A=SIT Technology
E-Government Innovationszentrum

5. März 2024 Überblick · Privatsphäre & Krypto · 25 Min. Lesezeit

Evaluierung von MPC's Stand der Technik

Karl W. Koch



Autorinnen und Autoren	Karl W. Koch
Stand	2023-11-30
Quelle	https://technology2.a-sit.at/veroeffentlichungen/evaluierung-von-mpcs-stand-der-technik/

Kernaussagen

- MPC-Berechnungen laufen über Boolean oder Arithmetic Circuits; Marbled Circuits wechseln dazwischen und können je nach Berechnung effizienter sein.
- Stärkere Angreifermodelle bedeuten rechnerisch aufwendigere Protokolle; unterschieden wird nach Mehrheit, Art der Korruption und Zeitpunkt.
- MPC-Motoren sind entweder generisch (z. B. MP-SPDZ, MOTION, JIFF im Browser) oder auf bestimmte Parteienzahlen und PPML ausgerichtet (z. B. ABY-3, CrypTen).
- Einsatzfelder sind etwa die dänische Zuckerrüben-Auktion 2008, verteilte Datensicherung per Secret Sharing und privatsphären-bewahrendes föderiertes Lernen.

- Neben MPC gibt es mit TEEs, HE und ZKPs drei weitere Methoden für Berechnungen auf verschlüsselten Daten, die sich kombinieren lassen.

Daten bergen das enorme Potenzial, viel aus ihnen zu lernen und damit praktisch alle Lebens-Bereiche zu verbessern. Vor allem, wenn wir Daten aus verschiedenen Quellen (z.B. Organisationen oder Einzelpersonen) kombinieren. Eine der größten Herausforderungen in solchen Szenarien ist die Privatsphäre von Einzelpersonen zu schützen.

„(Secure) Multi-Party Computation“ (MPC) ist eine Technik, mit der unter anderem Datenanalysen aus verschiedenen Quellen durchgeführt werden können, wobei die Privatsphäre der einzelnen Datenlieferanten gewahrt bleibt. Einerseits wird MPC in der Praxis immer häufiger eingesetzt; vor allem wegen relativ ausgereiften MPC-Motoren bzw. -Frameworks wie MP-SPDZ. Andererseits hat die relativ große Weiterentwicklung der theoretischen und praktischen Aspekte von MPC in den letzten zwei Jahrzehnten zu zahlreichen Auswahlmöglichkeiten und Anforderungen geführt, wenn man MPC in realen Anwendungsfällen einsetzen will.

So gibt es beispielsweise mehrere MPC-Angreifer-Modelle. Außerdem stehen je nach Anwendungsfall und allgemeinen Anforderungen an MPC verschiedene MPC-Motoren bzw. -Frameworks und MPC-Protokolle/Berechnungsmodelle zur Auswahl. Diese Vielfalt kann eine Anfangshürde auf dem Weg zu einem reibungslosen Einsatz von MPC in realen Szenarien darstellen. Deshalb wird in diesem Bericht der Stand der Technik von MPC betrachtet. Der Fokus wird auf die Bereiche von Berechnungsparadigmen und Umgebungsparameter, MPC-Motoren bzw. -Frameworks, Angreife-Modelle, Anwendungsfälle und MPC in bzw. von Unternehmen gelegt. Abschließend wird der allgemeine Stand zum Thema Berechnungen auf verschlüsselten Daten betrachtet. Im Anhang werden weiterführende Ressourcen zu MPC und sicheren Berechnungen auf verschlüsselten Daten generell gezeigt.

1. Einleitung

Daten bergen das enorme Potenzial, viel aus ihnen zu lernen und damit praktisch alle Lebens-Bereiche zu verbessern. Vor allem, wenn wir Daten aus verschiedenen Quellen (z.B. Organisationen oder Einzelpersonen) kombinieren. Eine der größten Herausforderungen in solchen Szenarien ist die Privatsphäre von Einzelpersonen zu schützen.

„(Secure) Multi-Party Computation” ((sichere) Mehrparteien-Berechnung / MPC) ist eine Technik, mit der unter anderem Datenanalysen aus verschiedenen Quellen durchgeführt werden können, wobei die Privatsphäre der einzelnen Datenlieferanten gewahrt bleibt. Einerseits wird MPC in der Praxis immer häufiger eingesetzt; vor allem wegen relativ-ausgereiften MPC-Motoren bzw. -Frameworks (z.B. MP-SPDZ [1]). Andererseits hat die relativ große Weiterentwicklung der theoretischen und praktischen Aspekte von MPC in den letzten zwei Jahrzehnten zu einem „Dschungel“ von Auswahlmöglichkeiten und Anforderungen geführt, wenn man MPC in realen Anwendungsfällen einsetzen will. Das Feld von MPC – so wie generell der

Unterbereich „Berechnung auf verschlüsselten Daten“ in der Kryptographie – hat sich in den letzten ~10 Jahren relativ-schnell weiterentwickelt; vor allem der praktische Aspekt. Dabei haben sich je nach Anwendungsfall bzw. Umgebungsparameter unterschiedliche Berechnungs-Paradigmen ergeben.

So gibt es beispielsweise mehrere MPC-Angreifer-Modelle. Außerdem stehen je nach Anwendungsfall und allgemeinen Anforderungen an MPC verschiedene MPC-Motoren und -Protokolle/Berechnungsmodelle zur Auswahl. Dieser „MPC-Dschungel“ stellt eine Anfangshürde auf dem Weg zu einem reibungslosen Einsatz von MPC in realen Szenarien dar.

