Grant Agreement Number: 825225

Safe-DEED

www.safe-deed.eu

Requirements for Secure Computations on Large Datasets with Multiple Owners

Deliverable number	*D5.1*
Dissemination level	*Public*
Delivery data	*due 31.5.2019*
Status	*Final*
Authors	*Lukas Helminger, Sebastian Ramacher*

Horizon 2020
European Union Funding
for Research & Innovation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825225.

Executive Summary

Data marketplaces enable data owners to exchange and exploit their data sets. As the data sets might contain sensitive data, the owners might be reluctanct to actually share their data. When the data contains privacy-sensitive information, legal frameworks even forbid them simply sharing them in plain. Modern cryptographic tools including secure multiparty computation or homomorphic encryption can help to leviate that reluctancy and overcome the obstacle imposed by legal frameworks. These methods ensure that the confidentiality of the data is ensured even if results of some computation on them is revealed to interested customers up to a degree that could also be derived from the result.

Before deploying a system enabling multiple data owners to exploit their data sets while at the same time ensure the confidentiality of these sets, it is of paramout importance to define 1. the desired functionality, 2. who provides input data, 3. who participates in the computation, and finally 4. who defines the results. As these base parameters will later determine the possibles choices of efficient protocols, the use-cases of Safe-DEED will depend on setting them correctly. To help the use-case domain experts to select the suitable setting and parameters, we describe possible scenarios and extend these scenarios with examples.

Contents

List of Figures

1 Introduction

"Big data" became a promising and valuable resource in the last decade. In some instances, data is now considered even more valuable than oil as captured by "data is the new oil" analogy [Eco17], which appeared in research, technology, and business. Developing data marketplaces goes a step further and enable customers to share and capitalize their data sets. Yet, these data marketplaces will only take off if users are willing to share data assets. Often, the benefits and costs of data sharing are not aligned between actors: while one actor gains from the data sharing another only bears privacy or confidentiality threats. Thus, to incentivize data sharing, we need to boost user willingness to share data, e.g. by increasing the trust in data marketplaces. One of the goals of Safe-DEED is to build cryptographically secure methods to enable data owners to capitalize on their data sets without running into privacy or confidentiality issues.

In the cryptographic literature two main lines of work consider privacy- and confidentiality-preserving computations on data sets which are of particular interest for Safe-DEED: *secure multiparty computation* (MPC) and *homomorphic encryption*. On a high level, secure multiparty computation allows data owners to benefit from computing on the data while still retaining the secrecy of the data. In general, the idea of multiparty computation applies to any scenario where data owners would send their input data to a trusted third party to compute the result based on all data sets and then obtain the result from the trusted third party. With MPC, such a trusted third party, that would learn all input data, can be replaced by an MPC protocol. Thereby the need to trust an additionally party can be removed completely. At the same time, the same security guarantee should be achieved: the parties should only be able to learn the result of the computation as in the case where they would send the data to a trusted third party. This power of MPC protocols make them an ideal candidate for leveraging it for secure computations for data markets. MPC protocols also fit scenarios where legislation prohibits the combination of the plaintext data due to privacy concerns [BKL$^+$14].

Besides defining the desired functionality, desired inputs, and results for an MPC-based system, one of the main obstacles to deploy such systems from a technical perspective is the choice of a suitable protocol. While we know of the theoretical existence, finding efficient protocols for a specific functionality is a highly non-trivial task. Related to the choice of functionality and the protocol, an important question to answer is also which parties are in the end performing the computations. If, for example, all nodes performing any of the operations are deployed on a single cloud service, the use of an MPC protocol may only give a false sense of security. The confidentiality and privacy of the data in such a protocol relies to some extent on the assumption that at least some parties do not collaborate by combining their inputs or deviate from the protocol. In other words, designing the topology of the system is paramount to obtain any security guarantees.

For more specialized settings, less general approaches based on, e.g., homomorphic encryption can be a suitable alternative. In particular, it offers interesting opportunities when the parties involved in the computation and the receiver of the results are distinct anyway [GDL$^+$16; DRS17]. While this setting naturally occurs when one is interested in outsourcing computations to, say, a computationally more powerful cloud service provider, it is also applicable to the data marketplace. For example, the input providers and receiver of result would be customers whereas the data market could additionally offer computational power.

As for the approaches based on MPC, defining the functionality is important to select a suitable scheme for homomorphic encryption based solutions. Similarly, inputs, results and involved parties need to be defined clearly. Consequently, we lay out requirements that cover these basic constraints that need to be fixed. Only after this step, one can look at schemes and protocols that efficiently implement the desired functionality.

In this deliverable, we discuss the different settings for MPC and the special cases where homomorphic encryption can be applied. In Section 2 we will discuss aspects of multiparty computation. In Section 2.1 we will introduce a somewhat more formal notion, and introduce the central security aspect – the "real-ideal" paradigm – in Section 2.2. We conclude this section with a discussion of different models in Section 2.3, whereas we distinguish them based on the roles of the involved parties. Finally, in Section 3 we will discuss homomorphic encryption schemes and their suitability for some of the presented scenarios. In Section 4 we re-state the derived requirements and conclude this deliverable.

Target Audience. This deliverable is targeted towards software engineers and researchers working with large data sets. As such we require some computer science background for the general discussion of the topics. For an excellent introduction on some of the more technical details, we refer to the book of Smart [Sma16].

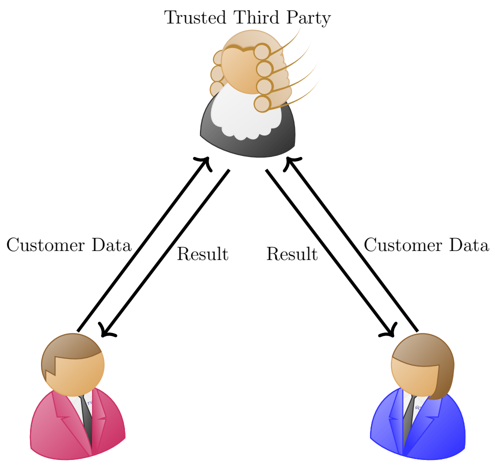
1.1 Notation

Notation-wise, we use the following conventions. We let $\kappa \in \mathbb{N}$ be the security parameter. To sample from a set S uniformly at random, we write $x \xleftarrow{R} S$. For an algorithm A, let $y \leftarrow \mathsf{A}(1^\kappa, x)$ be the process of running A, on input 1^κ and x, with access to uniformly random coins and assigning the result to y. We assume that all algorithms take 1^κ as input and we will sometimes omit to make this explicit. For an probabilistic algorithm A, we make the random coins r explicit by writing $\mathsf{A}(1^\kappa, x; r)$. An algorithm A is probabilistic polynomial time (PPT) if its running time is polynomially bounded in κ. In this case we also say that A is efficient. We use calligraphic letters, e.g., $\mathcal{A}$, for the algorithms representing adversaries in the security games. A function $f \colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ is negligible if

