D2.1 State of the Art Analysis of MPC-Based Big Data Analytics

Peter S. Nordholt (ALX), Nikolaj Volgushev (ALX), Mark Abspoel (PHI), Meilof Veeningen (PHI), Frank Blom (TUE), Niek J. Bouman (TUE), Mykola Pechenizkiy (TUE)

The project SODA has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731583.

Project Information

Scalable Oblivious Data Analytics

Project number:	731583
Strategic objective:	H2020-ICT-2016-1
Starting date:	2017-01-01
Ending date:	2019-12-31
Website:	https://www.soda-project.eu/

Document Information

Title: D2.1 State of the Art Analysis of MPC-Based Big Data Analytics

ID:	D2.1	Type:	R	Dissemination level:	PU
		Month:	M9	Release date:	September 29, 2017

Contributors, Editor & Reviewer Information

Contributors (person(partner): sections)	Peter S. Nordholt (ALX) Nikolaj Volgushev (ALX) Mark Abspoel (PHI) Meilof Veeningen (PHI) Frank Blom (TUE) Niek J. Bouman (TUE) Mykola Pechenizkiy (TUE)
Editor (person/partner)	Frank Blom (TUE), Niek J. Bouman (TUE)
Reviewer (person/partner)	Prastudy Fauzi (AU), Claudio Orlandi (AU), Peter Scholl (AU), Mark Simkin (AU), Meilof Veeningen (PHI)

Release	Date issued	Release description / changes made
1.0	September 29, 2017	First release to EU

SODA Consortium

Full Name	Abbreviated Name	Country
Philips Electronics Nederland B.V.	PHI	Netherlands
Alexandra Institute	ALX	Denmark
Aarhus University	AU	Denmark
Göttingen University	GU	Germany
Eindhoven University of Technology	TUE	Netherlands

Table 1: Consortium Members

Executive Summary

The objective of work package 2 is to improve the state of the art in privacy-preserving data mining protocols for use with big data, focusing on distributed and streaming algorithms. This deliverable provides a background on privacy-preserving data mining primitives, (plain-text) data mining, streaming algorithms and surveys previous works on privacy preserving data mining. Chapter 2 discusses preliminaries and primitives required for privacy preserving computation. We focus on primitives which are used as building blocks in privacy-preserving data mining, such as secure comparison, secure shuffling, and private set intersection. Chapter 3 provides a background on the field of machine learning in the plain-text setting. Chapters 4 and 5 discuss data streams, which are relevant especially in the context of big data analysis. Chapter 4 focuses on the evolving nature of data streams, introduces the notion of concept drift, and discuss adaptive strategies to deal with those, while Chapter 5 gives some history and a theoretical introduction into some classical data stream problems. The inclusion of streaming algorithms within the SODA project originates from their seeming absence in privacy-preserving data-mining literature. Chapter 6 of this state-of-the-art deliverable gives an overview of prominent existing solutions of MPC-based privacy-preserving data mining. From this literature study we confirm our suspicion regarding the lack of works on *privacy-preserving data mining in the streaming setting*. So we believe there is sufficient opportunity for further research.

About this Document

Role of the Deliverable

This deliverable contains a state of the art analysis in MPC-based privacy-preserving data mining protocols for use with Big Data.

Relationship to other SODA Deliverables

This state-of-the-art deliverable will serve as a basis for further research on both general- and special-purpose protocols in MPC-based privacy-preserving data mining, which will be presented in deliverable D2.2 and D2.3 respectively. Deliverable D1.1 also contains a state of the art analysis but focuses on theory and practice of secure multi-party computation, including a survey on oblivious RAM and oblivious data structures that are used in privacy-preserving data mining.

Structure of this Document

Chapter 1 provides an introduction to this document. Chapter 2 introduces secure computation of primitives required for machine learning functionality. Chapter 3 discusses the state of the art of data mining and machine learning without the notion of security or privacy. Chapter 4 and Chapter 5 introduce the streaming setting data mining. Chapter 6 reviews state of the art protocols used in privacy preserving data mining.

Table of Contents

1 Introduction

It is estimated that by 2020 we will have more than 16 zettabytes ($1.6 \cdot 10^{13}$ GB) of useful data in the world [98]. To mine such vast quantities of data for valuable information effectively, data mining strategies specifically designed to handle *Big Data* are of great importance. Over the past decades, research in data mining and machine learning has yielded many useful algorithms for this task. While these algorithms are usually designed to run on cleartext data, in practice many data sets are highly privacy-sensitive and/or confidential. Genomic records, for example, could reveal whether an individual carries certain diseases. When data is shared outside an organization (e.g., to run some machine learning algorithm in the cloud) data is typically anonymized, to prevent unwanted leakage of privacy-sensitive and confidential information. Unfortunately, it turns out to be extremely tricky to properly anonymize data while preserving utility [19]. More than once, researchers have succeeded in de-anonymizing "anonymized" data after it was published [199, 23, 79]. While these events boosted the formal study of anonymization (research into methods like *k-anonymity* and *differential privacy*), there are alternatives to anonymization for protecting the privacy or confidentiality of data.

In the Work Package associated to the present deliverable (SODA WP2), we focus on a class of data mining methods that leverage techniques from the field of cryptography, like *secure multiparty computation* (MPC), to keep their data secure. In MPC, a number of parties can jointly perform a computation on data (provided by some or all the parties), such that each party learns nothing beyond what can be deduced from the output of the computation. MPC offers very strong privacy and confidentiality properties.

In 2002, Lindell and Pinkas [170] were among the first to discuss the feasibility of privacy-preserving data mining by means of secure multi-party computation. In particular, the authors focus on a secure (i.e., privacy-preserving) version of the ID3 decision tree construction algorithm in the two-party setting. Since then, many works have appeared on the cryptographic approach to privacy-preserving data mining (too many to cover them all), of significantly varying quality. Most of these works are based on one of the following paradigms within MPC, namely *homomorphic encryption*, *garbled circuits* and *secret sharing*, which each have their own use case (number of parties, network latency and bandwidth vs. computational complexity) and come with different practical pros and cons.

In the present document, we attempt to give an overview of the state of the art of MPC-based privacy-preserving data mining. We first introduce the field of data mining and machine learning as well as relevant MPC primitives, after which we present a selection of papers from the literature that seem relevant for discussion and could serve as a foundation for further research on the subject. Each such paper is accompanied by a brief review. One motivation for the SODA project was the seeming lack of works of *secure data mining in the streaming setting*. Our literature study indeed further confirms this gap in the state of the art; hence, we put special emphasis on introducing the topic of streaming data mining in this document, and we remain convinced that *secure* streaming data mining provides ample opportunities for further study.

The document is structured as follows. Chapter 2 introduces secure computation of primitives required for machine learning functionality, such as arithmetic, comparison and set intersection. Here we focus on previous work on such primitives in the secure-computation setting. To combine the knowledge of MPC to the field of data mining, Chapter 3 discusses a state of the art of data mining and machine learning without the notion of security or privacy. Chapters 4 and 5 present a basic overview of data streams. The techniques applied in this field are promising with regards to mining 'Big' data. Streaming algorithms are tailored to limit the memory footprint, assuming that the data does not fit in memory. These ideas originate from hardware constraints in the beginning of the late 1970s. Such notions are relevant once again in the age of Big data, for which it is infeasible to keep

everything in memory. We proceed by reviewing privacy preserving data mining in Chapter 6. The chapter starts with an overview of privacy preserving data mining in general, and the role of secure multi-party computation therein. We continue by focusing on prominent papers in the field of privacy preserving data mining and machine learning by means of secure multi-party computation. For each paper we present a short summary including their main goal(s), technique(s), and trade-offs involved.

2 Secure Multiparty Computation: Preliminaries and Primitives

In this section we discuss several aspects and primitives from MPC that are relevant in the context of data mining algorithms: basic arithmetic, linear algebra, and important primitives like comparison, shuffling and sorting, and higher-level primitives like private set intersection. This section aims to be complementary to the material presented in SODA Deliverable D1.1; most basic MPC primitives are explained in more detail in that document.

2.1 Secure Arithmetic

Essentially all multiparty computation paradigms (based on arithmetic secret sharing, garbled circuits, or homomorphic encryption) offer integer arithmetic in some finite ring or finite field.[1] Instead, many practical applications, such as data-mining algorithms, require arithmetic over the integers, rationals, or reals (in fact, approximations thereof, like fixed-point or floating-point numbers). Hence, an important question in secure data mining is how to *emulate* arithmetic over the integers, rationals, or reals using the ring or field operations provided by the MPC framework.

2.1.1 Arithmetic over the Integers

The ring of integers modulo m, where m is an arbitrary positive integer immediately gives "integer arithmetic bounded by m", by which we mean that one can perform additions and multiplications as long as all operands remain strictly smaller than m, to prevent reductions modulo m (called *"wrap-arounds"*). Wrap-arounds should be prevented from happening, because after a wrap-around the one-to-one correspondence between integer-arithmetic and modular arithmetic is lost. Addition of any nonzero k-bit and any ℓ-bit number with $k \geq \ell$ gives an n-bit result with $n \in \{k, k+1\}$. Multiplication of any nonzero k'-bit and any nonzero ℓ'-bit number gives an n'-bit result with $n' \in \{k'+\ell'-1, k'+\ell'\}$. This implies that, for example, in an algorithm that performs repeated multiplications, the bit-length of the multiplicand grows rapidly: linearly or even exponentially. On the other hand, for practical reasons (performance-related) m cannot be chosen arbitrarily large (it is limited to, say, a few thousands of bits). Hence, the goal of avoiding wrap-arounds might impose limits elsewhere, e.g., on the number of iterations of an algorithm that may be executed.