Um mehr Licht in diesen „MPC-Dschungel“ zu bringen, wird in diesem Projekt der Stand der Technik von MPC im Hinblick auf Berechnungsparadigmen und Parameter, Angreifer-Modelle, MPC-Motoren bzw. -Frameworks, Anwendungsfälle, und MPC in bzw. von Unternehmen betrachtet. Beispielsweise die Unterkategorie der MPC-Motoren zeigt gängige Motoren, welche unterschiedliche Tradeoffs in, z.B., verschiedenen Programmiersprachen anbieten. So gibt es, z.B., einige Motoren mit der Berechnungs-Basis in C++, aber auch zumindest einen in JavaScript, um MPC direkt im Browser auszuführen; und einige Motoren für das Ausführen beliebiger MPC-Programme mit einer (grundsätzlich) beliebigen Anzahl an berechnenden Parteien, und andererseits Motoren mit dem Fokus auf privatsphären-bewahrendes maschinelles Lernen („privacy-preserving machine learning“ / PPML) und/oder eine spezifische Anzahl an berechnenden Parteien. Das heißt je nach Anwendungsfall (Programmiersprache/PPML/etc.) und Umgebungsparameter (Anzahl der gewünschten bzw. benötigten Parteien/etc.), sind bestimmte MPC-Motoren potentiell interessant (und die anderen nicht).

Darüber hinaus wird der allgemeine Stand zum Thema Berechnungen auf verschlüsselten Daten („Computing On Encrypted Data“ / COED) beleuchtet; mit dem Fokus auf potentielle Weiterführende Arbeiten. Da einerseits der Platz beschränkt ist und man andererseits hierbei „nur“ eine Momentaufnahme wiedergeben kann, werden weiters auch Weiterführende Ressourcen zu MPC im Anhang aufgezählt; welche die Möglichkeit bieten, noch tiefer in MPC einzutauchen.

2. Berechnungsparadigmen und Parameter

In diesem Abschnitt werden Berechnungsparadigmen und Parameter von MPC gezeigt.

Grundsätzlich unterscheidet man in MPC Berechnungen via „ Garbled/Boolean Circuits“ [2] (verworrener bzw. boolescher Schaltkreis) und „Arithmetic Circuits“ (arithmetischer Schaltkreis). Boolean Circuits basieren auf logische Gatter, wie AND und XOR, und MPC-Programme werden quasi als boolescher Schaltkreis dargestellt. Arithmetic Circuits basieren auf arithmetische Operationen, wie die Addition und Multiplikation, und werden üblicherweise mit der Methode des geheimen Teilens („Secret Sharing“) realisiert. Bei Secret Sharing gibt es wiederum unterschiedliche Ansätze; doch meist wird eine Variation von Shamir’s Secret Sharing [3] verwendet, wie z.B. das Protokoll von Maurer [4]. Arithmetic Circuits funktionieren grundsätzlich effizienter bei Operationen mit Modulo einer Primzahl, wobei Boolean Circuits grundsätzlich effizienter bei Binär-Operationen sind, wie z.B. der Vergleich ob zwei Zahlen gleich sind. Durch diesen Effizienz-Unterschied basierend auf den unterliegenden Operationen, gibt es auch „Marbled Circuits“ (arithmetisch & boolesch gemischt), welche während der Ausführung eines MPC-Programms zwischen einem Arithmetic und BooleanCircuit wechseln können. Dabei kann - je nach Berechnung - der Mehraufwand des Circuit-Wechsels bei Marbled Circuits effizienter sein, wie wenn man die Berechnung nur via eines Boolean oder Arithmetic Circuits realisiert.

Darüber hinaus hat das MPC-Protokoll von „Smart-Pastro-Damgård-Zakarias“ (SPDZ-Protokoll [5]) zu effizienteren MPC-Berechnungen beigetragen. Beim SPDZ-Protokoll gibt es eine Vorbereitungsphase („Offline Phase“) und eine Berechnungsphase („Online Phase“). Bei der Offline Phase werden eingabedaten-unabhängige aufwendige Berechnungen durchgeführt, welche dann in der eigentlichen eingabedaten-abhängigen Online Phase verwendet werden. Hauptsächlich werden sogenannte „Beaver Triples“ [6] in der Offline Phase generiert, mit denen man Multiplikationen mit Zahlen realisieren kann, welche geheim aufgeteilt wurden (Secret Shared); Additionen können von den unterschiedlichen berechnenden Teilnehmern lokal durchgeführt werden.

Neben den unterliegenden Berechnungsparadigmen, sind auch die entsprechenden Umgebungsparameter relevant für die MPC-Berechnungen. Die relevantesten generellen Umgebungsparameter umfassen (A) die Anzahl der teilnehmenden berechnenden Parteien (2, 3, 4, ..., 10, ..., etc.), (B) die angestrebte bzw. vorhandene Netzwerkumgebung (z.B. LAN mit z.B. 0,1ms Latenz und 10Gbit/s Bandbreite, WAN mit 50ms Latenz und 100Mbit/s Bandbreite, oder verschiedenste Abstufungen zwischen LAN und WAN), (C) die Arten der MPC-Parteien (Peer-to-Peer, wo jede Partei, welche Daten zur Verfügung stellt, an der Berechnung teilnimmt, oder Ausgelagert („Outsourced“), wo wenige dedizierte Berechnungs-Parteien die Eingabedaten erhalten, die Berechnung durchführen, und anschließend das Ergebnis generell oder dedizierten Parteien zur Verfügung stellen), (D) welcher MPC-Motor bzw. -Framework, und (E) welches darunterliegende Protokoll und Angreifer-Modell verwendet werden soll.