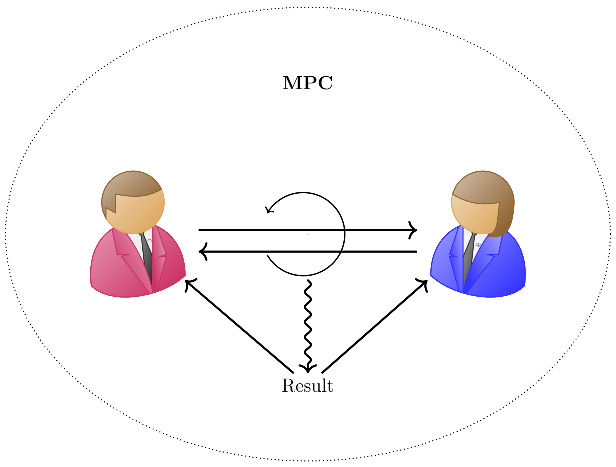
$$\forall c\, \exists \kappa_0\, \forall \kappa \geq \kappa_0 \colon f(\kappa) \leq \frac{1}{\kappa^c}.$$

2 Multi-Party Computation

The aim of MPC is to enable a group of participants $P_1, \ldots, P_n$ to jointly compute an agreed-upon function without revealing their private input $(x_1, \ldots, x_n)$ to the other participants. Once the group has agreed on a function f which they want to compute, MPC is best described in an idealized world. In this ideal world, all participating parties send their inputs to a trusted third party $\mathcal{T}$, e.g., a notary. The trusted third party then computes the result $f(x_1, \ldots, x_n)$ and sends the result to the intended receivers as depicted in Figure 1. Since in the real world there is no third party taht all participants fully trust this ideal world does not exist. The goal of MPC protocols is that they can simulate the functionality given by $\mathcal{T}$ with essentially the same security as in the ideal world.



(a) Data owners provide their data to a trusted third party. This party computes the desired functionality on the data and returns it to the data owners.



(b) Data owners perform an MPC protocol to obtain the same result without the need to involve an additional trusted party.

Figure 1: From computation of a functionality in an ideal world involving a completely trusted third party to the real world implementation without such a third party.

To implement an MPC protocol in practice, we have to first define the functionality in the ideal world. This includes the number of parties and their input, the output including who should receive the output, the security model and the performance metric. However, it requires domain knowledge of the desired functionality to complete this task. Depending on the description in the ideal world one chooses either a generic MPC protocol or a custom built protocol to perform the defined function without a trusted third party.

2.1 Definitions and Notations

Historically, cryptography was mainly concerned with concealing the content of communication against an adversary outside of the system. With the question if it would be possible to play a fair game of "mental poker", attributed to Floyd [SRA81], the paradigm changed to also consider adversaries internal to a system. Mental poker is played just like traditional poker, except that there are no cards. Instead, the game is realized (including shuffling, dealing, ...) by only sending messages between the player, but without any trusted third parties. In 1979 Shamir et al. [SRA81] examined whether it is possible to play a fair game of mental poker and presented the first scheme realizing this functionality.

In 1982, Yao [Yao82] formalized the first protocol, in which two mutually suspecting parties try to compute a function on their private input data, as for example in mental poker or coin flipping over the telephone [Blu81]. In a seminal paper in 1987, Goldreich et al. [GMW87] generalized secure computation protocols to the multi-party case. A modern way to define secure multi-party computation (MPC) following the one given by Cramer et al. [CDN15] is:

Definition 1 (Multi-Party Computation). Let $P_1, \dots, P_n$ be n parties in the computation, for $n \geq 2$. For each party P_i let X_i be the domain of input values (private) and Y_i an arbitrary set. The n parties agree on a function

$$f \colon X_1 \times \cdots \times X_n \longrightarrow Y_1 \times \cdots \times Y_n.$$

Further, let $(x_1, \dots, x_n) \in X_1 \times \cdots \times X_n$ be the private input vector of the participating parties $P_1, \dots P_n$. Then their goal is to compute $(y_1, \dots, y_n) = f(x_1, \dots, x_n)$ such that the following properties are satisfied

- Correctness: the correct value $(y_1, \dots, y_n)$ is computed
- Privacy: y_i is the only new information that the party P_i gets.

In practice and also for simplicity we often consider the case where $y_1 = \dots y_n$, i.e., there is only a single output, which is the same for every party.

Example 1. Let P_1, P_2, P_3 and P_4 be four groceries stores of the same company. They all have a certain amount $x_1, x_2, x_3, x_4 \in \mathbb{N}$ of oranges in stock. Their goal is to compute the aggregated amount of oranges without revealing the exact amount of their store. Then f can be defined in the following way

$$\begin{aligned} f : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} &\longrightarrow \mathbb{N} \\ (x_1, x_2, x_3, x_4) &\longmapsto x_1 + x_2 + x_3 + x_4. \end{aligned}$$

We can see that every store gets the new information $y_1 = \dots y_n = x_1 + x_2 + x_3 + x_4$, the total amount of oranges. But none of the stores learns the exact amount of a different store.

Back in the 1980s, MPC was merely of academic interest. This fact did not change over the following 20 years. In the late 2000s the combination of higher computing power and algorithmic improvements lead to the first large scale commercial deployment of an MPC protocol. In Denmark researchers, the government, and stakeholders launched an auction platform for sugar beets [BCD+09]. In recent years, MPC found more and more applications in practice, as illustrated in a study conducted on Estonian students, where MPC techniques made it legally possible to perform data analysis across different ministries [Cyb15].

2.2 Security of MPC

From the early days, many efforts by the cryptographic community have been devoted to finding a general definition for secure multi-party computation [Gol97]. Since the applications of MPC range from electronic voting systems to joint database computations to secure machine learning, it is difficult to come up with a definition for the security of MPC, which covers the properties of all the possible forms of MPC. For this reason, the conventional approach, known from other areas of cryptography, of writing down specific requirements for a protocol turned out to be not applicable for MPC. Instead, the real-ideal paradigm prevailed as the concept that captures the notion of security for MPC protocols best.

Definition 2 (Real-Ideal paradigm). In the ideal world all parties $P_1, \dots P_n$ send their encrypted inputs $(\mathsf{Enc}(\mathsf{pk}_T, x_1), \dots, \mathsf{Enc}(\mathsf{pk}_T, x_n))$ to a trusted third party T. Then T computes the function f on the decrypted input values

$$f(x_1, \dots, x_n) = (y_1, \dots, y_n).$$

Every party P_i receives their encrypted output $\mathsf{Enc}(\mathsf{pk}_{P_i}, y_i)$. In the real world we remove the trusted third party T. Instead all parties $P_1, \dots, P_n$ communicate with each other using a protocol π to compute f. We say π securely realizes f if one can not distinguish the output of the protocol π and the output of a corresponding simulation in the ideal world.

