed by the „Alexandra Institute“ (Denmark), and is based on Java. MPyC [13] („Multiparty Computation in Python“) is developed by Berry Shoenmakers (Eindhoven University of Technology, Netherlands), and is based on Python. JIFF [14], [15] („JavaScript library for building web-based applications that employ MPC“) is developed by „multiparty“ (Boston University, USA), and is based on JavaScript. By using JavaScript, JIFF can also be run directly in the web browser . swanky [16] („Suite of Rust libraries for MPC“) is developed by Galois Inc. (Portland, Oregon, USA), and offers libraries and individual functionalities for MPC in Rust.

4.2. MPC Engines for Specific Computations

This section lists one of the most common engines for specific MPC computations. Specific can mean, e.g., that the number of computing parties is limited, or that the engine was developed specifically for PPML.

Some of these MPC engines focus on execution with 3 computing parties: ABY-3 [17] („Three Party MPC framework for Machine Learning and Databases“) is developed by Peter Rindal (Visa Research, USA), and is based on C++. ABY-3 focuses on PPML and database operations. For the use case with 2 computing parties there is also ABY [18] („Framework for Efficient Mixed-protocol Secure Two-party Computation“), which is developed by TU Darmstadt (Germany). TF Encrypted [19] („Framework for Encrypted Machine Learning in TensorFlow“) is developed by Dropout Labs / Cape Privacy (capeprivacy.com) and is based on Python. TF-Encrypted focuses on PPML via TensorFlow [20]. Rosetta [21] („Privacy-Preserving Framework

Based on TensorFlow“) is developed by Yuanfeng Chen, Gaofeng Huang, Junjie Shi, Xiang Xie, Yilin Yan (LatticeX Foundation, Singapore), and is based on Python and C++. Rosetta also focuses on PPML via TensorFlow.

CrypTen [22],[23] („Framework for Privacy Preserving Machine Learning“) is developed by Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, Laurens van der Maaten (Facebook AI Research, California, USA), and is based on Python. CrypTen works with an arbitrary number of computing parties and focuses on PPML via PyTorch [24].

EzPC [25],[26],[27],[28] („Easy Secure Multiparty Computation“) is developed by Microsoft Research India, and defines a language for PPML and links other tools to enable PPML, which results in CryptFlow. CryptFlow takes, e.g., TensorFlow code as input and translates it into a PPML protocol. It supports different MPC engines that execute the actual MPC protocol in the background, such as ABY.

5. Use Cases

MPC is in principle applicable wherever data sets are distributed and different (participating) parties want to compute a „function“ such that only the result becomes known. For example, when each participating party contributes data – such as fitness data from a watch/smartwatch. The result can be made available only to a dedicated party, or also to all or some participating parties. Furthermore, nobody learns the input data, except – of course – the inputting party itself. The MPC approach is therefore particularly relevant wherever none of the parties wants to or can disclose their data sets. The first real large-scale use of MPC was the Danish sugar beets in 2008 [29]. A sugar beet auction was carried out between Danisco – the Danish processor of sugar beets – and Danish farmers in order to coordinate the sugar beet price. It was above all important to the farmers to keep their input data private, which (potentially) contributed to increased economic fairness; made possible by the use of MPC.

In addition, the underlying concept of secret sharing is also of independent interest for certain use cases. For example, it can be used to create distributed backups relatively easily. A user can, e.g., split the data sets across 5 different cloud providers such that no single cloud provider learns anything about the data, while at the same time it is enough to bring the data of 3 cloud providers back together to reconstruct the original data set. This (e.g.) „3-out-of-5“ data set method – to reconstruct the original data set – is usually realized by means of „Shamir’s Secret Sharing“ [30].

The use case of privacy-preserving federated learning can also use MPC techniques to protect the ML updates of the individual users. For example, the updated ML parameters of the respective users can be split among all other users in such a way that the aggregating server only learns the global sum over all users, and cannot draw conclusions about individual users. One of the best-known such concepts is „SecAgg“ [31] by Google.