3. Angreifer-Modelle

Wie am Ende des vorigen Abschnitts erwähnt, gibt es bei MPC mehrere Angreifer-Modelle, welche unterschiedliche Trade-Offs bieten. Grundsätzlich ist ein Protokoll, welches gegenüber „stärkeren“ Angreifern sicher ist, rechnerisch aufwendiger wie ein Protokoll, welches gegenüber „schwächeren“ Angreifern sicher ist. Dieser Effizienz-Unterschied bei Protokollen mit mehr Sicherheit, resultiert aus mehr Sicherheitschecks, wie z.B. einem MAC-Check bei jedem Austausch von Secret-Shared Daten. Grundsätzlich gibt es bei den Angreifer-Modellen die Unterscheidung zwischen (A) der Mehrheiten von vertrauenswürdigen Parteien („Majority“), (B) der von Korruption von potentiellen Angreifern, und (C) den Zeitpunkt einer solchen Korruption.

In Bezug auf die Mehrheit der teilnehmenden berechnenden Parteien gibt es die Unterscheidung zwischen einer vertrauenswürdigen Mehrheit („Honest Majority“) und keiner vertrauenswürdigen Mehrheit („Dishonest Majority“) . Bei einer Honest Majority sind zumindest t Parteien vertrauenswürdig. Je nach Protokoll ist t beispielsweise $n/2$ oder $n/3$ (n = Anzahl der teilnehmenden berechnenden Parteien). Bei einer Dishonest Majority ist bzw. bleibt zumindest eine berechnende Partei vertrauenswürdig; das heißt, es können alle außer einer Partei korrupt sein.

In Bezug auf die Art der Korruption von potentiellen Angreifern - wie ein potentieller Angreifer das Protokoll stört und versucht Informationen von vertrauenswürdigen Teilnehmern herauszufinden – gibt es unterschiedliche Varianten bzw. Definitionen. Passiv bzw. „Semi-Honest“ folgen dem Protokoll bzw. Programmablauf, aber versuchen soviel wie möglich von den anderen berechnenden Teilnehmern zu lernen. Aktiv bzw. „Malicious“ können zusätzlich auch vom Protokoll bzw. Programmablauf abweichen. „Robust MPC“ (Smart, 2023) bezeichnet, dass auch wenn Angreifer vom Protokoll abweichen, die vertrauenswürdigen berechnenden Teilnehmer trotzdem weiter das Protokoll bzw. Programm berechnen können. Robust MPC ist die stärkste Sicherheitseigenschaft bei den Arten der Korruption, und funktioniert

(bis dato) nur für eine Honest Majority. „Security with Abort“ bezeichnet, dass wenn ein berechnender Teilnehmer vom Protokoll bzw. Programmablauf abweicht, die anderen berechnenden Teilnehmer dies erkennen und die MPC-Berechnung abbrechen. „Covert“ ist von der Definiton her grundsätzlich gleich wie Aktiv bzw. Malicious, plus, dass mit einer gewissen Wahrscheinlichkeit herausgefunden wird, welche der berechnenden Parteien ein Angreifer (geworden) ist. Das Angreifer-Modell Covert ist besonders bei Szenarien interessant, wo die berechnenden Parteien z.B. wie bei einem Punktesystem Vertrauen durch erfolgreiche MPC-Berechnungen aufbauen, und deshalb quasi einen „Ruf zu verlieren haben“.

In Bezug auf den Zeitpunkt wann eine potentielle Korruption stattfindet gibt es die Unterscheidung zwischen Statisch („Static“) und Adaptiv („Adaptive“). Bei statischen Korruptionen ist am Beginn der MPC-Berechnung fixiert wie viele Parteien korrupt werden können. Bei adaptiven Korruptionen können im Laufe der MPC-Berechnung beliebig viele unterschiedliche berechnende Parteien korrupt werden; im praktischen Kontext natürlich gegeben, dass sie im Rahmen der jeweiligen Mehrheit sind (Honest oder Dishonest Majority).

4. MPC-Motoren bzw. -Frameworks

Mit MPC-Motoren bzw. -Frameworks („MPC-Engines“) werden Software-Systeme bezeichnet, welche die Funktionalität von MPC in der Praxis erst ermöglichen. Im Normalfall betreibt jede teilnehmende berechnende Partei einen eigenen Motor. Im Wesentlichen gibt es zwei Arten von MPC-Motoren: (1) spezifisch auf die Anzahl der teilnehmenden berechnenden Parteien ausgerichtet oder (2) generisch (grundsätzlich beliebige Anzahl an teilnehmenden Parteien). Darüber hinaus gibt es auch MPC-Motoren, welche sich auf die Berechnung von maschinellem Lernen (ML) via MPC fokussieren; sozusagen privatsphären-bewahrendes ML („privacy-preserving ML“ / PPML). Darüber hinaus unterstützen die verschiedenen MPC-Motoren unterschiedliche darunterliegende (Sicherheits-)Protokolle und basieren auf unterschiedlichen Programmiersprachen.