If the modulus m is prime, then the finite ring is in fact a field, and in this case exact divisions can be performed by multiplying with the modular inverse of the divisor. Non-exact divisions are much more involved; they form the basis for fixed-point arithmetic and will be covered in Section 2.1.3. Some algorithms exploit knowledge about the presence of multiplicative factors, which can then be divided out, in order to limit coefficient-growth. An example is Jordan's fraction-free Gaussian elimination algorithm [24].

2.1.2 Rational Arithmetic

A generalization of integer arithmetic is to mimic secure exact rational arithmetic by maintaining two secret integers, one for the numerator and one for the denominator.

Although the arithmetic operations for the secret-rational-number type are straightforward to define in terms of the operations for the secret integers, the issue of rapid coefficient-growth is also

[1] For an introduction into finite rings and fields, the reader is referred to Serge Lang's algebra textbook [163].

present here. Furthermore, reducing fractions by removing common factors (common to the numerator and denominator) involves divisibility tests, which are expensive operations in secure computation, unlike in the non-secret setting.

The coefficient-growth issue could be avoided by using fixed-point arithmetic (see below), at the expense of introducing round-off errors and an increased computational and round complexity.

2.1.3 Fixed-Point Arithmetic

Protocols for secure fixed-point arithmetic were introduced in a series of papers by Catrina et al. [50, 51, 49]. In their representation, a tuple $(x, f) \in \mathbb{Z}^2$ represents the value $x \cdot 2^{-f} \in \mathbb{Q}$, where 2^{-f} is called the *precision*. Addition and multiplication are defined for any two inputs that have the same precision, and are achieved by ordinary secure integer addition and multiplication *followed by a truncation*, in order to lower the precision of the result towards that of the inputs. An important choice with respect to the selection of the truncation protocol is whether the least significant bit of the truncated result must be correct (deterministic rounding) or is allowed to be random (probabilistic rounding). In the former case, truncation involves secret comparison and is rather expensive in terms of round complexity, whereas in the latter case, the truncation procedure is cheaper but the rounding error will accumulate over multiple arithmetic operations. Catrina et al. also propose secure division protocols, based on Newton–Raphson and Goldschmidt methods [50, 51, 49].

2.1.4 Floating-Point Arithmetic

By now, several ways to achieve secure floating-point arithmetic have been proposed in the literature. Franz et al. [108] consider the two-party setting and propose a logarithm-based representation of numbers, which makes multiplication and division rather easy, while renders addition and subtraction into more involved operations. This approach uses a table-based design which means that the bit-size for which the approach is usable is limited (up to ≈ 20 bits, according to the authors). In his doctoral thesis [107], Franz also proposes protocols (for the 2-party case) for dealing with floating-point arithmetic resembling the IEEE 754 standard [132], using garbled circuits.

Aliasgari et al. [12] propose a floating-point representation and associated protocols for arithmetic operations for secret-sharing-based multiparty computation. In their approach, the four-tuple $(f, e, s, z) \in [2^{\ell-1}, 2^{\ell}] \times (-2^{k-1}, 2^{k-1}) \times \{0,1\} \times \{0,1\}$ represents the value $(1-z)(-1)^s f^e \in \mathbb{R}$. Note that zero is encoded explicitly using the bit z. Beyond supporting the basic arithmetic operations, they propose protocols for logarithm, square root and exponentiation. Subsequent work of Aliasgari et al. [11] further improves the exponentiation operation.

Liu et al. [175] as well as Kamm and Willemson [143] propose protocols for a floating-point representation similar to the one above but without an explicit zero bit. Liu et al. focus on the two-party case. Kamm and Willemson target focus on the multi-party case and, in particular, use polynomial evaluation to approximate several non-linear functions. Krips and Willemson [160] propose an optimization to mix floating-point with fixed-point arithmetic to achieve a better trade-off between precision and speed when computing elementary functions.

Pullonen and Siim [218] combine secret sharing and garbled circuits to achieve secure floating-point arithmetic for the *multi*-party case. Two non-colluding coalitions of players execute the garbler and evaluator part of the garbled-circuit computation. In particular, they achieve a fully compliant secure IEEE 754 floating-point implementation by compiling an existing IEEE 754 soft-float implementation into a circuit, which can then be garbled.

Dimitrov et al. [80] propose alternative approaches to fixed-point and floating-point numbers, respectively, namely *Golden numbers* (based on the golden ratio) and a logarithm-based representation.

2.2 Secure Linear Algebra

Linear algebra is the basis for many applied problems, such as optimization, machine learning, and network analysis. Solving a linear system of equations $Ax = b$ is one of the fundamental primitives in linear algebra. For the case where this equation is to be interpreted over a finite field, Cramer and Dåmgard [65] provide a protocol which accomplishes this for any A and b arbitrarily secret-shared between any number of parties. Their protocol takes a constant number of rounds and requires roughly ℓ^5 secure multiplications[66], if $A \in K^{k\times\ell}$ with $k \leq \ell$. It works except for with negligible probability, assuming the field K is large (specifically, that $\ell \ll |K|$). In their construction, they directly use generic MPC protocols for secure addition, multiplication, and sharing a random element as primitives without making any cryptographic assumptions. Hence their construction is unconditionally secure (or secure against an active adversary, respectively), if the underlying primitives are. Cramer, Kiltz and Padró [66] give an improvement that only requires $k^4 + \ell^2 k$ secure multiplications.

In the case of two semi-honest parties in the computational model, various protocols that solve linear equations, over finite fields, exist. Most use either garbled circuits and/or additive homomorphic encryption. Nissim and Weinreb [203] use a combination of the two for oblivious Gaussian elimination with communication complexity roughly $O(k^2)$ and round complexity $(k^{0.275})$. Kiltz et al. [152] give an improvement that lowers the round complexity to $O(\log k)$.

Mohassel and Weinreb [190] give a constant round protocol for deciding matrix singularity with communication $O(\ell^{2+1/t})$ for every constant t. Their protocol is unconditionally secure against a malicious (honest-but-curious) adversary controlling up to one third (one half) of the parties. Security is guaranteed except for with negligible probability if $\ell^2 \ll |K|$. Assuming public key additive homomorphic encryption, the protocol is also secure against a *covert adversary*, which roughly means that cheating will be detected with some "good" probability, that controls a majority of the parties. Using reductions from Kiltz et al. [152], an efficient solution for solving a linear system of equations is possible.

When A is symmetric and positive-definite, Cholesky decomposition is often used to calculate solutions of the system. However, Gascón et al. [116] point out that the cubic cost in the dimension of A can be prohibitive, and that for large matrices an iterative method is preferable. They give a modified version of the Conjugate Gradient Descent (CGD) algorithm that improves numerical stability for fixed-point arithmetic, which is easier than floating-point arithmetic to perform securely. For performing the CGD algorithm securely in the two-party semi-honest case, the authors suggest a garbled circuit approach. The authors also give a verification protocol to check whether the output is correct in presence of a malicious adversary; however, as they mention, this does not prevent malicious attacks such as selective failure [130].

Another essential functionality in linear algebra is computing the eigenvectors and eigenvalues of a matrix, which is used for example in PageRank, data compression and principal component analysis (PCA). While there are numerous proposed solutions [125, 248, 57], none of these have a formal simulation-based security proof, which makes it hard to reason about their security properties.

Sharemind implements several of these linear algebra primitives, including LU decomposition and securely computing rank [34]; and computing eigenvalues and eigenvectors for 3×3 matrices [143].

In the verifiable outsourcing model, where a client outsources an expensive computation to a server and receives the result along with the proof that the computation was executed correctly, Zhang

and Blanton [256] give a protocol for square $\ell \times \ell$ matrix multiplication which requires $O(\ell^2)$ cryptographic operations for the client and $O(\ell^3)$ work for the server.

2.3 Secure Comparison

Secure comparison was one of the first proposed applications of MPC. Namely, Yao introduced MPC with the so called *Millionaires Problem*: Two millionaires meet on the street and decide to determine who has the largest fortune. Concerned with their privacy, they want to determine this without revealing the exact amounts of their riches. In other words, the Millionaires Problem reduces to securely evaluating the predicate $A > B$, for numbers A and B denoting the amount of the millionaires wealth respectively.

The applications of secure comparison though goes far beyond the Millionaires problem. Applications range from simple computational building blocks such as search and sorting to more complex applications such as auctions, data mining and optimization problems.

In this section we will present a few of the best known approaches to secure comparison and their trade-offs. Specifically, we will present protocols for two types of integer comparison namely the *equality* ($=$) operations and the *larger than* ($>$). We note that this is with out loss of generality as given these predicates we can obtain all the usual integer comparison predicates ($\leq, \geq, <$) by either negation or by switching the arguments.

We will focus on protocols that can be applied given a general MPC scheme. I.e., protocols that can be instantiated using MPC schemes that can evaluate any circuit. This means that the security provided by the described protocols is inherited by the concrete MPC scheme used to instantiate them. We do however distinguish comparison protocols for MPC schemes working on Boolean circuits from protocols for MPC over arithmetic circuits.

We describe the complexity of the protocols in terms of the number of non-linear operations to be securely evaluated, i.e., AND and multiplications in Boolean and arithmetic circuits respectively. These operations in most known MPC protocols require communication between the parties, meaning that the communication complexity of the overall protocol is proportional to the number of these operations. On the other hand linear operations, XOR and addition, can usually be done non-interactively. Additionally, we will note the depth of the circuit needed for evaluating the comparison. For MPC protocols that are inherently constant round (such as the various garbling schemes) this metric is not very important, however, for many MPC protocols (e.g., those based on secret sharing) circuit depth translates to round complexity of the overall protocol. In the following all computations are done securely using MPC unless stated otherwise.