A more formal way of describing what security means in the context of MPC is given by Evans et al. [EKR18], and a rigorous information theoretic approach can be found in [CDN15].

2.2.1 Security Models

In the following section we discuss *semi-honest* and *malicious* security of MPC protocols. Additionally, we cover the limits for the number of corrupted parties participating in an MPC protocol whereas security can still be guaranteed.

Semi-Honest. In the semi-honest (or honest-but-curious) security model, we assume that every party $P_1, \dots, P_n$ follows the specified protocol. However, there can exist a subset $C \subset \{P_1, \dots, P_n\}$ of corrupted parties which are interested in the private input of other parties. Those corrupt parties are also called adversaries. They, for example, are allowed to collaborate by combining their input data to get information about the private input of non-corrupt parties.

The performance of semi-honest protocols is generally very good. Unfortunately, in many real-world applications, we can not guarantee that all parties follow the protocol. Hence we need a stronger security model dealing with adversaries which are not bound to the protocol.

Malicious Security. In the malicious security model, a corrupt subset $C \subset \{P_1, \dots, P_n\}$ is no longer required to follow the protocol by the letter. Corrupted parties may deviate from the protocol. In particular, a malicious party can send incorrect data to other parties or abort at any desired point in time. As in the semi-honest model, the corrupt subset can collude to violate the security of the protocol. Due to the flexibility of the adversary, the malicious protocols have to be more robust which usually translates to worse performance characteristics compared to protocol that are only secure in the semi-honest model.

Theoretical Limits Obviously, no meaningful security model can allow the number of corrupt parties equal the total number of parties. So we see that there are limits to the size of the group of corrupted parties independent of the specific protocol. More concretely, let $P_1, \dots, P_n$ be the parties participating in a given MPC protocol. Then there is a theoretical limit, depending on the security model and the computational power of the adversaries, on the number of parties whose corruption can be tolerated [BGW88; CFG+96; CCD88; GMW87]. The limits are summarized in Table 1.

	computationally unbounded	computationally bounded
semi-honest	$\leq {}^{n}/_{2}$	$\leq n-1$
malicious	$\leq {}^{n}/_{3}$	$\leq {}^{n}/_{2}$

Table 1: Limits on number of corrupted parties for MPC protocols involving n parties.

Once these limits are exceeded, there does not exist a protocol which guarantees the privacy of the inputs or the correctness of the computation. Hence, if the assumed trust model implies that the number of corrupt parties is higher than the theoretical limits performing multiparty computation can be harmful to the honest parties. In the special case of two-party computation, i.e. $n = 2$, the limits imply that in the presence of a computationally bounded malicious adversary can be tolerated. Indeed, Yao's garbled circuits can be lifted from the semi-honest setting to the malicious setting [LP07].

2.3 Basic Computational Models

In this section, we discuss different multiparty computation scenarios we denote as *basic computational models*. We will distinguish between parties that provide input data (🛢), perform computations (⚙) and receive outputs (▦). We assume that all participating parties are connected by a secure channel, i.e., the channel provides end-to-end confidentiality for example provided by TLS [Res18].

The discussion follows the classification of Bogdanov et al. [BKL+14], but we attach names to the models that make sense in the context of Safe-DEED. For each model, we also provide an example where the distribution of roles fits into the Safe-DEED context. We also provide references to applications of these models in the literature.

2.3.1 Classic

In the most basic case, all parties provide inputs, are an active part of the computation, and obtain the result in the end. Figure 2 depicts this scenario. One of the first application in this setting is Yao's millionaires' problem [Yao82]. An example of recent work includes private set intersection protocols [PSS+09; PSZ18].

Example 2. Imagine a manufacturer who wants to sell a large quantity of a product to different customers. The unit price is not fixed. Instead, it depends amongst other things on the requested quantity and wish delivery date from the customers. Hence the price

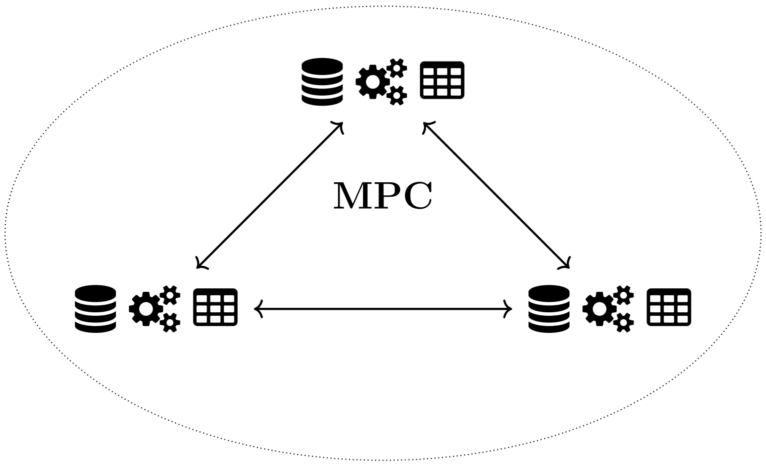


Figure 2: Classic: All parties provide input, play an active role in the protocol and receive results.

finding process is a highly interactive one, where every participant is interested in disclosing only as much information as necessary. This can be accomplished by transforming the price finding algorithm into an MPC protocol. By performing the correct MPC protocol in the end, each customer only knows his order but has not gained any information about other customers or the currently available quantity of the manufacturer. On the other hand, the manufacturer gets the order of each customer, but not the exact price range at which the customers would still be willing to buy the product.

Input: manufacturer: current quantity, possible delivery dates, price range; customer: demand, wish date, price range

Computation: customers and manufacturer

Result: manufacturer: all orders; customer: own order

2.3.2 Outsourcing

In this setting, the data owners provide the data and receive the results. The computations are performed by dedicated computations notes. Figure 4 depicts this scenario. Since the discovery of fully homomorphic encryption [Gen09b], this model is widely used, as for example by Gilad-Bachrach et al. [GDL$^+$16].