Nachfolgend werden ausgewählte gängige MPC-Motoren für generische Berechnungen, sowie auch für spezifische Berechnungen, gezeigt. Weitere MPC-Motoren können z.B. auf der Überblicksseite von „@rdragos“ auf GitHub gefunden werden (Rotaru, 2022).

4.1. Generische MPC-Motoren

In diesem Abschnitt werden eine der gängigsten Motoren für generische MPC-Berechnungen aufgezählt.

SCALE [7] („Secure Computation Algorithms from LEuven“) wurde von KU Leuven (Belgien) entwickelt, und ist einer der zwei (offiziellen) Nachfolger von Bristol’s „SPDZ-2“ [8], einer der ersten MPC-Motoren der das SPDZ-Protokoll implementiert (hat). Das BackEnd von SCALE, welches die eigentlichen Berechnungen durchführt, basiert auf C++. Das FrontEnd, in denen Anwender das MPC-Protokoll definieren, basiert auf „MAMBA“ (Multiparty AlgorithMs Basic Argot; Dateiendung .mpc), welches eine eigens entwickelte Python-ähnliche Programmier-Sprache ist. Weiters gibt es auch die Möglichkeit die Programmier-Sprache Rust für das FrontEnd zu verwenden. Für die MPC-Berechnungen wird der FrontEnd-Code zu einem Bytecode kompiliert, welcher anschließend von dem BackEnd eingelesen wird. Jedoch wird SCALE seit ca. Sommer 2022 nicht mehr weiterentwickelt. Grundsätzlich ist MP-SPDZ aber sehr ähnlich, und bietet inzwischen um einiges mehr an Funktionalität an.

MP-SPDZ [9] („Multi-Protocol SPDZ") wird von Marcel Keller (Data61, Australien) entwickelt, und ist der zweite (offizielle) Nachfolger von Bristol's „SPDZ-2". So wie auch bei SCALE, basiert das BackEnd auf C++ und das FrontEnd auf „MAMBA" (.mpc).

MOTION [10],[11] („Framework for Mixed-Protocol Multi-Party Computation")wird von "encryptogroup" (Lennart Braun, Daniel Demmler, Thomas Schneider, Oleksandr Tkachenko, et al.; TU Darmstadt, Deutschland) entwickelt. "An efficient, user-friendly, modular, and extensible framework for mixed-protocol secure multi-party computation with two or more parties" (ein effizientes, benutzerfreundliches, modulares und erweiterbares Framework für MPC mit 2 oder mehr Parteien und gemischten Protokollen). Bei MOTION basiert das BackEnd und FrontEnd auf C++.

FRESCO [12] („FRamework for Efficient and Secure COmputation") wird vom "Alexandra Institute" (Dänemark) entwickelt, und basiert auf Java. MPyC [13] („Multiparty Computation in Python") wird von Berry Shoenmakers (Eindhoven University of Technology, Niederlande) entwickelt, und basiert auf Python. JIFF [14],[15] („JavaScript library for building web-based applications that employ MPC") wird von "multiparty" (Boston University, USA) entwickelt, und basiert auf JavaScript. Durch die Verwendung von JavaScript kann JIFF auch direkt im Web-Browser ausgeführt werden. swanky [16] ("Suite of Rust libraries for MPC") wird von Galois Inc. (Portland, Oregon, USA) entwickelt, und bietet Bibliotheken bzw. einzelne Funktionalitäten für MPC in Rust an.

4.2. MPC-Motoren für Spezifische Berechnungen

In diesem Abschnitt werden eine der gängigsten Motoren für spezifische MPC-Berechnungen aufgezählt. Spezifisch kann z.B. bedeuten, dass die Anzahl der berechnenden Parteien begrenzt ist, oder der Motor spezifisch für PPML entwickelt wurde.

Einige solcher MPC-Motoren fokussieren sich auf die Ausführung mit 3 berechnenden Parteien: ABY-3 [17] („Three Party MPC framework for Machine Learning and Databases") wird von Peter Rindal (Visa Research, USA) entwickelt, und basiert auf C++. ABY-3 fokussiert sich auf PPML und Datenbanken-Operationen. Für den Anwendungsfall mit 2 berechnenden Parteien gibt es auch ABY [18] („Framework for Efficient Mixed-protocol Secure Two-party Computation"), welches von der TU Darmstadt (Deutschland) entwickelt wird. TF Encrypted [19] („Framework for Encrypted Machine Learning in TensorFlow") wird von Dropout Labs / Cape Privacy (capeprivacy.com) entwickelt, und basiert auf Python. TF-Encrypted fokussiert sich auf PPML via TensorFlow [20]. Rosetta [21] („Privacy-Preserving Framework Based on TensorFlow") wird von Yuanfeng Chen, Gaofeng Huang, Junjie Shi, Xiang Xie, Yilin Yan (LatticeX Foundation, Singapur) entwickelt, und basiert auf Python und C++. Rosetta fokussiert sich ebenfalls auf PPML via TensorFlow.

CrypTen [22],[23] ("Framework for Privacy Preserving Machine Learning") wird von Brian Knott, Shobha Venkataraman, Awni Hannun, Shubho Sengupta, Mark Ibrahim, Laurens van der Maaten (Facebook AI Research, Kalifornien, USA) entwickelt, und basiert auf Python. CrypTen funktioniert mit einer beliebigen Anzahl an berechnenden Parteien und fokussiert sich auf PPML via PyTorch [24].