Boolean MPC

In the Boolean setting Kolesnikov *et al.* [159], give general Boolean circuit constructions for a number of basic integer operations. These constructions are size optimized for MPC using garbled circuits in the sense that they minimize the use of AND operations, not the depth of the circuit. Their construction for the equality predicate computes equality between two bits x and y in the straightforward manner as $x \oplus y \oplus 1$. To compute equality of ℓ bit numbers $x = x_\ell, x_{\ell-1}, \ldots, x_1$ and $y = y_\ell, y_{\ell-1}, \ldots, y_1$ they compute the AND of the bitwise equalities, i.e.,

$$\bigwedge_{i=1}^{\ell} x_i \oplus y_i \oplus 1.$$

Doing this in a tournament tree fashion this leads to ℓ secure AND operations and $\log(\ell)$ depth, for equality.

The larger than operator for single bits can be computed as $(x \wedge y) \oplus x$. For ℓ bit numbers $x = x_\ell, x_{\ell-1}, \ldots, x_1$ and $y = y_\ell, y_{\ell-1}, \ldots, y_1$, we can then use that $x > y$ if either $x_\ell > y_\ell$ or $x_\ell = y_\ell$ and $x' > y'$ where x' and y' are the numbers made up of the $\ell - 1$ least significant bits of x and y respectively. Following this observation Kolesnikov *et al.* propose computing the comparison by iteratively computing

$$c_\ell = (c_{\ell-1} \oplus x_\ell) \wedge (c_{\ell-1} \oplus y_\ell) \oplus x_\ell$$

where c_i for $i = 1, \ldots \ell - 1$ is the comparison of the i least significant bits of x and y and $c_0 = 0$. Using this approach uses only ℓ secure AND operations, but the sequential computation of the c_i bits leads to depth ℓ.

Alternatively, Garay *et al.* [115] describe an approach to computing $x > y$ using a circuit of depth $\log(\ell)$ and approximately $3\ell - \log(\ell) - 2$ AND operations. This approach is based on the observation that $x > y$ if either $x_\top > y_\top$ or $x_\top = y_\top$ and $x_\perp > y_\perp$, where $x_\top, y_\top$ and $x_\perp, y_\perp$ are the $\ell/2$ most and least significant bits of x and y respectively. Thus the comparison can be computed by recursively computing $c_\top \oplus (e_\top \wedge c_\perp)$, where $c_\top \leftarrow x_\top > y_\top$ and $c_\perp \leftarrow x_\perp > y_\perp$ and $e_\top \leftarrow x_\top = y_\top$. Computing $c_\top$ and $c_\perp$ recursively results in a recursion tree where at the ℓ leaves single bit comparison can be done using ℓ ANDs with the method described above. In each of ℓ internal nodes one equality and AND must be computed. Using the observation that some of the equalities can be reused we get to the complexity of $3\ell - \log(\ell) - 2$.

Arithmetic MPC

Secure comparison in the arithmetic setting is more difficult than the Boolean setting as there is no clear arithmetic way to express the comparison operation. In this section we present two sets of protocols for secure comparison in the arithmetic setting that both work over a field $\mathbb{Z}_q$ but emulate Boolean protocols using the elements $0, 1 \in \mathbb{Z}_q$ to represent bits. We note that using this approach we can given two *bits* a and b compute AND and XOR arithmetically as ab and $a + b - 2ab$. All protocols in this setting require the underlying MPC scheme to be *reactive*, meaning that intermediate values in the secure computation can be revealed, computed on in the clear and then put back into the secure computation. The protocols in this section use this technique to limit the complexity by doing much of the computation in the clear. This *reactive* property is one shared by most secret sharing based MPC schemes in the arithmetic setting.

Lipmaa and Toft [173] rely on a technique of essentially generating a random *mask* $r \in \mathbb{Z}_q$ along with its bit representation $r_1, \ldots, r_\ell \in \{0, 1\} \subset \mathbb{Z}_q$, where ℓ is the bit length of elements in $\mathbb{Z}_q$ and $r = \sum_{i=1}^{\ell} 2^i r_i$.

To compute the equality $x = y$ for they first compute $m = (x - y) + r$ for an r computed as above. As m masks $a = (x - y)$ using the random r it is safe to reveal m and extract its bit representation $m_1, \ldots, m_\ell$ in the clear. The equality then reduces to computing whether $s = \sum_i^\ell m_i \oplus r_i = 0$, as this implies $m = r$ and in turn $a = 0$ and $x = y$. This reduces the problem to the equality check $s = 0$, for an $s \leq \ell$. This can be done by securely by evaluating $P_\ell(s)$ where P_ℓ is the degree ℓ polynomial so that $P_\ell(0) = 1$ and $P_\ell(x) = 0$ for all $x \in \{1, \ldots, \ell - 1\}$. Since the bits m_i and the coefficients of P_ℓ are public most of the protocol can be evaluated locally using linear operations. Only $O(\ell)$ secure multiplications in $O(1)$ rounds are needed to compute $P_\ell(s)$ and to generate r along with its bit representation. Lipmaa and Toft [173], use a clever trick to preprocess the evaluation of P_ℓ meaning the majority of the work can be done in preprocessing. The online phase of the protocol only uses a single secure multiplication

and three rounds in total. Additionally, if the bit length of a is upper bounded by some $k << \ell$ the complexity can be improved to $O(k)$ multiplications.

Lipmaa and Toft further use their equality check protocol in a protocol to compute the comparison $x > y$, using $O(\ell)$ multiplications $O(1)$ rounds in preprocessing and $O(\log(\ell))$ multiplications and rounds online. We here give a rough intuition of this protocol. The protocol uses a recursive method very similar to the method of Garay *et al.* above. However, instead of computing the expression $c_\top \oplus (e_\top \wedge c_\bot)$, by computing the entire recursion tree, they compute only along a single path of length $O(\log(\ell))$. The intuition being to first compute $e_\top$ and then only compute the comparison $x' > y'$ where $x' = e_\top(x_\bot - x_\top) + x_\top$ and $y' = e_\top(y_\bot - y_\top) + y_\top$. This gives the above complexity as the preprocessing for each of the $\log(\ell)$ comparison can be done in parallel in constant rounds while online each recursive step only uses a constant amount of rounds and multiplications but must be done sequentially.

Catrina and de Hoogh [48], give an alternative set of protocols for the greater than operation using a rather different approach. In contrast to Lipmaa and Toft, these protocol come in variants with both constant and logarithmic rounds. Namely, they start from the fact that $a = y - x < 0$ implies that $x > y$. They then use the observation that if x and y are bounded so that $a = y - x \in \{-2^{k-1}, \ldots, 2^{k-1} - 1\}$, where $k < \ell$ then we have for $d = \lfloor a/2^{k-1} \rfloor$, that either $d = 0 \mod 2^k$ implying $a \geq 0$ or $d = -1 \mod 2^k$ implying $a < 0$. To help the intuition, this essentially means that if we think of $a \mod 2^k$ as a k bit number then we have that $a < 0$ if and only if the most significant bit of a is set to 1. Catrina and de Hoogh use the same masking technique as described for Lipmaa and Toft, to compute and reveal $m = a + r$ and then show that extracting the k'th bit of a essentially reduces to a bit wise comparison between $r_1, \ldots, r_k$ and the $m \mod 2^k$. To do this they present three different techniques using $4, 6$ and $O(\log(k))$ rounds respectively and all using $O(k)$ multiplications (for small constants). Keller *et al.* [147] report on an experimental implementation of these protocols based on the SPDZ protocol. Surprisingly, they note that on 32 and 64 bit integers the logarithmic round protocol (using 5 and 6 rounds respectively) performs about 20% better than the version using just 4.

2.4 Secure Sorting

Secure sorting allows a set of parties $P = \{p_1, ..., p_m\}$ to sort a list L obliviously, i.e., without revealing its elements. L may be the result of preceeding secure computation or the concatenation of the parties' inputs, i.e., $L = l_1 \mid ... \mid l_m$ where party $p_i \in P$ holds list l_i. Secure sorting is most commonly used as a building block in a larger secure computation (after all, revealing the result of a secure sort directly leaks almost all information about the input); existing work [140, 124, 123, 120, 254] therefore focuses on instantiating secure sorting protocols via general MPC such that further secure computation can be performed on the resulting sorted list. Applications of secure sorting range from common data analytics tasks such as aggregation [140], top-k queries [140], and private set-intersection [128], to the instantiation of other cryptographic primitives such as oblivious RAM [70]. Most work on secure sorting, with the exception of [140], assumes that the lengths of the input lists are public and may be leaked.

An immediate obstacle to secure sorting is that many standard, efficient sorting algorithms are not data-oblivious, i.e., have input-dependent control-flow which may leak the participating parties' private inputs. Prior to elaborating on secure sorting, we briefly review existing constructions for sorting in-the-clear, that is without privacy constraints.

Sorting In-The-Clear

Most sorting algorithms fall within the family of *comparison sorts*. Comparison sorts order elements using only pair-wise comparison between elements. For practically efficient comparison sorts such as quicksort and merge sort, the input determines the order in which the comparisons are performed. *Sorting networks* [26, 205], on the other hand, are comparison sorts with input-independent control flow. Sorting networks were first developed in the sixties as a means to parallelize sorting and implement sorting in-hardware [26, 205]. A sorting network consists of a circuit of *compare-exchange gates*. A compare-exchange gate takes as input two elements and swaps these based on the result of their comparison. The output is routed on to further compare-exchange gates. The topology of a sorting network and therefore the order in which comparisons are performed is determined solely by the size of the input (and the particular type of sorting network used). The AKS network [10] requires $\mathcal{O}(n \log n)$ comparisons where n is the number of elements to be sorted; this is asymptotically optimal for comparison-based sorting. For practical input sizes, however, other sorting sorting networks with worse asymptotic performance of $\mathcal{O}(n^2 \log n)$ such as the bitonic sort or odd-even merge sort [26] achieve significantly better performance due to the large constant overhead of the AKS sorting network [155].