Example 3. A small-size company may have a lot of data and even an idea of how to analyze it. However, it does not possess the computing power to run data analysis algorithms. One way to solve this issue would be to outsource the computational task to a powerful server on the cloud. Naively doing this (without privacy enhancing techniques) can raise two major problems. If the data contains customer information, sending it without protection to a cloud operator could violate the GDPR and thus be illegal. On the other hand, if we are dealing with business figures, it is doubtful that the company is

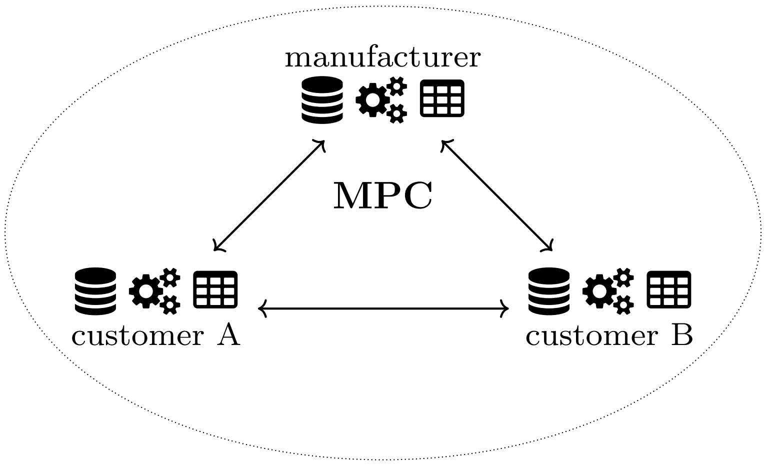


Figure 3: Classic: the manufacturer and the customers, play an active role in the protocol and receive results.

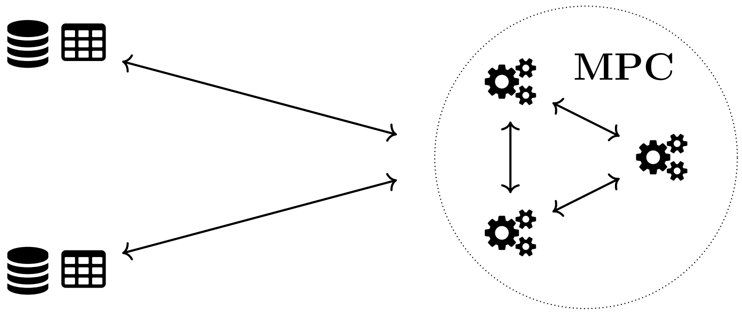


Figure 4: Outsourcing: Computations are outsourced to dedicated parties.

willing to give this sensitive data to a cloud company. To remove both constraints and therefore enable the small-size company to benefit from big data analytics, their sensitive data could be protected by means of MPC before transmitting it to the cloud. If this is done properly, the cloud operator would learn nothing about the private information of the small-size company or their customers.

Input: small-size company: data that should be analysed together with corresponding algorithm

Computing: cloud operator

Result: small-size company

2.3.3 Reporting

In this special situation, the data providers also perform the computation. The results are then received by a dedicated note. Figure 5 depicts this scenario. The studies on linked databases [BKL^{+}14] demonstrate how useful this setting can be.

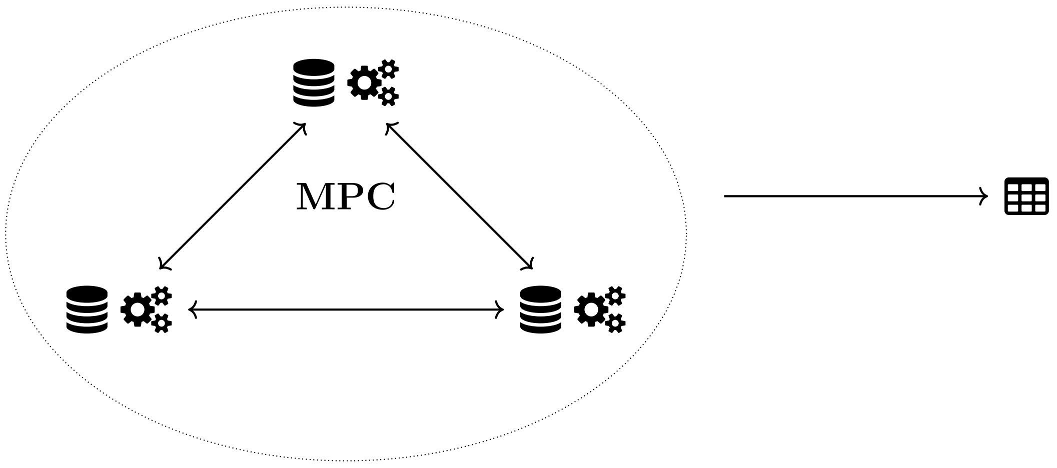


Figure 5: Reporting: Parties provide inputs and perform computation, a dedicated party receives the results.

Example 4. A federation of the industry, e.g. the Confederation of European Business, would like to perform some analysis of the business indicators of its members. A prerequisite for the analysis is that every member discloses its business metrics to the federation. Even though the different entrepreneurs decided to work together within the frame of an organization, they are still competitors and thus are afraid to share their business key figures. A way around would be that all members take part in an MPC protocol which performs the analysis of the desired business indicators. The result is then send to the federation. There is no leakage of information to any party, i.e. neither to the other members nor to the organization.

Input: members of the federation: business figures

Computing: all members of the federation

Result: federation

2.3.4 Machine Learning

In this model, every party acts as a data provider and gets results. One of the party does not perform any computation. Figure 7 depicts this scenario. Examples, where this model has been applied, include various machine learning applications [RSC$^+$19; LJL$^+$17; BIK$^+$17].

Example 5. Consider an example where an innovative data mining startup has developed its own neural network for a specific task. There are two traditional business models for the startup. First they could sell their neural network program to companies. The drawback is that a rival data mining company could buy the program and easily reverse engineer the parameters of the neural network and thus eliminate the technical advantage of the startup. To prevent this and protect the inner workings of the neural network

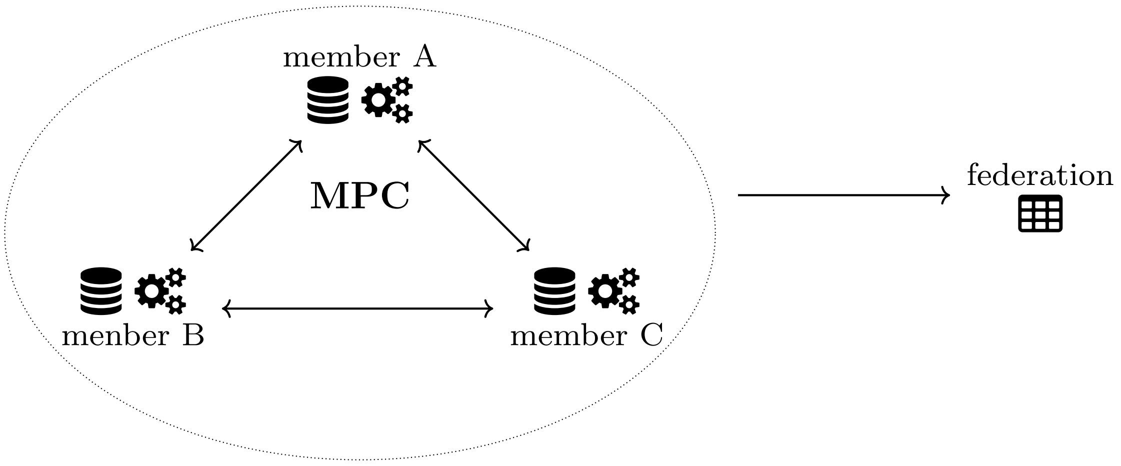


Figure 6: Reporting: three members provide input and perform computation, the federation receives the results.