Furthermore, the computation approaches of MPC are so generic that practically any program can be computed (in a distributed manner). A prominent example is running ML via MPC (PPML). In this case, e.g., the input data, such as ML training data, can stay with the respective user and the ML model can still be trained.

Another side aspect of MPC is the application to so-called „Zero-Knowledge Proofs“ (ZKPs), which are used to prove to a verifying party that one knows a „secret“ without revealing the secret itself. Such ZKPs can be used, e.g., to prove that one is older than 18 years without stating the specific age. The method for realizing ZKPs via MPC is usually called „MPC in the Head“ [32].

6. MPC within and by Companies

This section lists companies that use MPC or offer it to customers. The selection of the companies is based on the overview article on computing on encrypted data by Nigel Smart (Smart, 2023).

Most companies offer solutions for privacy-preserving computations via MPC – and in some cases additionally via other methods: CipherMode Labs [33],[34] offers (e.g.) a Python module for computing MPC. Cybernetica [35] offers a wide range of – „business-to-business“ – solutions on cybersecurity and privacy-preserving computations; such as digital identities. Inpher [36] focuses on „PPML-as-a-Service“, and also offers, e.g., a „TFHE“ [37] library, which enables efficient computations with homomorphic encryption. Roseman Labs [38] offers various – „business-to-business“ – solutions on privacy-preserving computations; e.g. in the area of smart meters, health, the public sector or finance. Partisia [39] offers general solutions to customer-specific challenges on privacy-preserving computations. In addition, Partisia has developed its own blockchain, on the basis of which its own MPC token ecosystem was developed.

The global companies and „technology giants“ Meta and Google use MPC internally in some areas. Meta uses MPC, e.g., in its advertising ecosystem [40]. Google uses MPC, e.g., in parts of its cloud ecosystem [41].

Furthermore, there are also companies that focus on the aspect of secret sharing of MPC. Astran [42] offers distributed cloud storage via „Secret Sharing“. Fragmentix [43] offers storing data in general via „Secret Sharing“.

7. Further Work

This section points out potentials for further work with a focus on general encrypted computations and their mix of methods.

MPC is one method for realizing computations on encrypted data („Computing on Encrypted Data“ / COED). In a scientific article from 2023, Smart described four essential COED methods (Smart, 2023): MPC, „Trusted Execution Environments“ [44] (TEEs), „Homomorphic Encryption“ [45] (HE), and „Zero-Knowledge Proofs“ [46] (ZKPs). Each of these methods offers its own trade-off.

TEEs make it possible to execute computations shielded in a secure area (e.g. in a dedicated processor or hardware module). The data enters the TEE encrypted and is only decrypted there. After execution, the data is encrypted again and stored outside the TEE. Examples of TEEs are Intel’s SGX [47] („Software Guard eXtension“) and TDX [48] („Trust Domain Extension“), or ARM’s TrustZone [49]. TEEs are used by, e.g., the following companies for specific applications: Cape Privacy, Cosmian, Anjuna.

HE makes it possible to encrypt data in such a way that computations can be performed on the ciphertext, and that, e.g., only the result becomes known upon decryption. No conclusions about the original plaintext can be drawn in the process. E.g., 2 HE ciphertexts can be multiplied with each other and the result is the

multiplication of the corresponding plaintext elements: Decrypt(HE-Encrypt(x) * HE-Encrypt(y)) = x - y . There are different variations of HE. E.g., a distinction is made between „Fully Homomorphic Encryption“ (FHE) and „Somewhat Homomorphic Encryption“ (SHE). SHE generally has more restrictions, but is usually faster; which is why it is more suitable for certain applications. HE is used or offered by, e.g., the following companies: Microsoft (e.g. password manager in the Edge browser [50]), Google, CryptoLab, Desilo, Duality, Enveil, Inpher, Ravel, Tune Insight, Zama.