Comparison-based sorts are not the only class of sorting algorithms. *Distribution sorts* such as counting sort or radix sort do not use the comparison operator to order elements. Consequently, the lower bound of $\mathcal{O}(n \log n)$ does not apply to such algorithms. Counting sort, for instance, runs in time $\mathcal{O}(n+k)$ where k is the value of the largest input element. This potential gain in performance comes at the cost of loss of generality: distribution sorts place restrictions on the domain of the input, e.g., counting sort only works over the integers. Furthermore, the complexity of distribution sorts depends on the range of the input which is not the case for comparison-based sorts.

Secure Sorting

The above constructions have since been adopted to the domain of secure sorting. Sorting networks offer a natural basis for secure sorting algorithms as they are inherently data-oblivious; the use of sorting networks to implement secure sorting is explored in [120, 243, 140, 128]. Perhaps surprisingly, it is also possible to convert data-dependent sorting algorithms such as quicksort into efficient secure sorting protocols; Hamada et al develop this approach in [124]. Non-comparative sorting algorithms have also been studied in the context of secure sorting; Hamada et al implement a secure version of radix sort in [123]. Zhang does so for counting sort in [254].

As previously noted, secure sorting is treated as a building block for secure computation; all work we discuss here develops secure sorting protocols that can be instantiated via a general MPC scheme and as such inherits the specific schemes' security guarantees. We note that the work in [140, 124, 254, 123] develops protocols tailored to arithmetic secret-sharing based schemes, whereas [243, 128] targets garbled circuits [251]. Next we discuss existing work in more detail.

In [120] Goodrich develops the *randomized Shellsort* algorithm, a randomized version of Shellsort [232] (just as the name suggests). *Randomized Shellsort* is data-oblivious and sorts n-length inputs using $\mathcal{O}(n \log n)$ comparisons, with very high probability (the probability of failure is $1/n^b$ where b is a constant and $b \geq 1$). Goodrich proposes that randomized Shellsort can be used to efficiently implement secure sorting; the algorithm can be expressed as a circuit of compare-exchange gates, which in turn can be implemented using only secure arithmetic and secure comparison, for which a number of efficient protocols exist [115, 48, 173]. Furthermore, Goodrich provides a conversion of the original algorithm to a Las Vegas version which always succeeds, and uses $\mathcal{O}(n \log n)$ comparisons with very

high probability. Goodrich is the first to consider the use of data-oblivious sorting algorithms in the context of secure computation. In subsequent work [243], the algorithm is instantiated via garbled circuits and evaluated empirically.

Wang et al [243] develop protocols for secure sorting in the two-party setting. They consider the use of sorting networks and construct secure sorting protocols from bitonic sort and odd-even merge sort [26]. The authors also construct a protocol based on randomized Shellsort [120] discussed previously. Since sorting networks as well as randomized Shellsort are data-oblivious algorithms, the authors are able to restrict the use of secure computation to evaluating a sequence of compare-exchange gates. To evaluate the compare-exchange gates, the authors use an implementation of garbled circuits; the input and output of each gate is in XOR secret-shared form such that neither of the parties learn the intermediate results. Both the bitonic sort and odd-even merge sort require $\mathcal{O}(n\log^2 n)$ such evaluations while randomized Shellsort runs in $\mathcal{O}(n\log n)$. The authors further implement the sorting protocols in the Fairplay framework [176] and evaluate their performance. They observe that odd-even merge sort always outperforms bitonic sort while randomized Shellsort outperforms both sorting network based approaches on larger inputs (for $n = 10^6$ odd-even merge sort requires $\sim 9 * 10^7$ compare-exchange gates whereas randomized Shellsort requires only $\sim 4 * 10^7$).

Jónsson et al implement secure sorting in [140] using odd-even mergesort [26]. Just as [243], this work restricts the use of secure computation to evaluating compare-exchange gates in their sorting protocol; in contrast to [243], the authors work in the multi-party setting and use an arithmetic secret-sharing based MPC scheme for instantiating their protocols. The sorting algorithm uses $\mathcal{O}(n\log^2 n)$ secure comparisons and requires $\mathcal{O}(\log^2 n)$ rounds of communication. The authors further propose several use-cases for secure sorting as a building block for more complex secure algorithms, e.g., secure aggregation and weighted set-intersection. They combine the resulting protocols to realize a distributed, data-oblivious intrusion-detection system (IDS). Some of the higher level protocols, for instance secure aggregation, require *keyed sorting*. Instead of sorting singleton elements, a *keyed sort* sorts a list of *key-value pairs* of the form (k_0, v_0), ..., (k_n, v_n) by the key entries k_i.

In contrast to other work, the authors also consider secure sorting with the additional requirement that input lengths must remain private. To this end, the parties first securely compute the total length l of the result of the sort, i.e., the sum of the lengths of all inputs, and pad their inputs with dummy elements up to length l before running the original secure sorting protocol[2]. This hides the lengths of the parties' individual inputs but incurs a running time overhead linear in the number of contributing parties. The authors implement their protocols in Sharemind [37] and demonstrate a throughput of 2^{14} elements in three and a half minutes.

Hamada et al. propose a general technique to converting data-dependent comparison sorts into secure sorting algorithms in [124]. Common comparison sorts operate by comparing elements and branching based on the result; this clearly incurs input leakage. The authors observe that this leakage can be avoided by *obliviously shuffling* (see Section 2.5) the input before sorting; since the randomly permuted elements cannot be mapped back to the original list, securely comparing two elements and revealing the result leaks no information about the original input elements (assuming that all elements are distinct). Subsequently, one can branch on the results of the comparisons and implement efficient data-dependent comparison sorts. The authors extend the approach to handle duplicate elements by modifying the input with tie-breakers to force uniqueness. The resulting protocol works in the multi-party setting and can be instantiated via any general MPC scheme that supports *reactive* computation,

[2]Setting dummy elements to be the minimum (or maximum) of the values to be sorted allows for them to be removed from the resulting list L' by simply discarding the first (or last) $|L'| - l$ elements of L'. Should the extremes of the input values not be known, each input element can be annotated with a flag indicating if it is a dummy element or not and filtered out at the end by performing a keyed-sort on the flags.

i.e., a scheme that allows for intermediate values to be revealed and computed on in the clear. The authors provide a C++ implementation of the protocol based on a passively-secure three-party scheme based on Shamir secret-sharing and evaluate the efficiency of their approach on quicksort, showing that their approach outperforms sorting network based alternatives. They demonstrate a throughput of 10^6 elements in $\sim$1226 seconds (whereas other work such as the previously discussed approach by Jónsson et al takes $>$3600 seconds) The authors use the secure shuffling algorithm from [166] which requires only $\mathcal{O}(n \log n)$ *communications* (as opposed to secure comparisons) over a constant number of rounds; the sorting step therefore presents the main bottle-neck of the overall algorithm. Quicksort (on average) requires $\mathcal{O}(n \log n)$ comparisons and $\mathcal{O}(\log n)$ communication rounds.

We note that while the above approach by Hamada et al. yields the best performance for the case when the list to be sorted is already secret-shared among the parties, an even faster approach exists when the list is a direct concatenation of the parties' inputs. In this case, the parties may pre-sort their input lists locally and use the secure merging approach described by Huang et al. in [128] which has $\mathcal{O}(n \log n)$ worst case performance with low constants and as such may outperform quicksort.

2.5 Secure Shuffling

Secure Shuffling, also known as *Oblivious Shuffling*, refers to the process of randomly shuffling a list of elements using secure computation. I.e., given a vector of elements $\mathbf{x} = (x_1, \ldots, x_n)$ securely computing the vector $\mathbf{x}_\pi = (x_{\pi(1)}, \ldots, x_{\pi(n)})$ for a uniformly random permutation π unknown to the adversary. This functionality is a useful subroutine in a number of situations where the order of elements may otherwise leak information in a larger protocol. For example, secure shuffling has been used in protocols for private set intersection [128], secure database operations [165], oblivious RAM [147] and even secure sorting as described in Section 2.4.

In this section we will describe three state of the art techniques for secure shuffling offering a trade off between depth (round complexity) and the amount of non-linear gates (communication complexity) in the circuit to be evaluated. Both techniques (as well as most known techniques for secure shuffling) are based on the observation that composing several permutations results in a new permutation. Further, to apply a random permutation unknown to the adversary, we can let each party in the protocol supply a random permutation and securely evaluate each permutation consecutively. As long as a single honest party exists the resulting permutation will be uniformly random in the view of the adversary. Thus, secure shuffling reduces to securely computing a permutation on $\mathbf{x}$, and the techniques described below mainly differ in how they represent and evaluate permutations.

The first technique we will describe is due to Laur et al. [166] and is based on *permutation matrices*. It achieves low circuit depth at the cost of high number of non-linear gates. Specifically, the circuit requires $O(m)$ depth and $O(mn^2)$ non-linear gates, where m is the number of parties participating in the protocol. A variation of the technique reduces the depth of the circuit to $O(\log(m))$ at the cost of increasing the number of non-linear gates to $O(mn^3)$.

The second technique, described by Huang et al. [128], is based on a so-called *permutation network*. This technique reduces the non-linear gate complexity compared to that of Laur et al. at the cost of increasing the depth by a $O(\log(n))$ factor. This results in a shuffle with $O(m \log(n))$ depth and $O(mn \log(n))$ non-linear gates. In [147], Keller and Scholl extend the passively-secure protocol above to the malicious case.

The third technique is also due to Laur et al. [166] and is based on *resharing*. This technique is particularly efficient in the honest majority setting with semi-honest corruption. As opposed to the other two techniques, it only requires secret-sharing operations (and no multiplications). It has $O(2^m)$ depth and requires $O(2^m n \log(n))$ calls to the secret-sharing functionality.