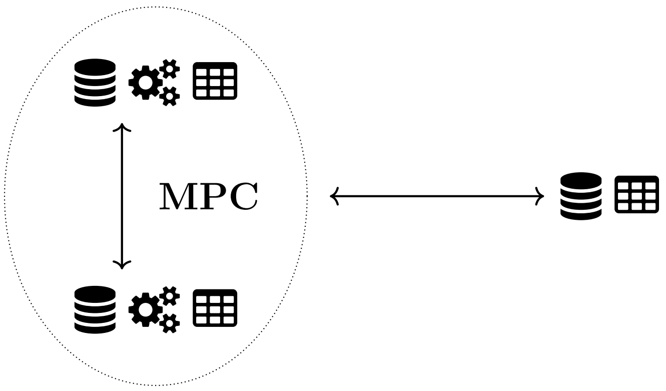


Figure 7: Machine learning: All parties provide input and receive results, a dedicated party must not perform any computation

the startup does not give away the program, instead it receives the data which should be analysed and sends the result back to the customer. However, now the customer presumably has concerns, namely with the forwarding of her data to the startup. MPC can help us to find a way out of the dilemma. By methods of cryptography the customer could transmit its data to the servers of the startup without giving away any information. In the end the customer gets the result of the computation and the startup can update its neural network.

Input: startup: neural network model; customer: data which should be analysed

Computing: startup

Result: customer: computed value; startup: updated neural network model

2.3.5 Consulting

In this case, we have a party only providing input. A dedicated party gets the results, takes part in the computation and sometimes also acts as a data provider. There are additional computing notes. Figure 8 depicts this scenario. This model was used to do financial reporting in a consortium [BTW12].

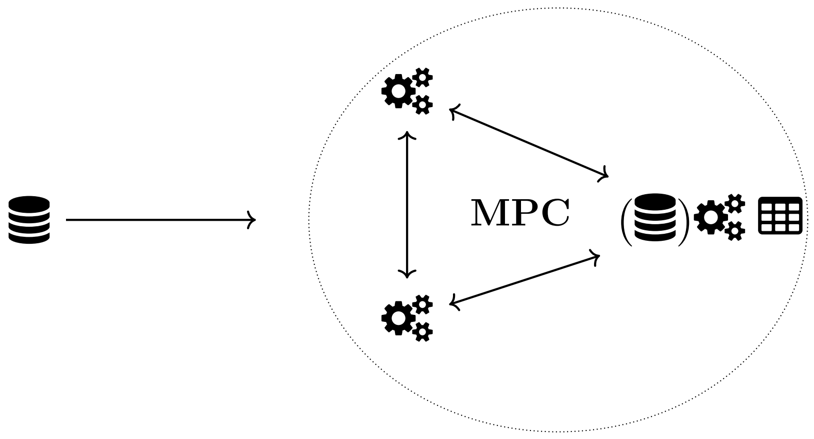


Figure 8: Consulting: A dedicated data provider, a dedicated party receives the output and is also a part of the computation, several computation notes

Example 6. One possible use-case for the consulting model is the following. A bank hires a consulting firm and in order to advice the bank properly the consulting firm has to perform data analysis on the bank's internal data. Despite a certain level of trust between the two parties, the bank understandably is not willing to disclose all its information. Now the idea could be that the consulting firm gives away the analyzing tool to the bank. The bank then internally runs the tool and reports back the results. At first glance, there is nothing wrong with this approach since it is in the sake of the bank's own interest to correctly report the results. Under further considerations, we see that the algorithms in this tool are part of the business secret of the consulting firm. Hence giving away the tool without protection is no favorable option for the consulting firm. In the ideal scenario, the analyses are performed without leaking any information besides the results. Exactly this can be accomplished if an MPC protocol is used. To guarantee the security one needs two additional parties for the computation, they have no input and receive no output.

Input: bank: internal data

Computing: consulting firm, two additionally parties

Result: consulting firm

2.3.6 Aggregation

A variant of the outsourcing scenario discussed in Section 2.3.2 is depicted in Figure 9. In this case, the parties providing the input data and the result of the computation is distinct. An example includes the aggregation of sensor data as outlined by Derler et al. [DRS17] which discuss in more detail below.

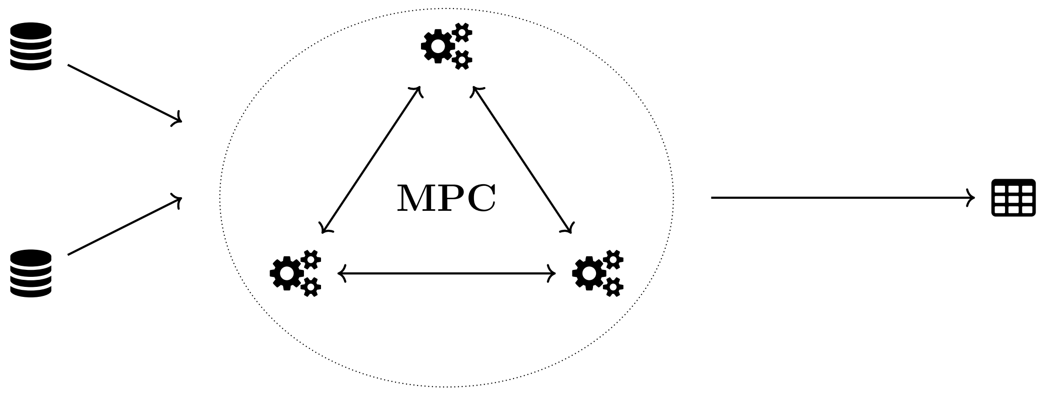


Figure 9: Aggregation: Data owners provide data to computation nodes, which provide the result to a dedicated receiver.

Example 7. Assume a setting where n senders, e.g., sensor nodes, regularly report data to some entity denoted the receiver. The avoid computation on the receiver side, it is outsourced to an aggregator. The aggregator collects the data and then reports computations on these data to the receiver. For example, consider environmental monitoring in a so-called "smart home". The sensor data is analyzed by the aggregator and then forwarded that to a device, e.g., a smartphone, controlled by the owner of the sensors.