ZKPs make it possible to create proofs about data without revealing the data itself. That is, in „y = F(x)“, y and F are known, and x remains secret. E.g., this can be used to prove that a person is ≥ 18 years old without revealing the specific age. There are in principle different variations of ZKPs. E.g., the distinction between interactive (prover and verifier are needed synchronously for the protocol) and non-interactive (the verifier can verify the proof asynchronously). Well-known ZKP systems are, e.g., „Succinct Non-interactive ARguments of Knowledge“ (SNARKs) [51], „Scalable Transparent Arguments of Knowledge“ (STARKs) [52], or „Bulletproofs“ [53]. ZKPs are used or offered by, e.g., the following companies: StarkWare, Chain Reaction, Cysic, Ingonyama.

As a next step, it would be interesting to examine the three other methods more closely in order to then assign potentially relevant use cases to the respective method or to a mix of methods. This is because some COED methods unfold their full potential – in the respective use case – when they are combined with other COED methods. E.g.: HE & ZKP: In the attack scenario on HE decryption with a corrupt ciphertext, ZKP helps to show that it is a valid HE ciphertext without knowing the plaintext; thus this attack vector is eliminated entirely. MPC & TEEs: In recent years, numerous side-channel attacks on TEEs have been discovered (e.g. on Intel's SGX), which is why MPC can be used to split the computation across several TEEs. Through the „encryption“ or secret sharing with MPC, a side-channel attack on a single TEE is not enough; with certain MPC protocols, an attacker would have to successfully attack all TEEs to obtain the plaintext. MPC & ZKP: Some ZKP systems require a trusted setup before transactions can start; if the data of this setup becomes known, the entire rest of the process is no longer secure, and therefore there are so-called trusted setup ceremonies in which, e.g., MPC is used to compute this setup in a distributed manner; this increases the security of the whole system, as it drastically reduces the probability that someone learns something about the setup, and was carried out, e.g., by the ZKP system Zcash [54].

COED methods have developed relatively quickly over the last decade from a theoretical building block, via first proofs of concept, to already quite practical methods. In his overview article, Smart suspects that in ~10 years COED methods – and thus computations on encrypted data – will become the new standard, as is the case today with TLS („Transport Layer Security“), which protects data in a transaction from A to B (e.g. in the web browser on the connection between client and server).

Practicable COED methods – in all their variations and mixes – combined with machine learning can potentially usher in a paradigm shift of our digital age. An age in which it is quite normal to constantly collect data and – together with many other individuals/organizations – process it in a privacy-preserving way in order to learn from it and make better predictions, so as to make life (ideally) „more worth living“.

References

Institute, A. (27. February 2023). GitHub/aicis/fresco. Abgerufen am May 2023 von FRESCO - A FRamework for Efficient Secure COmputation: https://github.com/aicis/fresco

Keller, M. (15. May 2023). GitHub/data61/MP-SPDZ. Abgerufen am May 2023 von Versatile framework for multi party computation: https://github.com/data61/MP-SPDZ

KULeuven. (30. March 2022). GitHub/KU Leuven/COSIC/SCALE-MAMBA. Abgerufen am May 2023 von Repository for the SCALE-MAMBA MPC system: https://github.com/KULeuven-COSIC/SCALE-MAMBA

Schoenmakers, B. (March1 2023). win.tue.nl. Abgerufen am May 2023 von MPyC: Multiparty Computation in Python: https://www.win.tue.nl/%7Eberry/mpyc/

Rindal, P. (4. March 2023). The ABY3 Framework for Machine Learning and Database Operations. Abgerufen am May 2023 von GitHub/ladnir/aby3: https://github.com/ladnir/aby3

TF-Encrypted. (8. February 2023). A Framework for Encrypted Machine Learning in TensorFlow. Abgerufen am May 2023 von GitHub/tf-encrypted/tf-encrypted: https://github.com/tf-encrypted/tf-encrypted

Yan, Y. C. (26. April 2022). Rosetta: A Privacy-Preserving Framework Based on TensorFlow. Abgerufen am May 2023 von GitHub/LatticeX-Foundation/Rosetta: https://github.com/LatticeX-Foundation/Rosetta/

Rotaru, D. (14. June 2022). A curated list of multi party computation resources and links. Abgerufen am May 2023 von GitHub/rdragos/awesome-mpc: https://github.com/rdragos/awesome-mpc

Lindell, Y. (December 2020). Secure Multiparty Computation. Communications of the ACM, 64(1), 86-96.