We note that, while the protocols below are described in an arithmetic setting, the computation mainly involves values in the set $\{0,1\}$, and thus they easily carry over to the Boolean setting with the same complexities. Additionally, depending on the underlying scheme used for general secure computation, both protocols are secure against either semi-honest and malicious adversaries. Only, in the malicious case the input of each party must be proven to be consistent with the protocol. E.g., in the arithmetic setting the inputted values must be proven to be in the $\{0,1\}$ set. These proofs are, however, easily done using general secure computation and does not change the overall asymptotic complexities of the protocols. We refer to [166] and [147] for details.

2.5.1 Permutation Matrix based Shuffling

Any permutation, π, on a vector $\mathbf{x}$ of dimension n can be represented by a permutation matrix $M_\pi \in \{0,1\}^{n\times n}$ with exactly one 1 entry in each row and each column. I.e., for each permutation π there is a permutation matrix M_π so that $M_\pi \mathbf{x} = \pi(\mathbf{x})$.

The shuffling techniques of Laur et al., proceeds by letting each party, P_i, input a random permutation matrix M_{π_i} and then computing $\Pi_{i=1}^{m} M_{\pi_i}\mathbf{x} = M_\pi \mathbf{x} = \pi(\mathbf{x})$, where π is the permutation resulting from composing the permutations $\pi_1,\ldots,\pi_m$. If we let $\mathbf{x}_0 = x$ and $\mathbf{x}_i = M_{\pi_i}\mathbf{x}_{i-1}$ we can compute the permuted vector $\pi(\mathbf{x}) = \mathbf{x}_m$ by computing each of the $\mathbf{x}_i$ iteratively. This requires m times multiplying a $n\times n$ matrix with a n dimensional vector which can be done in $O(1)$ depth using n^2 multiplications, leading to a circuit of depth $O(m)$ with $O(mn^2)$ multiplications. Alternatively, we could first compute the matrix $M_\pi = \Pi_{i=1}^{m} M_{\pi_i}$ and then compute $M_\pi \mathbf{x} = \pi(\mathbf{x})$. If we compute M_π in a tournament tree fashion we can thus reduce circuit depth to $log(m)$. This, however, requires m matrix multiplications which uses $O(n^3)$ multiplications. Thus the overall protocol requires $\log(m)$ depth and $O(mn^3)$ multiplications.

2.5.2 Permutation Network based Shuffling

Permutation networks are circuits similar to sorting networks as described in Section 2.4, only permutation networks replace the compare-exchange gate with the more general *conditional-exchange* gate. A conditional-exchange gate takes two elements x and y swaps them based the value of a *control bit* b. Note, that such a gate is easily implemented using general secure computation by computing $x' = b(y-x)+x$ and $y' = b(x-y)+y$. For each permutation π a set of control bits can be computed so that the permutation network computes π.

The secure shuffling protocol of Huang et al. proceeds by having each party P_i pick a random permutation π_i, compute the corresponding control bits for a permutation network and input these bits to a secure computation (we stress that simply picking the control bits at random does not necessarily give a uniformly random permutation). The permutation is then computed by evaluating the resulting m permutation networks consecutively. Concretely, Huang *et al.* uses the permutation network of Waksman [242] which has depth $O(\log(n))$ and uses $O(n\log(n))$ conditional-exchange gates. Thus, the overall shuffling circuit to be securely evaluated has $O(mlog(n))$ depth and $O(mn\log(n))$ conditional-exchange gates.

2.5.3 Resharing based Shuffling

The authors aptly describe this approach as a hide and seek game among the parties in P: for each possible adversarial coalition A and its complement $C = P \setminus A$, the parties in C jointly agree on a permutation and apply it to the input vector $\mathbf{x}$ such that all parties in A remain oblivious to the permutation. For each such group A and its complement C, the protocol proceeds as follows: the parties in

A secret-share their shares among the parties in C such that the original input vector $\mathbf{x}$ is now secret-shared only among the parties in C; the parties in C jointly choose a permutation and apply it to the resulting secret-shared vector; the parties in C then secret-share the permuted vector[3] across P. Note that the protocol requires that the parties in C cannot reconstruct the original input vector, i.e., $|C|$ is below the corruption threshold. The authors also develop a covert and actively-secure variant of the above protocol.

Unlike the other two approaches, which can be implemented using any general MPC scheme that supports multiplication and addition, this technique requires a secret-sharing based backend. Furthermore, its performance degrades exponentially in the number of participating parties. However, when the number of parties is small and especially in the passive security, honest majority setting (for which the authors provide additional optimizations) this approach significantly outperforms the other two techniques: if the number of parties is treated as constant, the protocol is asymptotically optimal, requiring constant communication rounds and only $O(n\log(n))$ calls to the secret-sharing functionality.

2.6 Private Set Intersection

The Private Set Intersection (PSI) problem, in its basic form, requires that two parties, each holding a private set of elements, learn the intersection of their sets without either party learning any additional information about the other party's input. Many other variations of the problem exist, such as multi-party PSI, or PSI-cardinality (determining the size of the intersection). PSI has a wide range of realistic applications, from collaborative botnet detection [197] to determining ad-conversion rates [134]. Due to its practical relevance, PSI is a well-studied problem–there is a large body of prior work [111, 128, 215, 86, 71, 72] that ranges back over a decade–and remains an active area of research [223, 209, 216, 158, 142]. Since its initial formalization in 2004 [111], several approaches to PSI have been developed. We first summarize the evolution of PSI protocols and further discuss the most recent advances in each category. Though we focus foremost on two-party PSI, we also address other settings in Section 2.6.3.

2.6.1 Definitions

Most work we cover here targets the two-party PSI problem. We refer to the two parties as S (alternatively the *server*) and C (*client*). The inputs of the parties are sets I_S and I_C, respectively. Unless otherwise stated we assume that only C receives the result of the intersection.

PSI protocols can be categorized as *special-purpose* and *general-purpose*. Special-purpose protocols consider PSI in isolation, i.e., as the only computation the parties wish to perform, with the result revealed in-the-clear. General-purpose PSI on the other hand embeds PSI in the broader context of secure computation and provides circuit-based solutions such that the result of the intersection may be further computed on securely, be it via garbled-circuits or another secure computation scheme. While state-of-the-art special-purpose protocols are more efficient than general-purpose protocols [216], general-purpose protocols offer the obvious advantage of generality.

Finally, PSI protocols come with different security guarantees; we discuss both *passively-secure* and *actively-secure* protocols (we will also refer to these settings as *semi-honest* and *malicious* respectively).

[3] This is a slight over-simplification: the parties in C secret-share their shares of the vector. Consequently, each party $p \in P$ receives multiple shares for each element in the permuted vector; p then recombines these shares into a single share.

2.6.2 Building Blocks

Over a decade of research has naturally produced protocols that make use of a wide array of building blocks (cryptographic and otherwise). We summarize these here.

Hashing

State-of-the-art PSI protocols rely heavily on hashing for performance improvements. Here we review at a high-level the hashing techniques used, namely *simple hashing*, *Cuckoo hashing*, and *Bloom filters*.

Simple hashing maps elements into a hash table via a random hash function. The hash table consists of bins which can hold multiple elements. More formally, given a hash function $h : D \mapsto D'$, an element $e \in D$, and a hash-table T with a *bin* for each possible element in D', we map e into T by computing $a = h(e)$ and adding the result to the bin $T[a]$. An element e can be retrieved from T by computing $a = h(e)$ and searching bin $T[a]$ for e.

Cuckoo hashing, unlike simple hashing, maps elements into a hash table T for which each bin can hold at most one element. To resolve collisions (an element mapping to an occupied bin), Cuckoo hashing uses k hash functions $H = \{h_1, ..., h_k\}$: given an element e, if $T[h_1(e)]$, ..., $T[h_k(e)]$ all contain elements already, one of the elements e' is chosen at random and replaced by e; the hash functions in H are used to remap e'. This can of course lead to further evictions; the process is only repeated a limited number of times. If after reaching the threshold an element remains unmapped, it is stored in a *stash*.

Bloom Filters are data structures which allow for space-efficient membership testing; Bloom filters are probabilistic and as such may give false positives (state that an element is in the Bloom filter when it is not); false negatives however are guaranteed never to occur. A Bloom filter consists of an m-bit array T (where m is the capacity, proportional to the number of elements it is expected to hold) and k hash functions $H = \{h_1, ..., h_k\}$ each of which maps arbitrary length inputs to range $\{1, ..., m\}$. To insert an element e, compute $I = \{h_1(e), ..., h_k(e)\}$ and set $T[i] = 1$ for each $i \in I$; likewise, to check membership of e', compute $I = \{h_1(e'), ..., h_k(e')\}$ and check that *all* corresponding bits in T are set to 1.

Oblivious Transfer

Oblivious transfer (OT) [220] allows a sender S with two messages m_0, m_1 to transfer message m_b to receiver R where R chooses bit b; S does not learn b and R does not learn m_{1-b}. We refer to this as 1-out-of-2 OT. A common, more general variant is 1-out-of-N OT [99]; here S holds N messages, instead of two. Another variation is *random oblivious transfer* (ROT) in which S does not provide inputs but rather learns two random messages r_0, r_1 as a result of the transfer (R learns r_b as before). *OT extension* [27] greatly reduces the cost of OT by bootstrapping any number of OTs via a small number of base OTs; the base OTs require costly public-key cryptography operations whereas far more efficient symmetric primitives can be used for the remaining OTs. Subsequent work has improved the performance of OT by orders of magnitude and is instrumental to state-of-the-art PSI protocols such as [216, 158, 223, 209].

Oblivious Pseudo-Random Function Evaluation

Oblivious pseudo-random function evaluation (OPRF evaluation) allows two parties, A and B with respective inputs k_A and x_B to evaluate a *pseudo-random function* $f_k(\cdot)$, keyed by k_A on input x_B, such

that A learns nothing and B learns the result of $f_{k_A}(x_B)$ and nothing else. The OPRF construction was first formalized in [110].