EzPC [25],[26],[27],[28] („Easy Secure Multiparty Computation") wird von Microsoft Research India entwickelt, und definiert eine Sprache für PPML und verknüpft andere Tools um PPML zu ermöglichen, welches CryptFlow ergibt. CryptFlow nimmt, z.B., als Input TensorFlow-Code und übersetzt diesen in ein PPML-

Protokoll. Es unterstützt unterschiedliche MPC-Motoren, welche im Hintergrund das eigentliche MPC-Protokoll ausführen, wie z.B. ABY.

5. Anwendungsfälle

MPC ist grundsätzlich überall anwendbar wo Datensätze verteilt sind und verschiedene (teilnehmende) Parteien eine „Funktion" berechnen wollen, sodass nur das Resultat bekannt wird. Beispielsweise, wenn jede teilnehmende Partei Daten beiträgt – wie etwa Fitness-Daten von einer Uhr/SmartWatch. Das Resultat kann nur einer dedizierten Partei zur Verfügung gestellt werden, oder auch allen oder manchen teilnehmenden Parteien. Weiters lernt niemand die Eingabedaten, außer – natürlich – der eingebenden Partei selbst. Der Ansatz von MPC ist deshalb überall dort besonders relevant, wo keine der jeweiligen Parteien ihre Datensätze preisgeben will bzw. preisgeben kann. Der erste reale großflächige Einsatz von MPC war bezüglich der dänischen Zuckerrüben in 2008 [29]. Dabei wurde eine Zuckerrüben-Auktion zwischen Danisco – dem dänischen Verarbeiter von Zuckerrüben – und dänischen Bauern durchgeführt, um den Zuckerrüben-Preis zu koordinieren. Dabei war es vor allem den Bauern wichtig ihre Eingabedaten privat zu halten, was (potentiell) zu einer erhöhten wirtschaftlichen Fairness beigetragen hat; ermöglicht durch den Einsatz von MPC.

Darüber hinaus ist auch das unterliegende Konzept vom geheimein Teilen („Secret Sharing") für gewisse Anwendungsfälle von unabhängigem Interesse. Zum Beispiel lassen sich so relativ einfach verteilte Datensicherungen („Distributed Backups") erstellen. Ein Nutzer kann, z.B., die Datensets auf 5 verschiedene Cloud-Anbieter aufteilen, sodass kein einzelner Cloud-Anbieter etwas über die Daten lernt und es gleichzeitig aber reicht, die Daten von 3 Cloud-Anbietern wieder zusammenzuführen, um das ursprüngliche Datenset zu rekonstruieren. Diese (z.B.) „3-von-5"-Datensets-Methode – um das ursprüngliche Datenset zu rekonstruieren - wird meist mittels „Shamir's Secret Sharing" [30] realisiert.

Auch der Anwendungsfall vom privatsphären-bewahrendem föderierten ML („privacy-preserving federated learning") kann Techniken von MPC benutzen, um die ML-Aktualisierungen der einzelnen Nutzer zu schützen. So können z.B. die aktualisierten ML-Parameter der jeweiligen Nutzer in einer Art und Weise auf alle anderen Nutzer aufgeteilt werden, sodass der aggregierende Server nur die globale Summe von allen Nutzern lernt, und keine Rückschlüsse auf einzelne Nutzer ziehen kann. Eines der bekanntesten solcher Konzepte ist „SecAgg" [31] von Google.

Die Berechnungsansätze von MPC sind, weiters, so generisch, dass praktisch jedes Programm (verteilt) berechnet werden kann. Ein prominentes Beispiel ist das Ausführen von ML via MPC (PPML). Dabei können, z.B., die Eingabedaten, wie ML-Trainingsdaten, beim jeweiligen Nutzer bleiben und das ML-Modell kann trotzdem trainiert werden.

Ein weiterer Nebenaspekt von MPC ist die Anwendung für sogenannte „Zero-Knowledge Proofs" (ZKPs), welche verwendet werden um einer verifizierten Partei zu beweisen, dass man ein „Geheimnis" kennt, ohne das Geheimnis selbst preiszugeben. Solche ZKPs können z.B. verwendet werden um zu beweisen, dass man älter als 18 Jahre ist, ohne das konkrete Alter anzugeben. Die Methode um ZKP via MPC zu realisieren wird üblicherweise als „MPC in the Head" [32] (MPC im Kopf) bezeichnet.

6. MPC in bzw. von Unternehmen

In diesem Abschnitt werden Unternehmen aufgezählt, welche MPC verwenden bzw. für Kunden anbieten. Die Auswahl der Unternehmen basiert auf den Überblicksartikel zur Berechnung auf verschlüsselten Daten von Nigel Smart (Smart, 2023).