Lindell, Y. (17. May 2023). Resources for Getting Started with MPC. Abgerufen am May 2023 von u.cs.biu.ac.il: https://u.cs.biu.ac.il/~lindell/MPC-resources.html

Evans, D., Kolesnikov, V., & Rosulek, M. (2018). A Pragmatic Introduction to Secure Multi-Party Computation. Boston: NOW Publishers.

Smart, N. (2023). Computing on Encrypted Data. IEE S&P. https://ieeexplore.ieee.org/document/10194492.

Lindell, Y. (2024). Resources for Getting Started with MPC. Von Yehuda Lindell: https://yehudalindell.com/resources for-getting-started-with-mpc/ abgerufen

Appendix: Further Resources

This section lists further resources on MPC and on COED in general (computation on encrypted data). Especially with regard to COED in general, the scientific article „Computing on Encrypted“ [55] by Nigel Smart – published in the journal „IEEE Security and Privacy“, 2023 – offers an up-to-date overview. The report gives an overview of the current state and developments in the (practical) area of COED. In particular, four technologies are examined that currently count as among the most reliable cryptographic approaches for COED computations: „Trusted Execution Environments“, „Homomorphic Encryption“, „Zero-Knowledge Proofs“ and MPC.

In the following, mainly workshops, courses and books are listed that on the one hand offer an extended introduction to MPC and on the other hand also treat certain topic areas of MPC and COED in general in greater depth.

Workshops

- „Theory and Practice in Multi-Party Computation“ [56] (TPMPC) aims to bring together practitioners and theorists working in the area of MPC. TPMPC started in 2012 in Aarhus (Denmark) and takes place annually. For the events of 2020 [57] and 2022 [58] the talks are also available online on YouTube.
- Workshop at IIT Bombay [59] (27–29.03.2017) offers materials for an introduction to MPC, and contains 10 units: (1) „What is MPC?“, (2) „Zero Knowledge“, (3) „Garbled Circuit“, (4) „Randomized Encoding“, (5) „Oblivious Transfer“, (6) „Composition“, (7) „MPC Complexity“, (8) „Honest-Majority MPC“, (9) „MPC in the Head“ and (10) „Asynchronous MPC“.
- Workshops at the „Simons Institute“ of the University of Berkeley (California) [60] offer various workshops on, e.g., MPC and COED.

Courses

- „Cryptographic Computing“ [61] from Aarhus University (Denmark) and mainly by Claudio Orlandi, 2023, covers secure computations in general (e.g. MPC, „Homomorphic Encryption“, „Oblivious Transfer“).
- „Secure Computation“ [62] from the Department of Computer Science at the University of Maryland and mainly by Jonathan Katz, 2013, focuses on MPC and „Zero-Knowlege Proofs“ and mainly contains references to scientific articles.
- „Secure Multi-Party Computation at Scale“ [63] from Boston University via Piazza, 2017, puts the main focus on MPC (e.g. mathematical and algorithmic foundations of MPC, and how to apply MPC protocols in practice).
- „Secure Computation“ [64] from the „Indian Institute of Science“ (Bangalore), 2015, covers secure computations in general (e.g. MPC, „Zero-Knowledge Proofs“, „Private-Set Intersection“, blockchain).
- „Topics in Cryptography“ [65] from Stanford (CS 355) offers, on the one hand, a general introduction to cryptography, and on the other hand also more specialized topics such as MPC, secret sharing, „Zero-Knowledge Proofs“, or „Homomorphic Encryption“.
- „Introduction to Secure Distributed Computation“ [66] from the University of Maryland and mainly by Jonathan Katz, 2021, offers mainly notes and references on general foundations of secure computations and on different computation paradigms of MPC, and also on further COED topics such as „Zero-Knowledge Proofs“ and „Homomorphic Encryption“.