Current state-of-the-art approaches [216, 158] instantiate PSI via OPRF evaluation. Conceptually, this can be done as follows (in practice a number of optimizations are applied, as we will discuss later): given server S with input set $I_S = \{s_1, ..., s_{n_S}\}$ of length n_S, and client C with input set $I_C = \{c_1, ..., c_{n_C}\}$ of length n_C, S chooses key k_S; jointly, C and S evaluate OPRF $f_{k_S}(\cdot)$ on each element in I_C such that C learns $I'_C = \{f_{k_S}(c_1), ..., f_{k_S}(c_{n_C})\}$; S evaluates $f_{k_S}(\cdot)$ on its own input I_S, obtaining $I'_S = \{f_{k_S}(s_1), ..., f_{k_S}(s_{n_S})\}$, permutes the result to I''_S, and sends I''_S to C; C obtains the result by evaluating the intersection $I'_C \cap I''_S$. This approach was first suggested in [110].

2.6.3 The History of PSI

For context, we summarize the milestones in two-party PSI research. The first constructions for PSI-like protocols date back to the 80s [231, 182]; the first rigorous treatment of PSI is due to Freedman et al. in 2004 [111]. The authors develop a *special-purpose* protocol based on *oblivious polynomial evaluation* (OPE), instantiated via additively-homomorphic encryption (such as realized by the Paillier cryptosystem [212]). The authors use *hashing* to improve runtime complexity. They put forward passive- and active-security protocols. Further, Freedman et al. formalize *oblivious pseudo-random function evaluation* (OPRF evaluation) and mention its application to PSI in [110]. De Cristofaro and Tsudik develop a special-purpose, actively-secure PSI protocol based on *blind-RSA* [71]; notably, they are the first to achieve communication and computation complexity linear in the number of input elements. In [128] Huang et al. challenge the belief that *general-purpose* PSI-protocols are inherently slower than special-purpose protocols. They develop an optimized boolean circuit for PSI (that has $\mathcal{O}(n \log n)$ non-XOR gates) and demonstrate that a garbled-circuit-based evaluation outperforms special-purpose protocols such as [111, 71] when the underlying public-key cryptosystems use large security parameters. Dong et al. follow with special-purpose protocols based on *oblivious transfer* and *Bloom filters* in [86]; they provide both passively- and actively-secure protocols with linear complexity; the protocols are practically efficient as they mostly consist of symmetric-key operations.

Despite these advances, *anno* 2014 PSI protocols still do not meet the performance requirements of real world applications [179]; to this end, Kamara et al. make a departure from traditional security models for PSI and propose *server-aided* PSI in [142]. In this setting they are able to match the performance of insecure, plain-text intersection.

Pinkas et al. push the boundaries of traditional PSI in the semi-honest setting in a line of work [215, 214], culminating in special-purpose protocol [216] based on OPRF evaluation (instantiated via OT) and an array of hashing techniques; the complexity of their protocol depends on the length of the representation of the inputs; Kolesnikov et al. develop a special-purpose, OPRF-based protocol that scales independently of input representation length [158]. The performance of both protocols falls within an order of magnitude of insecure set intersection.

In the malicious setting, Rindal and Rosulek [223] fix a security bug in the aforementioned actively-secure protocol by Dong et al. [86] and improve its performance; as the original protocol, their protocol is based on OT and Bloom filters. Orru et al. extend the 1-out-of-N OT protocol by Kolesnikov and Kumaresan [157] to the malicious setting and apply the resulting protocol to PSI in [209]; the protocol has a very similar structure to the semi-honest ORPF-based PSI protocol by Kolesnikov et al. [158] mentioned above; Orru et al. achieve malicious security via a consistency check akin to the one developed for actively-secure 1-out-of-2 OT in [146].

Though state-of-the-art special-purpose protocols outperform *general-purpose* protocols, the latter approach remains competitive because of its generality. The most efficient protocol in this space

is due to Pinkas et al. [216]; it uses a pair-wise comparison circuit, optimized via hashing.

Pinkas et al. put forward a comprehensive empirical evaluation of a wide range of approaches to two-party PSI in [216]; the authors implement approaches from [128, 142, 72, 86, 182] and benchmark these within a single environment, furthermore applying state-of-the-art optimizations to the underlying cryptographic primitives. As such, this work forms an excellent overview of the landscape of PSI research today.

While most research focuses on two-party PSI, other settings also exist. Kissner and Song first tackle *multi-party PSI* and *private multi-set operations* in [153]; they build upon the OPE-based approach put forward by Freedman et al. in [111]. Kissner and Song develop special-purpose protocols for intersection, union, and element reduction; the protocols can be composed arbitrarily to express more complex operations.

In the setting of multi-party PSI, the most recent advances are due to Hazay and Venkitasubramaniam [127]. Hazay and Venkitasubramaniam revisit the work by Freedman et al. [111] in the multi-party setting; as such they develop OPE-based protocols in the semi-honest and malicious setting. The authors assume a *star topology*, in which parties are connected via a central party (as opposed to the common complete topology). At a high level, the authors use the 2-party protocol by Freedman et al. [111] between each party and the central party to incrementally compute the result. In the semi-honest setting this achieves communication complexity linear in the size of the input sets and linear computation complexity for all parties but the central party (which has quadratic computation complexity).

Blanton and Aguiar develop general-purpose protocols for a wide range of set and multi-set operations in [33]. The authors optimize their protocols for an additive-secret-sharing based MPC scheme (reducing communication rounds) and achieve communication and computation complexity of $\mathcal{O}(n \log n)$ where n is the size of the input. Since all protocols are general-purpose they may be arbitrarily composed with other functionality; this work presents the most comprehensive suite of private set operations to-date.

2.6.4 PSI Today

Further, we discuss state-of-the-art *special-purpose* protocols for two-party PSI in the semi-honest setting [216, 158] and the malicious setting [223]. We also describe work that considers an alternative threat model, namely *sever-aided* PSI [142].

Semi-Honest Setting

The most efficient special-purpose two-party PSI protocols for the semi-honest setting are due to Pinkas et al. [216] and Kolesnikov et al. [158]. We will further refer to these protocols as PSZ-PSI and KKRT-PSI respectively.

PSZ-PSI [216] by Pinkas et al. builds upon techniques the authors previously put forth in [215, 214]. The protocol is based on *OPRF evaluation*. To instantiate OPRF evaluation the authors use an improved version of the *OT-extension* protocol developed in [157] as well as the *random OT* protocols of [200, 18]. At a high-level, the protocol operates by both parties S and C first hashing the elements in their input sets to hash tables using the same pre-agreed on hash functions, and then applying an OPRF-based protocol conceptually akin to the one outlined in (2.6.2) to each bin in their hash tables to compute and output masks of their elements to C. C obtains the final result by computing the intersection over the masked sets. The complexity of the previous state-of-the-art PSI protocol [214] depends on the bit-length l of the input representation; to this end, [214] suggests the use of

permutation-based hashing [16] which reduces the length of the elements on which the OT protocol operates by $\log n_C$ bits, where n_C is linear in the number of elements in the input set of C. The protocol presented here also makes use of permutation-based hashing and achieves complexity that is *independent of the bit-length for input sets of up to a billion elements* by improving the underlying OT-extension protocol.

Further we describe the protocol in more detail: both parties S and C first hash the elements in their sets, I_S and I_C respectively; S uses *simple hashing* to map inputs I_S into two-dimensional hash-table T_S, and pads all bins up to a pre-computed maximum bin size with dummy elements; C uses Cuckoo hashing[4] to map inputs I_C into one-dimensional hash-table T_C (and stash) and fills all empty bins with dummy elements; the protocol is designed such that $|T_S| = |T_C|$ and that for any $e_C \in I_C$ and $e_S \in I_S$ if $e_C = e_S$ then the bin e_C maps to in T_C has the same index as the bin e_S maps to in T_S (for details on how this is achieved we refer the reader to the original paper); S generates key k_b for each bin $b \in T_S$; for each bin $b \in T_C$ which holds a single element e (since C used Cuckoo hashing), S and C jointly evaluate OPRF $f_{k_b}(e)$ (C receives the results); C thus obtains T'_C which corresponds to the original hash table T_C but with all entries masked; for each bin $b \in T_S$, S evaluates $f_{k_b}(e)$ for each $e \in b$, obtaining T'_S with masked entries, collects all the entries into one set V, permutes it to V' and sends the result to C; C obtains the final intersection result by checking for each masked entry $m \in T'_C$ if $m \in V'$ and outputting the corresponding unmasked entry in T_C; a similar procedure is repeated for elements in the stash of C.

The protocol has computation overhead linear in n (the size of the input sets) and requires $\mathcal{O}(n \log n)$ bits to be sent; an implementation of the protocol realizes the highest throughput demonstrated in the semi-honest setting: the intersection of two sets of one billion ($\sim 2^{30}$) elements each runs in approximately 34 hours.

Just as PSZ-PSI, **KKRT-PSI** [158] by Kolesnikov et al. builds upon the protocol in [214]. While the protocol in [214] uses OT extensions to implicitly instantiate OPRF evaluation, Kolesnikov et al. make the OPRF evaluation explicit. We have already seen the overall structure of the KKRT-PSI protocol in PSZ-PSI (in fact PSZ-PSI adopts it from KKRT-PSI with minor modification): the parties S and C hash their inputs to hash-tables, use OPRF evaluation to mask the elements in the tables, and compute the intersection over the masked results. However, unlike [214] and PSZ-PSI, Kolesnikov et al. achieve complexity that is *entirely independent of the length of the representation of the input*. They do so by using a novel protocol for the OPRF evaluation step which they construct by modifying the OT extension protocol of [157]. The key insight is that the error correcting code used to implement [157] can be replaced by a pseudo-random code which does **not** depend on length of the elements the OPRF is evaluated on.