Input: sensors: environmental data

Computation: cloud providers

Result: data owner: aggregated data and its analysis

3 Homomorphic Encryption

Public-key encryption schemes usually ensure that any modification of the ciphertext is detected. To securely perform computations on encrypted data, this restriction needs to be lifted to some extent. In particular, we want to enable the computation of a class of functions by evaluating them on ciphertexts. Thereby it is possible to outsource computations to a third party, e.g. a cloud service. Shortly after the publication of RSA [RSA78], the possibility to compute on encrypted data was deemed interesting for applications [RAD+78]. In this case however – as long as the encryption scheme is secure – no additional trust assumptions have to be placed on the third party to keep the encrypted data confidential.

The class of public-key encryption schemes that allow others to perform computation on the encryption plaintext is called *homomorphic encryption*. To perform computations in this context, only the ciphertexts need to be known. Depending on the type of computable function, schemes can be classified as *linearly* homomorphic,[1] *multiplicatively* homomorphic, *fully* homomorphic (i.e., linearly and multiplicatively homomorphic), or *somewhat* homomorphic (i.e., fully homomorphic for a limited number of operations).

However, we note that homomorphic encryption schemes only enable the evaluation of functions on ciphertexts that were encrypted with respect to the same public key. As a consequence, the resulting ciphertext is also encrypted with respect to the same public key. Thus, homomorphic encryption directly used as a basis to obtain secure computation on data of multiple owners only applies to a certain scenario. All of the data owners contribute their data, and a designated party receives the result, and the computation itself is outsourced to a computation party. In that case, the data owners would encrypt their data with respect to the public key of the receiver. As long as the designated party does not obtain the individual ciphertexts, but only obtains the encrypted result, the confidentiality of the inputs is still guaranteed. However, if the designated receiver would obtain any of the input ciphertexts, they could be decrypted as well.

3.1 Definitions and Notations

A $\mathcal{F}$-homomorphic public key encryption (HPKE) scheme is a public-key encryption scheme that is homomorphic with respect to a class of functions $\mathcal{F}$, i.e., given a sequence of ciphertexts to messages $(m_i)_{i\in[n]}$ one can evaluate a function $f\colon \mathcal{M}^n \to \mathcal{M} \in \mathcal{F}$ on the ciphertexts such that the resulting ciphertext decrypts to $f(m_1,\ldots,m_n)$. More formally, we define it as follows:

Definition 3 (($\mathcal{F}$-)HPKE). A $\mathcal{F}$-homomorphic public key encryption ($\mathcal{F}$-HPKE or HPKE for short) scheme with message space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and a function family $\mathcal{F}$ consists of the PPT algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$:

$\mathsf{Gen}(1^\kappa)$: On input security parameter κ, outputs public and secret keys $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, m)$: On input a public key pk, and a message $m \in \mathcal{M}$, outputs a ciphertext $c \in \mathcal{C}$.

$\mathsf{Dec}(\mathsf{sk}, c)$: On input a secret key sk, and ciphertext c, outputs a message $m \in \mathcal{M}$ or, in case of a failure, the error symbol $\perp$.

$\mathsf{Eval}(f, (c_i)_{i\in[n]})$: On input a function $f\colon \mathcal{M}^n \to \mathcal{M} \in \mathcal{F}$, a sequence of ciphertexts $(c_i)_{i\in[n]}$ encrypted under the same public key, outputs a ciphertext c.

For correctness we require the standard correctness of public-key encryption schemes: if for all $\kappa \in \mathbb{N}$, for all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\kappa)$ and for all $m \in \mathcal{M}$ it holds that

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m] = 1,$$

then HPKE is perfectly correct. More importantly, we further require for the correctness of HPKE that for all security parameters $\kappa \in \mathbb{N}$, all key pairs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\kappa)$, all

[1]We want to note, that for linearly homomorphic schemes with a ring structure on the message space, transformations to somewhat homomorphic schemes exist [CF15].

functions $f\colon \mathcal{M}^n \to \mathcal{M} \in \mathcal{F}$, all message sequences $(m_i)_{i\in[n]}$ it holds that

$$\Pr\Big[\mathsf{Dec}\Big(\mathsf{sk}, \mathsf{Eval}\Big(f, (\mathsf{Enc}(\mathsf{pk}, m_i))_{i\in[n]}\Big)\Big) = f(m_1, \ldots, m_n)\Big] = 1.$$

For the security of homomorphic public-key encryption schemes, we require the typical indistinguishability security notions of public-key encryption. This notion ensures that the ciphertexts do not leak any information on the encrypted plaintexts. The goal is to achieve the same security notion also in the case of homomorphic encryption, i.e., adding the Eval algorithm does not change anything regarding the security of a scheme. More formally, indistinguishability under chosen message attacks (IND-CPA security) requires that an adversary $\mathcal{A}$ cannot decide which message is actually contained in a ciphertext c even when allowed to choose two challenge messages m_0 and m_1.

Definition 4 (IND-CPA). For a PPT adversary $\mathcal{A}$, we define the advantage function in the sense of indistinguishability under chosen message attacks (IND-CPA) as

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A},\mathsf{HPKE}}(\kappa) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A},\mathsf{HPKE}}(\kappa) = 1\right] - \frac{1}{2}\right|,$$

where the corresponding experiment is depicted in Experiment 1. If for all PPT adversaries $\mathcal{A}$ there is a negligible function $\varepsilon(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A},\mathsf{HPKE}}(\kappa) \leq \varepsilon(\kappa),$$

then HPKE is IND-CPA secure.

```
$\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\mathcal{A},\mathsf{HPKE}}(\kappa)$:
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$
    $b \xleftarrow{R} \{0, 1\}$
    $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pk})$
    $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
    $b^* \leftarrow \mathcal{A}(c^*, \mathsf{st})$
    return 1, if $b^* = b$
    return 0
```

Experiment 1: IND-CPA security experiment for (homomorphic) public-key encryption.

If a stronger notion of indistinguishability is required, the adversary can be given access to the decryption oracle before selecting the challenge ciphertext. That is, we can define the first variant of indistinguishability under chosen ciphertext attacks, IND-CCA1, also for homomorphic encryption:

Definition 5 (IND-CCA1). For a PPT adversary $\mathcal{A}$, we define the advantage function in the sense of indistinguishability under chosen message attacks (IND-CCA1) as

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca1}}_{\mathcal{A},\mathsf{HPKE}}(\kappa) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cca1}}_{\mathcal{A},\mathsf{HPKE}}(\kappa) = 1\right] - \frac{1}{2}\right|,$$

where the corresponding experiment is depicted in Experiment 2. If for all PPT adversaries $\mathcal{A}$ there is a is a negligible function $\varepsilon(\cdot)$ such that

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A},\mathsf{HPKE}}(\kappa) \leq \varepsilon(\kappa),$$

then HPKE is IND-CCA1 secure.

```
$\mathsf{Exp}^{\text{ind-cca1}}_{\mathcal{A},\mathsf{HPKE}}(\kappa)$:
    $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$
    $b \xleftarrow{R} \{0, 1\}$
    $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot))}(\mathsf{pk})$
    $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
    $b^* \leftarrow \mathcal{A}(c^*, \mathsf{st})$
    return 1, if $b^* = b$
    return 0
```

Experiment 2: IND-CCA1 security experiment for (homomorphic) public-key encryption.