Die meisten Unternehmen bieten Lösungen zu privatsphären-bewahrenden Berechnungen via MPC – und zusätzlich teilweise auch via andere Methoden – an: CipherMode Labs [33],[34] bietet (z.B.) ein Python-Modul für das Berechnen von MPC an. Cybernetica [35] bietet verschiedenste – „Business-to-Business" - Lösungen zum Thema Cybersicherheit und privatsphären-bewahrende Berechnungen an; wie z.B. digitale Identitäten. Inpher [36] fokussiert sich auf „PPML-as-a-Service", und bietet darüber hinaus, z.B., auch eine „TFHE" [37] -Bibliothek an, welche effiziente Berechnungen mit homomorphen Verschlüsselungen ermöglichen. Roseman Labs [38] bietet unterschiedliche – „Business-to-Business" – Lösungen zum Thema privatsphären-bewahrende Berechnungen an; z.B. im Bereich von Smart Meter, Gesundheit, öffentlichem Sektor oder Finanzen. Partisia [39] bietet generelle Lösungen zu kundenspezifischen Herausforderungen zum Thema privatsphären-bewahrende Berechnungen. Darüber hinaus hat Partisia eine eigene Blockchain entwickelt auf deren Basis ein eigenes MPC-Token-Ökosystem entwickelt wurde.

Die globalen Unternehmen und „Technologie-Riesen" Meta und Google verwenden MPC intern in manchen Bereichen. Meta verwendet MPC, z.B., in ihrem Werbe-Ökosystem [40]. Google verwendet MPC, z.B., in Teilen ihres Cloud-Ökosystems [41].

Weiters gibt es auch Unternehmen welche sich auf den Aspekt des geheimen Teilens („Secret Sharing") von MPC fokussieren. Astran [42] bietet verteilten Cloud-Speicher via „Secret Sharing". Fragmentix [43] bietet generell an Daten via „Secret Sharing" zu speichern.

7. Weiterführende Arbeiten

In diesem Abschnitt werden Potentiale für weiterführende Arbeiten mit dem Fokus auf generelle verschlüsselte Berechnungen und deren Methoden-Mix aufgezeigt.

MPC ist eine Methode um Berechnungen auf verschlüsselten Daten („Computing on Encrypted Data" / COED) zu realisieren. Smart hat in einem wissenschaftlichen Artikel von 2023 vier essentielle COED-Methoden beschrieben (Smart, 2023): MPC, „Trusted Execution Environments" [44] (TEEs), „Homomorphic Encryption" [45] (HE), und „Zero-Knowledge Proofs" [46] (ZKPs). Jede dieser Methoden bietet ihren jeweiligen Trade-Off.

TEEs ermöglichen es Berechnungen abgeschirmt in einem sicheren Bereich auszuführen (z.B. in einem eigenen Prozessor oder Hardware-Modul). Die Daten kommen dabei verschlüsselt in das TEE, und werden erst dort entschlüsselt. Nach der Ausführung werden die Daten wieder verschlüsselt - außerhalb des TEEs - abgelegt. Beispiele für TEEs sind Intel's SGX [47] („Software Guard eXtension") und TDX [48] („Trust Domain Extension"), oder ARM's TrustZone [49]. TEEs werden z.B. von folgenden Unternehmen für spezifische Anwendungen verwendet: Cape Privacy, Cosmian, Anjuna.

HE ermöglicht es Daten so zu verschlüsseln, dass man auf dem Ciphertext Berechnungen durchführen kann, und das dann z.B. nur das Resultat bei der Entschlüsselung bekannt wird. Dabei können keine Rückschlüsse auf den ursprünglichen Klartext gezogen werden. Z.B. kann man 2 HE-Ciphertexte miteinander multiplizieren und das Resultat ist die Multiplikation der entsprechenden Klartext-Elemente: Decrypt(HE-Encrypt(x) * HE-Encrypt(y)) = x - y . Es gibt unterschiedliche Variationen von HE. Z.B. wird zwischen „Fully Homomorphic Encryption“ (FHE) und „Somewhat Homomorphic Encryption“ (SHE) unterschieden. SHE hat grundsätzlich mehr Einschränkungen, ist aber in der Regel schneller; weshalb es für gewisse Anwendungen passender ist. HE wird z.B. von folgenden Unternehmen verwendet bzw. angeboten: Microsoft (z.B. Passwort-Manager im Edge-Browser [50]), Google, CryptoLab, Desilo, Duality, Enveil, Inpher, Ravel, Tune Insight, Zama.

ZKPs ermöglichen es Beweise über Daten zu erstellen, ohne die Daten selbst herzuzeigen. Also dass bei „y = F(x)“, y und F bekannt sind, und x geheim bleibt. Z.B. kann dadurch bewiesen werden, dass eine Person ≥ 18 Jahre ist ohne das konkrete Alter preiszugeben. Es gibt grundsätzlich unterschiedliche Variationen von ZKPs. Z.B. die Unterscheidung zwischen interaktiv (Beweiser („Prover“) & Prüfer („Verifier“) werden synchron fürs Protokoll benötigt) und nicht-interaktiv (Prüfer kann den Beweis asynchron verifizieren). Bekannte ZKP-Systeme sind z.B. „Succinct Non-interactive ARguments of Knowledge” (SNARKs) [51], “Scalable Transparent Arguments of Knowledge” (STARKs) [52], oder “Bulletproofs” [53]. ZKPs werden z.B. von folgenden Unternehmen verwendet bzw. angeboten: StarkWare, Chain Reaction, Cysic, Ingonyama.