KKRT-PSI has linear communication and computation complexity in the number of elements in the input sets and is thus asymptotically superior to PSZ-PSI. An intersection of two sets, each holding 2^{24} elements (with representation size of 128 bits), takes approximately one minute.

Malicious Setting

In the malicious setting, the current state-of-the-art protocol for 2-party PSI is due to Rindal et al. [223] which we will further refer to as RR-PSI.

RR-PSI builds upon the protocol by Dong et al. in [86]; it corrects a security flaw in the actively secure protocol and significantly improves its performance. Just as [86], RR-PSI realizes PSI via *oblivious transfer* and *Bloom filters*. Further we discuss the details of RR-PSI. To aid our narrative

[4] Using this combination of simple hashing and Cuckoo hashing is shown to be more efficient than other hashing approaches in [214].

we first outline the *passively-secure* protocol by Dong et al. [86]; we then describe the challenge of extending the protocol to active security and how RR-PSI overcomes this challenge.

The protocol by Dong et al. uses a construction the authors refer to as *garbled Bloom filter* (GBF). Like a regular Bloom filter, a GBF uses k hash functions and table T with m slots; however instead of storing single bits in T, a GBF stores m-bit strings; a GBF is initialized to uniformly random strings; to insert element e, the element is XOR-secret-shared to $S = \{s_1, ..., s_k\}$ and each share stored in T under positions $h_1(e), ..., h_k(e)$ respectively. To check membership of element e', we check if $e' = T[h_1(e')] \oplus ... \oplus T[h_k(e')]$.

The *passively-secure* version of the protocol proceeds as follows. Given our parties S and C, with respective inputs I_S, I_C, S and C agree on k hash functions to use for the underlying Bloom filters and Bloom filter capacity m; S stores its elements I_S in garbled Bloom filter F_S; C stores its elements I_C in regular Bloom filter F_C; for each position $s \in \{1, ..., m\}$, S and C perform an OT, where C inputs $F_C[s]$ as its choice bit and S supplies messages $(r, F_S(s))$ where r is a random string of length m (in the actual protocol this is realized via a single 1-out-of-m OT); C therefore learns F'_S where $F'_S[s] = F_S[s]$ if $F_C = 1$ and a random string otherwise; C can use F_C and F'_S to locally compute $I_S \cap I_C$.

This protocol is of course not secure against a malicious C: C can simply set all of its choice bits to 1 and thus recover I_S. For the malicious setting, Dong et al. develop a protocol which forces C to input a threshold of 0 choice-bits (otherwise C is unable to recover any of the true values of F_S). Unfortunately, their protocol suffers from a selective failure vulnerability (identified independently in [162, 223]) where a malicious S can recover some of I_C by abusing the very mechanism put in place to protect against a malicious C.

RR-PSI fixes this vulnerability with a *cut-and-choose* approach that protects against C setting too many choice-bits to 1. The authors also include several performance optimizations; we omit these for clarity. Unlike the original protocol, S and C will perform N 1-out-of-2 OTs where $N = m + R$, m is the capacity of the underlying Bloom filters, and R can be thought of as a security parameter for the subsequent cut-and-choose step; first, S prepares a list of N message pairs $M = \{(m_{1,0}, m_{1,1}), ..., (m_{N,0}, m_{N,1})\}$ (all messages are of length m and chosen uniformly at random); C prepares list of N bits $B = \{b_1, ..., b_N\}$ such that the proportion of 1 bits is within an accepted threshold; for each $i \in \{1, ..., N\}$ S and C perform an OT where S enters messages $(m_{i,0}, m_{i,1})$ and C enters as choice bit b_i; in the cut-and-choose step, S chooses random subset $T \subset \{1, ..., N\}$ of the OTs performed (such that $|T| = R$) and challenges C to prove that its choice bits set to 1 were in fact within the acceptable threshold for the selected OTs; C can do so by sending back m_{j,b_j} for each $j \in T$; if S discovers too many choice-bits set to 1, S aborts, otherwise the protocol continues; C finds a permutation π that maps the remaining choice bits B' for the unopened OTs to a regular Bloom filter F_C (which encodes I_C) and sends the permutation to S[5]; S obtains M' by removing from M all message pairs corresponding to the opened OTs, applies π to M', maps each pair $(m_{j,0}, m_{j,1})$ in M' to $m_{j,1}$, uses the result to construct garbled Bloom filter F_S; S sends F_S to C; C computes the resulting intersection using F_S and the messages it received in the OT step for choice bits set to 1.

RR-PSI operates in two stages: an *offline* and *online* phase. The offline phase is entirely data-independent and pre-computes material to be used in the data-dependent online phase. Communication (bits sent) for the offline phase scales linearly in the input set size n, but quadratically in the security parameter. Communication for the online phase scales with $\mathcal{O}(n \log n)$. Finally, the authors benchmark an implementation: an intersection of two sets, one million ($\sim 2^{20}$) elements each (with bit-length 128) requires approximately two minutes (of which the online phase takes only $\sim$14

[5] This requires that B' contains a sufficient number of choice bits set to 1. We refer the reader to the paper on how this is achieved.

seconds).

Server-Aided PSI

Even more efficient protocols for PSI exist if one concedes a relaxation in security guarantees. The work by Kamara et al [142]–further referred to as **KMRS-PSI**–is state-of-the-art within *sever-aided PSI* which considers a weaker security model than the protocols described above: it assumes the existence of a special party which aids the other parties in computing the intersection of their sets and is *trusted not to collude with any of them*.

More formally, we define the scenario of server-aided PSI as follows: given a set of n parties $P = \{p_1, ..., p_n\}$, each holding input sets $I_1, ..., I_n$ respectively, and a special party S (which we also refer to as the *server*), the parties in P compute $I_1 \cap ... \cap I_n$ with the aid of S; the parties in P learn the result; S learns nothing[6]. The authors provide protocols secure against a semi-honest, covert, and malicious server; the parties in P are assumed to be malicious in all protocols. The assumption upon which the protocols rest is that S does not collude with any parties in P, i.e., that an adversary can either corrupt several parties in P or the server, but not both. Apart from the protocols targeting different adversarial models, the authors also develop a protocol which hides the size of the result from the server, and a protocol which guarantees *fairness*, i.e., that either all parties in P learn the result, or none do.

Further we outline the base protocol secure against a semi-honest server upon which the other protocols are built. The parties in P agree on a key k to a pseudo-random permutation F; each party $p_i \in P$ with input set $I_i = \{e_1, ..., e_m\}$ computes $I_i' = \{F_k(e_1), ..., F_k(e_m)\}$, permutes the result to I_i'' and sends I_i'' to S; upon receiving all permuted, masked input sets, the server computes the intersection over these sets, permutes the result, and publishes it to all parties; the parties obtain $I_1 \cap ... \cap I_n$ by applying F_k^{-1} to each element in the masked result they received from S.

This approach yields an immense improvement in performance over state-of-the-art protocols in the traditional setting: in the semi-honest case, the intersection of two sets holding one billion elements each completes in approximately 10 minutes (only 10% slower than the insecure plaintext version).

[6]Depending on the protocol, the input sizes and result size are leaked to the server. Furthermore, should the server possess prior knowledge about one of the input sets, it may learn the existence of specific elements in the other input sets.

3 Data Mining and Machine Learning

Data mining is an active research field studying how to manage and analyze (large) datasets to discover and make use of new knowledge in a form of descriptive (explorative), predictive and prescriptive patterns and models of varying complexity. Data mining is often used as an umbrella term covering different aspects of data analytics including different modeling and processing steps facilitating knowledge discovery.

There are different methodological aspects of using data mining as a scientific toolbox, including aspects of data engineering, model selection and evaluation, among others. The discovered knowledge should be valid, (i.e. discovered pattern should be statistically significant, predictive models should generate accurate predictions on new data with some certainty), and useful (i.e. providing new insights or allowing for better decision making). It is also common to consider other requirements on the data mining process, e.g. providing transparency, and certain level of guarantees with respect to model accuracy and fairness, i.e. treating humans without discrimination. In our overview we omit these methodological perspectives and focus instead on providing a categorization of the common data mining tasks (Section 3.1) and techniques that are used to address them (Section 3.2). We consider data mining as a process in Section 3.3. Then we overview common application areas (Section 3.4) and operational settings (Section 3.5) that impose different requirements on data mining techniques.

3.1 Data Mining Tasks

Data mining tasks are typically divided into two main categories: *predictive modeling* including classification, forecasting, regression, prediction, ranking, and *descriptive modeling* including clustering, explorative data analysis, pattern mining and outlier analysis. Many description modeling tasks are formulated as unsupervised learning, i.e. we try to discover some interesting global or local structures from the unlabeled data.

Predictive modeling is often formulated as a *supervised learning* problem. It is worth mentioning that predictive modeling is not necessarily about predicting future events like e.g. in future demand forecasting. We can formulate predictive modeling tasks for nowcasting, e.g. identifying user intents or making a diagnosis and even making predictions for events occurred in the past (e.g. classifying some old texts by topic) or having no connection to temporal dimension. Another special case of predictive analytics is *prescriptive modeling* that usually refers to what-if analysis and may involve elements of predictive, descriptive and explanatory modeling. Uplift modeling, persuasion profiling, and recommendations task would be typical examples. E.g. uplift modeling can help us to study what would be a marginal effect of a movie ticket coupon or a 15% off coupon in getting more return customers. From the persuasion profiling perspective we can set the goal to learn how to choose an action/persuasion (type of coupon in this case) given an individual? Assuming that there is no globally better action, i.e. some people may prefer movie tickets and others a discount coupon.

In the following we discuss classification, clustering, pattern mining and outlier analysis as the most popular data mining tasks.