We, however, want to note, that homomorphic encryption schemes can not achieve the stronger variant of indistinguishability under chosen ciphertext attack where the adversary can also query the decryption oracle after obtaining the challenge ciphertext.

Additionally, we want the scheme to provide circuit privacy, i.e want to ensure that fresh encryptions and evaluations are identically distributed. More precisely:

Definition 6 (Circuit Privacy). HPKE provides circuit privacy if the distributions of the sets

$$\{\mathsf{Enc}(\mathsf{pk}, f(m_1, \ldots, m_n)) \mid m_1, \ldots, m_n \in \mathcal{M}\}$$

and

$$\left\{\mathsf{Eval}\Big(f, (\mathsf{Enc}(\mathsf{pk}, m_i))_{i\in[n]}\Big) \mid m_1, \ldots, m_n \in \mathcal{M}\right\}$$

for all $f \in \mathcal{F}$ and all pk are indistinguishable.

This notion ensures that ciphertexts do not reveal that they where homomorphically evaluated. In other words, ciphertexts output by the Eval algorithm look exactly the same as if the ciphertexts were directly encrypted with Enc. This notion is also paramount to distinguish between trivial and useful homomorphic encryption schemes. In a trivial homomorphic scheme, one could simply combine all ciphertexts and when decrypting, one then could simply evaluate the function after decryption. However, with such a scheme, we would loose the ability to guarantee any security notions with respect to the initially encrypted values.

Below we discuss some examples of homomorphic encryption schemes. We start with a short overview of linearly and multiplicatively homomorphic schemes. Such schemes are well known for more than four decades. Indeed, textbook RSA encryption [RSA78] enjoys multiplicatively homomorphic properties but does not provide IND-CPA security.

In the 80s, Goldwasser and Micali proposed a comparatively inefficient, but IND-CPA-secure and the first linearly homomorphic scheme [GM82]. One of the most notable schemes in factoring setting is the scheme due to Paillier [Pai99]. Besides being linearly homomorphic and IND-CPA-secure, the Paillier public-key encryption scheme also allows the homomorphic multiplication with a plaintext message very efficiently.[2]

For encryption schemes in the discrete logarithm setting, the plain ElGamal public-key encryption scheme [Gam84] enjoys multiplicatively homomorphic properties and is also IND-CPA secure under the decisional Diffie-Hellman assumption. Note though, that in this case, the message space consists of group elements, and thus might not be suitable for some applications. A variant dubbed linear ElGamal [BBS04] provides a linearly homomorphic scheme. The message space for this scheme is $\mathbb{Z}$ modulus the group order, however, it needs to map to the group for encryption. On decryption, one obtains the message encoded in the group and has to reverse the encoding. Thus decryption becomes tricky unless the actually used message space is reduced to a polynomial size.

The concept of fully homomorphic encryption was known since the 70s, but only since Gentry's seminal PhD thesis [Gen09a] we know of such a scheme based on hard problems over lattices.[3] The basic idea is the following: first one constructs a somewhat homomorphic encryption scheme allowing one to perform a limited number of homomorphic operations, i.e. many additions, but only a few multiplication. Then, one repeatedly refreshes the ciphertexts to allow unlimited operations in a process called bootstrapping. Also, approaches such as TFHE [CGG$^+$18], which was presented only recently, or FHWE [DM15] follow this general paradigm.

3.2 Applications of Homomorphic Encryption to Multi-Party Computation

As discussed before, homomorphic encryption is applicable if – in the most basic setting – the scenario only consists of one receiver of the results and the computation nodes are distinct from the receiver. In other words, we can also assume that the computation nodes and the receiver do not collude. For the models outlined in Section 2.3, this means that homomorphic encryption can be used in the cases of outsourcing computation (Section 2.3.2), machine learning (Section 2.3.4), and aggregation (Section 2.3.6), in principal. In these special settings, homomorphic encryption provides an important alternative approach to implement a protocol for computation on shared data securely. We also want to note, that such approaches can have advantages when bandwidth is limited as the encrypted data only needs to be sent from the input data providers to the computation nodes, and finally to the receiver. Additionally, if non-collusion between the receiving party and the computation nodes can be ensured, the amount of computation nodes can be reduced and the theoretical limits discussed in Section 2.2 can be avoided.

[2]Such functionality is always possible given a linearly homomorphic scheme with a double-and-add-style algorithm. However, its efficiently scales with the size of the plaintext coefficient. Paillier encryption allows one to perform this computation with one exponentiation $\mod N^2$ instead of logarithmically many.

[3]See also [Gen09b] for a preliminary version and [Gen10] for a high-level overview.

More generally, besides secret-sharing based and other approaches, homomorphic encryption is also used to build MPC protocols. One notable example of a generic purpose MPC library is `SCALE-MAMBA`[4] which, among others, is based on BDOZ [BDO+11] and SPDZ [DPS+12].

4 Conclusion

For the exploitation of the valuable resource "Big Data" in a sustainable way, it is inevitable that we develop technology that can handle the associated privacy and confidentiality threats. Secure multiparty computation (Section 2) and homomorphic encryption (Section 3) are the two concepts, which are widely regarded as the most promising to deal with the above-mentioned concerns. It is crucial to notice that multiparty computation and homomorphic encryption must be seen less as competing than complementary approaches. The theoretical results for both concepts developed so far guarantee us the mere existence of privacy preserving protocols. The challenge now is given a specific functionality to find a sufficiently fast protocol, so that it can be used in a real-world application.

In order to successfully develop an efficient and secure protocol, it is of great importance to rigorously define the desired functionality. This not only includes the involved parties respectively their input and output but also their role in the computational task. Only when these parameters are fixed, it makes sense to work on the concrete protocol.

Concerning, the selection of a model defined in Section 2.3, both legal requirements and the trust relationship between the parties must be considered. Once we have fixed the security model, we can concentrate on the right choice of a basic model. It has to be said that the enumeration of models in Section 2.3 is non-exhaustive. This means that we may have to define a new model or combine two or more of them. However, the presented models exemplify the possible relations between the parties that have been observed in the literature and practice.