Books

- „A Pragmatic Introduction to Secure Multi-Party Computation“ [67] by David Evans, Vladimir Kolesnikov, Mike Rosulek at NOW Publishers, 2018. „ A broad introduction to the field of secure multi-party computation, covering both the fundamental constructions and many of the recent improvements. The book emphazises the intuition and ideas behind the protocols rather than rigorous proofs.” (Rotaru, 2022) „This short book contains a friendly introduction and comprehensive

survey of techniques for achieving efficient secure computation. The book does not go into details regarding each technique and does not provide proofs. However, it gives a fantastic overview and pointers to where you can find all of the details. As such, it is an invaluable resource." (Lindell, Resources for Getting Started with MPC, 2024)

- „Secure Multiparty Computation and Secret Sharing" [68] by Ronald Cramer, Ivan Damgård, Jesper Nielsen at Cambridge University Press, 2015. „Comprehensive treatment of unconditionally secure techniques for multiparty computation (MPC) and secret sharing." (Rotaru, 2022) „This book includes definitions, constructions for both the computational and information theoretic settings, techniques for efficiency, and more. This is the most comprehensive treatment of the basics of MPC, and I recommend reading it methodically from the beginning." (Lindell, Resources for Getting Started with MPC, 2024)

- „Applications of Secure Multiparty Computation" [69] by Peeter Laud, Liina Kamm at IOS Press, 2015. „Collection of MPC protocols for several real-world tasks such as statistics." (Rotaru, 2022)

- „Engineering Secure Two-Party Computation Protocols" [70] by Thomas Schneider at Springer, 2012. „This book focuses specifically tools for optimizing secure computation in practice, including circuit optimizations, frameworks for constructing protocols, and more. The book provides a very good overview of different techniques and tools that MPC researchers should be familiar with." (Lindell, Resources for Getting Started with MPC, 2024)

- „Efficient Secure Two-Party Protocols: Techniques and Constructions" [71] by Carmit Hazay, Yehuda Lindell at Springer, 2010. „Comprehensive study of efficient protocols and techniques for secure two-party computation - both general constructions that can be used to securely compute any functionality, and protocols for specific problems of interest." (Rotaru, 2022) „This book focuses specifically on the two-party case and efficiency. It includes definitions, general constructions based on the garbled circuit technique, and chapters on specific tools like oblivious transfer, sigma protocols, and more. The specific techniques for general constructions are outdated (with much faster techniques available today), but the text is very relevant for study and includes basic results like a full proof of the security of Yao's garbled circuits construction." (Lindell, Resources for Getting Started with MPC, 2024)

- „Foundations of Cryptography Vol.2 - Basic Applications" [72] by Oded Goldreich at Cambridge University Press, 2004. „Chapter 7 of the book introduces two-party and multiparty computation, contains a thorough and comprehensive definitional treatment, provides a full and detailed proof of the GMW construction, and surveys advanced topics. The book's formal and rigorous treatment makes it a must-read for the MPC researcher." (Lindell, Resources for Getting Started with MPC, 2024)

---

1. github.com/data61/mp-spdz ↩
2. New Directions in Garbled Circuits @ YouTube ↩
3. arxiv.org/abs/2305.18465 ↩