In einem nächsten Schritt wäre es interessant, die drei anderen Methoden näher zu beleuchten, um dann potentiell-relevante Anwendungsfälle der jeweiligen Methode bzw. einem Methoden-Mix zuzuordnen. Denn manche COED-Methoden entfalten ihr volles Potential – im jeweiligen Anwendungsfall – wenn sie mit anderen COED-Methoden kombiniert werden. Z.B.: HE & ZKP: Beim Angriffsszenario auf das HE-Entschlüsseln mit einem korrupten Ciphertext hilft ZKP um zu zeigen, dass es sich um einen validen HE-Ciphertext handelt ohne den Plaintext zu kennen; somit hebelt man diesen Angriffsvektor zur Gänze aus. MPC & TEEs: In den letzten Jahren wurden zahlreiche Seitenkanalangriffe auf TEEs ausfindig gemacht (z.B. auf Intel’s SGX), deshalb kann MPC verwendet werden um die Berechnung auf mehrere TEEs aufzuteilen. Durch das „Verschlüsseln“ bzw. Secret Sharing mit MPC ist es nicht genug einen Seitenkanalangriff auf ein einzelnes TEE zu machen; bei gewissen MPC-Protokollen müsste ein Angreifer alle TEEs erfolgreich angreifen um den Plaintext zu erhalten. MPC & ZKP: Einige ZKP-Systeme benötigen einen vertrauenswürdigen Aufbau („Trusted Setup“) bevor die Transaktionen starten können; falls die Daten dieses Aufbaus bekannt werden, ist der ganze restliche Prozess nicht mehr sicher, und deshalb gibt es sogenannte Aufbau-Zeremonien („Trusted Setup Ceremonies“) in denen z.B. MPC verwendet wird um diesen Aufbau verteilt zu berechnen; dies erhöht die Sicherheit des ganzen Systems, da es die Wahrscheinlichkeit drastisch verringert, dass jemand etwas vom Aufbau lernt, und wurde z.B. vom ZKP-System Zcash [54] durchgeführt.

COED-Methoden haben sich im letzten Jahrzehnt relativ schnell von einem theoretischen Baustein, über erste Machbarkeitsnachweise („Proof of Concepts“), zu bereits durchaus praktischen Methoden weiterentwickelt. Smart vermutet in seinem Überblicksartikel, dass in ~10 Jahren COED-Methoden – und somit Berechnungen auf verschlüsselten Daten – der neue Standard werden, so wie es heute mit TLS („Transport Layer Security“) ist, welches die Daten bei einer Transaktion von A nach B schützt (z.B. im Web-Browser bei der Verbindung zwischen Clients und Server).

Praktikable COED-Methoden – in all ihren Variationen und Vermischungen – und der Kombination von maschinellem Lernen können potentiell einen Paradigmenwechsel unserer digitalen Zeit einläuten. Einer Zeit, in der es ganz normal ist ständig Daten zu sammeln und - gemeinsam mit vielen anderen Individuen/Organisationen - privatsphären-schützend zu verarbeiten um daraus zu lernen und bessere Vorhersagen zu treffen, um so das Leben (idealerweise) „lebenswerter“ zu gestalten.

Literaturverzeichnis

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

Anhang: Weiterführende Ressourcen

In diesem Abschnitt sind weiterführende Ressourcen in Bezug auf MPC und COED generell (Berechnung auf verschlüsselten Daten) gelistet. Vor allem in Bezug auf COED generell, bietet der wissenschaftliche Artikel „Computing on Encrypted" [55] von Nigel Smart – publiziert in der Fachzeitschrift („Journal") „IEEE Security and Privacy", 2023 - einen zeitgemäßen Überblick. Der Bericht gibt einen Überblick zum aktuellen Stand und Entwicklungen im (praktischen) Bereich von COED. Im Speziellen werden vier Technologien untersucht, welche zurzeit als eine der zuverlässigsten Kryptographie-Ansätze für COED-Berechnungen zählen: „Trusted Execution Environments", „Homomorphic Encryption", „Zero-Knowledge Proofs" und MPC.

Im Weiteren werden vor allem Workshops, Kurse und Bücher gelistet, welche einerseits einen erweiterten Einstieg in MPC bieten, und andererseits auch gewisse Themenbereiche von MPC und COED generell tiefer behandeln.

Workshops

- „Theory and Practice in Multi-Party Computation" [56] (TPMPC), zielt darauf ab Praktiker und Theoretiker, welche im Bereich von MPC arbeiten, zusammenzubringen. TPMPC startete 2012 in Aarhus (Dänemark) und findet jährlich statt. Zum Event von 2020 [57] und 2022 [58] gibt es die Vorträge auch Online auf YouTube.

- Workshop bei der IIT Bombay [59] (27.—29.03.2017), bietet Materialen für einen Einstieg in MPC, und enthält 10 Einheiten: (1) „Was ist 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" und (10) „Asynchronous MPC".

- Workshops beim „Simons Institute" der Universität Berkeley (Kalifornien) [60], bietet verschiedene Workshops zu, z.B., MPC und COED.

Kurse

- „Cryptographic Computing" [61] von der Universität Aarhus (Dänemark) und vor allem von Claudio Orlandi, 2023, behandelt sichere Berechnungen generell (z.B. MPC, „Homomorphic Encryption", „Oblivious Transfer").

- „Secure Computation" [62] von der Abteilung für Informatik der Universität Maryland und vor allem von Jonathan Katz, 2013, fokussiert sich auf MPC und „Zero-Knowlege Proofs" und enthält hauptsächlich Referenzen zu wissenschaftlichen Artikeln.

- „Secure Multi-Party Computation at Scale" [63] von der Universität Boston via Piazza, 2017, legt den Haupt-Fokus auf MPC (z.B. mathematische und algorithmische Grundlagen von MPC, und wie man MPC-Protokolle praktisch anwenden kann).