To sum it up, the following prerequisites need to be checked before investigating concrete protocols:

- Define the desired functionality including inputs and outputs. Check that the functionality can be evaluated in the "ideal" world. That is, check whether – assuming the desired input data is available – a trusted third party would be able to compute the result of the function.
- Define the involved parties including which parties provide input, receive the results of the computation, and which parties actively contribute to the computation. Check which of the basic models applies to the involved parties and their roles.
- Check whether all parties are able to communicate securely via end-to-end encrypted channels.

[4] `https://homes.esat.kuleuven.be/~nsmart/SCALE/`

- Define how many parties might be compromised or would potentially be interested in colluding to gain an advantage. Check whether security against malicious adversaries is required or if semi-honest security suffices.

After defining the required key components and checking the requirements as outlined above, the investigation of concrete protocols can commence.

References

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004. URL: `https://doi.org/10.1007/978-3-540-28628-8%5C_3`.

[BCD+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009. URL: `https://doi.org/10.1007/978-3-642-03549-4%5C_20`.

[BDO+11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2011. URL: `https://doi.org/10.1007/978-3-642-20465-4%5C_11`.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988. URL: `https://doi.org/10.1145/62212.62213`.

[BIK+17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1175–1191. ACM, 2017. URL: `https://doi.org/10.1145/3133956.3133982`.

[BKL$^+$14] Dan Bogdanov, Liina Kamm, Sven Laur, Pille Pruulmann-Vengerfeldt, Riivo Talviste, and Jan Willemson. Privacy-preserving statistical data analysis on federated databases. In Bart Preneel and Demosthenes Ikonomou, editors, *Privacy Technologies and Policy - Second Annual Privacy Forum, APF 2014, Athens, Greece, May 20-21, 2014. Proceedings*, volume 8450 of *Lecture Notes in Computer Science*, pages 30–55. Springer, 2014. URL: `https://doi.org/10.1007/978-3-319-06749-0%5C_3`.

[Blu81] Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *Advances in Cryptology: A Report on CRYPTO 81, CRYPTO 81, IEEE Workshop on Communications Security, Santa Barbara, California, USA, August 24-26, 1981.* Pages 11–15. U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04, 1981.

[BTW12] Dan Bogdanov, Riivo Talviste, and Jan Willemson. Deploying secure multi-party computation for financial data analysis - (short paper). In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security - 16th International Conference, FC 2012, Kralendijk, Bonaire, Februray 27-March 2, 2012, Revised Selected Papers*, volume 7397 of *Lecture Notes in Computer Science*, pages 57–64. Springer, 2012. URL: `https://doi.org/10.1007/978-3-642-32946-3%5C_5`.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988. URL: `https://doi.org/10.1145/62212.62214`.

[CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. ISBN: 9781107043053. URL: `http://www.cambridge.org/de/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure-multiparty-computation-and-secret-sharing?format=HB%5C&isbn=9781107043053`.

[CF15] Dario Catalano and Dario Fiore. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1518–1529. ACM, 2015. URL: `https://doi.org/10.1145/2810103.2813624`.

[CFG$^+$96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 639–648. ACM, 1996. URL: `https://doi.org/10.1145/237814.238015`.

[CGG+18] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: fast fully homomorphic encryption over the torus. *IACR Cryptology ePrint Archive*, 2018:421, 2018. URL: `https://eprint.iacr.org/2018/421`.

[Cyb15] Cybernetica. Track big data between government and education, 2015. URL: `https://sharemind.cyber.ee/big-data-analytics-protection/`.

[DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, 2015. URL: `https://doi.org/10.1007/978-3-662-46800-5%5C_24`.

[DPS+12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012. URL: `https://doi.org/10.1007/978-3-642-32009-5%5C_38`.

[DRS17] David Derler, Sebastian Ramacher, and Daniel Slamanig. Homomorphic proxy re-authenticators and applications to verifiable multi-user data aggregation. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, volume 10322 of *Lecture Notes in Computer Science*, pages 124–142. Springer, 2017. URL: `https://doi.org/10.1007/978-3-319-70972-7%5C_7`.

[Eco17] The Economist. The world's most valuable resource is no longer oil, but data. 2017. URL: `https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data` (visited on May 2, 2019).

[EKR18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018. URL: `https://doi.org/10.1561/3300000019`.

[Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984. URL: `https://doi.org/10.1007/3-540-39568-7%5C_2`.

[GDL+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. Cryptonets: applying neural networks to encrypted data with high throughput and accuracy. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 201–210. JMLR.org, 2016. URL: `http://jmlr.org/proceedings/papers/v48/gilad-bachrach16.html`.

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. URL: `https://crypto.stanford.edu/craig`.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009. URL: `https://doi.org/10.1145/1536414.1536440`.

[Gen10] Craig Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, 2010. URL: `https://doi.org/10.1145/1666420.1666444`.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982. URL: `https://doi.org/10.1145/800070.802212`.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987. URL: `https://doi.org/10.1145/28395.28420`.

[Gol97] Shafi Goldwasser. Multi-party computations: past and present. In James E. Burns and Hagit Attiya, editors, *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997*, pages 1–6. ACM, 1997. URL: `https://doi.org/10.1145/259380.259405`.

[LJL+17] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 619–631. ACM, 2017. URL: `https://doi.org/10.1145/3133956.3134056`.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer, 2007. URL: `https://doi.org/10.1007/978-3-540-72540-4%5C_4`.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999. URL: `https://doi.org/10.1007/3-540-48910-X%5C_16`.

[PSS+09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 250–267. Springer, 2009. URL: `https://doi.org/10.1007/978-3-642-10366-7%5C_15`.

[PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur.*, 21(2):7:1–7:35, 2018. URL: `https://doi.org/10.1145/3154794`.

[RAD+78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[Res18] Eric Rescorla. The transport layer security (TLS) protocol version 1.3. *RFC*, 8446:1–160, 2018. URL: `https://doi.org/10.17487/RFC8446`.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. URL: `http://doi.acm.org/10.1145/359340.359342`.

[RSC+19] M. Sadegh Riazi, Mohammad Samragh, Hao Chen, Kim Laine, Kristin E. Lauter, and Farinaz Koushanfar. XONN: xnor-based oblivious deep neural network inference. *IACR Cryptology ePrint Archive*, 2019:171, 2019. URL: `https://eprint.iacr.org/2019/171`.

[Sma16] Nigel P. Smart. *Cryptography Made Simple*. Information Security and Cryptography. Springer, 2016. ISBN: 978-3-319-21935-6. URL: `https://doi.org/10.1007/978-3-319-21936-3`.

[SRA81] Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman. Mental poker. In *The mathematical gardner*, pages 37–43. Springer, 1981.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982. URL: `https://doi.org/10.1109/SFCS.1982.38`.