4. sciencedirect.com/science/article/pii/S0166218X05002428 ↩
5. eprint.iacr.org/2011/535 ↩
6. link.springer.com/chapter/10.1007/3-540-46766-1_34 ↩
7. github.com/KULeuven-COSIC/SCALE-MAMBA ↩
8. github.com/bristolcrypto/SPDZ-2 ↩
9. github.com/data61/MP-SPDZ ↩
10. github.com/encryptogroup/MOTION ↩
11. ia.cr/2020/1137 ↩
12. github.com/aicis/fresco ↩
13. win.tue.nl/~berry/mpyc ↩
14. github.com/multiparty/jiff ↩
15. multiparty.org/jiff ↩
16. github.com/GaloisInc/swanky ↩
17. github.com/ladnir/aby3 ↩
18. github.com/encryptogroup/ABY ↩
19. github.com/tf-encrypted/tf-encrypted ↩
20. tensorflow.org ↩
21. github.com/LatticeX-Foundation/Rosetta ↩
22. github.com/facebookresearch/CrypTen ↩
23. crypten.readthedocs.io/en/latest/index.html ↩
24. pytorch.org ↩
25. github.com/mpc-msri/EzPC ↩
26. microsoft.com/en-us/research/project/ezpc-easy-secure-multi-party-computation ↩
27. microsoft.com/en-us/research/publication/ezpc-programmable-efficient-and-scalable-secure-two-party-computation-for-machine-learning ↩
28. microsoft.com/en-us/research/publication/cryptflow-secure-tensorflow-inference ↩
29. en.wikipedia.org/wiki/Danish_Sugar_Beet_Auction ↩
30. dl.acm.org/doi/pdf/10.1145/359168.359176 ↩
31. arxiv.org/abs/2305.18465 ↩
32. Verifiable Encryption from MPC-in-the-Head @ YouTube ↩
33. ciphermode.com ↩

34. github.com/ciphermodelabs ↩
35. cyber.ee ↩
36. inpher.io ↩
37. eprint.iacr.org/2018/421.pdf ↩
38. rosemanlabs.com ↩
39. partisiablockchain.com ↩
40. facebook.com/business/news/our-progress-on-developing-and-incorporating-privacy-enhancing- technologies ↩
41. cloud.google.com/blog/products/identity-security/how-confidential-space-and-mpc-can-help-secure-digital-assets ↩
42. astran.io ↩
43. fragmentix.com ↩
44. en.wikipedia.org/wiki/Trusted_execution_environment ↩
45. en.wikipedia.org/wiki/Homomorphic_encryption ↩
46. en.wikipedia.org/wiki/Zero-knowledge_proof ↩
47. eprint.iacr.org/2016/086 ↩
48. arxiv.org/abs/2303.15540 ↩
49. arm.com/technologies/trustzone-for-cortex-a/tee-and-smc ↩
50. microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge ↩
51. dl.acm.org/doi/10.1145/2090236.2090263 ↩
52. iacr.org/archive/crypto2019/116940201/116940201 ↩
53. eprint.iacr.org/2017/1066 ↩
54. https://z.cash ↩
55. ieeexplore.ieee.org/document/10194492 ↩
56. multipartycomputation.com ↩
57. TPMPC 2020 @ YouTube/Aarhus/Crypto ↩
58. TPMPC 2022 @ YouTube/Aarhus/Crypto ↩
59. cse.iitb.ac.in/~mp/crypto/mpc2017 ↩
60. “Securing Computation” @ “Simons Institute” (2015) ↩
61. users-cs.au.dk/orlandi/crycom ↩
62. cs.umd.edu/~jkatz/gradcrypto2/f13/scribes ↩
63. piazza.com/bu/fall2017/cs591v1/info ↩

64. csa.iisc.ac.in/~arpita/SecureComputation15.html ↩

65. crypto.stanford.edu/cs355/20sp/schedule ↩

66. cs.umd.edu/~jkatz/gradcrypto2/s21 ↩

67. securecomputation.org ↩

68. cambridge.org/dk/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure- multiparty-computation-and-secret-sharing ↩

69. ebooks.iospress.nl/volume/applications-of-secure-multiparty-computation ↩

70. encrypto.de/papers/S11Thesis ↩

71. springer.com/us/book/9783642143021 ↩

72. cambridge.org/core/books/foundations-of-cryptography ↩