- „Secure Computation" [64] vom „Indian Institute of Science" (Bangalore), 2015, behandelt generell sichere Berechnungen (z.B. MPC, „Zero-Knowledge Proofs", „Private-Set Intersection", Blockchain).

- Themen in der Kryptographie [65] („Topics in Cryptography“) von Stanford (CS 355), bietet einerseits eine generelle Einführung in die Kryptographie, und andererseits auch zu spezielleren Themen wie MPC, Secret Sharing, „Zero-Knowledge Proofs“, oder „Homomorphic Encryption“.

- Einführung in sichere verteilte Berechnungen [66] („Introduction to Secure Distributed Computation”) von der Universität Maryland und vor allem von Jonathan Katz, 2021, bietet vor allem Notizen und Referenzen zu generellen Grundlagen sicherer Berechnungen und zu unterschiedlichen Berechnungsparadigmen von MPC, und auch zu weiteren COED-Themen wie „Zero-Knowledge Proofs“ und „Homomorphic Encryption“.

Bücher

- „A Pragmatic Introduction to Secure Multi-Party Computation” [67] von David Evans, Vladimir Kolesnikov, Mike Rosulek bei 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) (via DeepL: „Eine umfassende Einführung in das Gebiet der sicheren Mehrparteienberechnung, die sowohl die grundlegenden Konstruktionen als auch viele der jüngsten Verbesserungen behandelt. Der Schwerpunkt des Buches liegt auf der Intuition und den Ideen hinter den Protokollen und nicht auf strengen Beweisen.”)„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) (via DeepL: „ Dieses kurze Buch enthält eine freundliche Einführung und einen umfassenden Überblick über Techniken zur Erreichung effizienter sicherer Berechnungen. Das Buch geht nicht auf die Einzelheiten der einzelnen Techniken ein und liefert keine Beweise. Es gibt jedoch einen fantastischen Überblick und Hinweise darauf, wo Sie alle Details finden können. Als solches ist es eine unschätzbare Ressource.”)

- „Secure Multiparty Computation and Secret Sharing” [68] von Ronald Cramer, Ivan Damgård, Jesper Nielsen bei Cambridge University Press, 2015. „Comprehensive treatment of unconditionally secure techniques for multiparty computation (MPC) and secret sharing.” (Rotaru, 2022) („Umfassende Behandlung von bedingungslosen sicheren Techniken von MPC und geheimen Teilen”) „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) (via DeepL: „Dieses Buch enthält Definitionen, Konstruktionen sowohl für den rechnerischen als auch für den informationstheoretischen Rahmen, Techniken zur Effizienzsteigerung und vieles mehr. Dies ist die umfassendste Behandlung der Grundlagen der MPC, und ich empfehle, es methodisch von Anfang an zu lesen.”)

- „Applications of Secure Multiparty Computation” [69] von Peeter Laud, Liina Kamm bei IOS Press, 2015. „Collection of MPC protocols for several real-world tasks such as statistics.” (Rotaru, 2022) („Eine Sammlung von MPC-Protokollen für verschiedenste realitäts-relevante Aufgaben, wie Statistiken”)

- „Engineering Secure Two-Party Computation Protocols" [70] von Thomas Schneider bei 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) (via DeepL: „Dieses Buch konzentriert sich speziell auf Werkzeuge zur Optimierung sicherer Berechnungen in der Praxis, einschließlich Schaltkreisoptimierungen, Rahmenwerke für die Konstruktion von Protokollen und mehr. Das Buch bietet einen sehr guten Überblick über verschiedene Techniken und Werkzeuge, mit denen MPC-Forscher vertraut sein sollten.")

- „Efficient Secure Two-Party Protocols: Techniques and Constructions" [71] von Carmit Hazay, Yehuda Lindell bei 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) (via DeepL: „Umfassende Untersuchung effizienter Protokolle und Techniken für die sichere Berechnung von Daten zwischen zwei Parteien - sowohl allgemeine Konstruktionen, die für die sichere Berechnung beliebiger Funktionen verwendet werden können, als auch Protokolle für spezielle Probleme von Interesse.")„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) (via DeepL: „Dieses Buch konzentriert sich speziell auf den Zweiparteienfall und die Effizienz. Es enthält Definitionen, allgemeine Konstruktionen, die auf der Technik der verstümmelten Schaltkreise basieren, und Kapitel über spezifische Werkzeuge wie die verdeckte Übertragung, Sigma-Protokolle und mehr. Die spezifischen Techniken für allgemeine Konstruktionen sind veraltet (da heute schnellere Techniken zur Verfügung stehen), aber der Text ist für das Studium sehr relevant und enthält grundlegende Ergebnisse wie einen vollständigen Beweis für die Sicherheit von Yaos Garbled Circuits Konstruktion.")

- „Foundations of Cryptography Vol.2 - Basic Applications" [72] von Oded Goldreich bei 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) (via DeepL: „Kapitel 7 des Buches führt in die Zwei- und Mehrparteienberechnung ein, enthält eine gründliche und umfassende definitorische Behandlung, liefert einen vollständigen und detaillierten Beweis für die GMW-Konstruktion und gibt einen Überblick über fortgeschrittene Themen. Die formale und strenge Behandlung des Buches macht es zu einer Pflichtlektüre für MPC-Forscher.")

---

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 ↩