Classification. Consider the task of classifying e-mails to spam vs. relevant ones. Given examples of both spam and relevant e-mails and knowing which are which, we want to be able to tell correctly for each new email whether it is spam or not. More generally, classification is the process of learning a model (from a given set of labeled examples) that would allow to classify *new* objects to one or more *predefined* categories (as accurately as possible). More formally, it can be defined as follows. We aim to predict a target variable y (binary or categorical) given a set of input features $X \in \Re^p$. An *example*

is one pair of (X, y). For instance, X is a set of terms contained in an e-mail and $y =$ "*spam*" is the true label indicating (ir)relevance of the example. In the *training examples*, that are used for model building, both X and y are known. In the new examples, X is known, and y should be predicted.

Classification can be thought through the geometric interpretation (finding a decision boundary separating examples of one class from the other) or the probabilistic interpretation. According to the Bayesian Decision Theory [91], a classification can be described by the prior probabilities of the classes $p(y)$ and the class conditional probability density functions $p(X|y)$ for all classes $y = 1, \ldots, c$, where c is the number of classes. The classification decision is made according to the posterior probabilities of the classes, which for class y can be represented as

$$p(y|X) = \frac{p(y)p(X|y)}{p(X)}.$$

The type of the target variable space depends on the task. In classification the target variable takes categorical values (class labels), while in regression the target variable takes continuous values, i.e. $y \in \mathbb{R}$. In case of learning to rank, we can also think of classification like settings, but instead of considering crisp category labels we can use probabilistic output of classifiers to rank instances by how likely there are to be in the class of our interest.

Clustering. A clustering task is typically defined as follows: given a set of examples (data points), partition them into groups containing subsets of examples that are similar to each other. The goal of clustering is to get groups that are meaningful and/or useful. But clustering, being an explorative data analysis in nature, is hard to formulate in such a way that we can expect meaningful and useful grouping by design. Even the notion of the cluster is not well-defined in many applications.

Consider k-Means, that is perhaps the most well-known and the most popular approach that belongs to the group of partitional clustering techniques, aiming to divide data into subsets of k non-overlapping clusters represented by their centroids or prototypes such that each example belongs to one of the clusters, namely the one centroid of which is the closest to the corresponding example. k-Means is defined as an optimization task in which, given examples X_1,...,X_n and a k set by a user, we search for best placement of centroids $C_1, \ldots, C_k$ such that the sum of squared (shortest) distances of examples to centroids is minimized, i.e. $\sum_{i=1}^{n}[min_j Dist(X_i, C_j)]$. The basic k-Means algorithm simply iterates two steps till convergence: 1) form k clusters by assigning each point to the closest of the current centroids, and 2) recompute centroids for each cluster based on current assignments. Even if we were to find an optimal solution, we have no guarantee that we find useful or understandable clusters. Evaluation of clustering as other unsupervised learning techniques and their results is rather difficult unless we have a way to measure such expected utility.

Besides k-Means there are lots of other clustering formulations and corresponding approaches, including hierarchical clustering, density-based clustering (e.g. DBSCAN [97]; cf. its variant for evolving data [113]), spectral clustering [30], subspace clustering (e.g. CLIQUE [6], projected clustering [3], probabilistic clustering (e.g. expectation-maximization [139]) among several others.

Pattern mining. Pattern mining is the process of finding interesting co-occurrences in the data. The most common problem formulation is association pattern mining in transactional data. E.g. in the context of market basket analysis we have sets of items bought by customers (transactions), and the goal is to determine associations between groups of items bought by customers. Frequent pattern mining and association rule mining have been the most popular tasks. However, mining rare association rules [156], class association rules [258], emerging patterns, subgroup discovery [204], exceptional

model mining [92], data summarization [20], motif discovery [106] and many other problem formulations have been considered and studied extensively over the past years.

Outlier analysis. Outlier analysis and anomaly detection deal with identifying examples (as individual as groups of related ones) that are "not normal" or anomalous, i.e. distant from other examples in some way. Typically, this is done in the context of some model of what normal behavior is.

Anomaly detection is often defined as a monitoring/detection task, e.g. monitoring for possible equipment failures, credit card fraud, intrusion detection, money laundering, data leakage etc. Unlike binary classification tasks, anomaly detection typically does not have enough anomalous examples for traditional supervised learning and therefore specialized approaches have been developed. A comprehensive overview of anomaly detection problem formulations and techniques to address them can be found in [53].

3.2 Data Mining Foundations and Techniques

There are many dimensions that can be considered for categorizing hundreds of popular data mining techniques. The theory-oriented data mining frameworks are based mainly on one of the four following paradigms:

- one of the three statistical paradigms: statistical experiment paradigm, statistical learning from empirical process paradigm, and structural data analysis paradigm,
- the data compression paradigm, i.e. considering data mining as data compression by means of finding some structure or knowledge within it,
- the database paradigm based on the idea that all the power of discovery is in the query language, and
- the machine learning paradigm where the idea is to let the data suggest a model.

Here we adopt the categorization of Domingos [83] that helps to link machine learning foundations to different groups of the data mining techniques. Domingos call them the five tribes of learning systems. They include symbolic learning, connectionism, evolution, probabilistic (Bayesian) inference, and learning by analogy. We briefly review each of these tribes one by one giving examples of classification techniques.

Symbolic Learning Symbolic learning is used an umbrella term for machine learning and AI approaches that focus on high-level symbolic (human-readable) representations of problems, logic and search.

The most common approaches to symbolic machine learning include induction (typically top-down) of decision trees and induction of rules (including inductive logic programming). Decision trees are classification functions represented as trees. Nodes of a tree are attribute tests. The branches of the tree are attribute values. The leaves correspond to class labels. Rules are implications in either propositional or predicate logic. Lost of data mining techniques have been developed to induce decision trees and rules from the training data. Popular approaches include e.g. C4.5 [219] and RIPPER [61]).

Inductive Logic Programming (ILP) investigates the construction of logic programs from training examples and background knowledge. ILP can be seen as a combination of logic, statistics,

and computational control; the logical part is related to the formation of hypotheses, the statistical part – to evaluating their degree of belief, and the computational control part – to guiding the search over the space of hypotheses. While decision trees typically have rather limited expressive power, ILP allows for much more flexible representation, because it uses an expressive first-order logic framework. The framework also allows the user to incorporate background knowledge. Recent advances in ILP can be found in [194].

Connectionism Artificial neural networks (ANNs) are the main kind of connectionist systems inspired by the biological neural networks of animal brains. Such systems can learn from examples without task-specific programming and explicit feature engineering that makes them particularly suitable for applications in which rule-based programming is not feasible, e.g. imaging, speech recognition and text translation.

An ANN consists of a collection of connected artificial neurons (cf. axons in a biological brain) organized in layers. Each connection (cf. synapse) between a pair of neurons can transmit a signal. Neurons and connections have weights that increase or decrease the strength of the signal. Signals travel from the first input layer to the last output layer. Different layers may perform different kinds of transformations on their inputs. Neurons may have different thresholds such that only if the aggregate signal is above that level they propagate signal further. ANNs have high expressive power as they can approximate arbitrary functions to any degree of accuracy by learning these weights. However, training an effective ANN is difficult – two major issues are overfitting and computation time. Neural networks have had ups and downs over the past decades. The original idea of ANNs was to mimic real brains in the way they learn. However, some ideas like backpropagation – passing information in the reverse direction – were found to be effective and adopted in learning schemes. A recent success of ANNs in several applications, notably in computer vision and speech recognition, is associated with Deep Neural Network (DNN) that are ANNs with multiple hidden layers between the input and output layers. The extra layers enable composition of features from lower layers,

Deep architectures include many variants of a few basic approaches, each finding success in particular application domains. DNNs are typically feedforward ANNs in which signal propagates from the input layer to the output layer without looping. Recurrent neural networks (RNNs), in which data can flow in any direction, and Long short-term memory (LSTM) are found effective for language modeling. Convolutional deep neural networks (CNNs) are used successfully in computer vision and speech recognition.

All DNNs must consider many training parameters, including the number of layers and number of units per layer, the learning rate and initialization of weights. This creates challenges not only from the computational perspective (i.e. cost in time and computational resources), but also from the perspective of overfitting. Furthermore, DNNs are also prone to overfitting because of the added layers of abstraction inviting to model various rare dependencies in the training data. Therefore, it is common to use regularization methods such as unit pruning, weight decay and sparsityand dropout that helps to exclude rare dependencies. Other different tricks like batching are used to speed up computation. The foundations of modern DNNs are covered in the recent book [119].

Evolutionism This tribe developes machine learning approaches that simulate the evolutionary process. Genetic algorithms would be a popular category of such approaches. Genetic algorithms are stochastic search algorithms which act on a population of possible solutions to find which

one solves the problem best. Genetic algorithm are loosely based on the mechanics of population genetics and selection. The candidate solutions are encoded as *genes*, i.e. strings of characters from some alphabet. New solutions can be produced by evolving current population of solutions by *mutating* and *mating* its members The better offspings have higher chances to breed, mutate and survive. Many problems in machine learning can be naturally formulated as genetic search, e.g. the problem of selecting best features.

Probabilistic Inference The main idea of machine leaning approaches based on probabilistic Bayesian inference is to reduce uncertainties by incorporating new evidence. The Bayes theorem serves as the basic foundation block. Popular examples of such approaches include Bayesian classification and topic modeling. One of the simplest yet effective approaches is Naive Bayes classification [85]. For developing techniques capable of inferring more complex models it is common to employ graphical models, an umbrella term for Bayesian and Markov networks of different kinds.

Learning by Analogy Learning by analogy is one of the most intuitive methods of inference in human cognition. While encountering a new problem a person is reminded of past similar situations. That past relevant experience helps to choose behaviors that were appropriate in the past adapting when needed to meet the peculiarities of the currently encountered problem. Being able to identify relevant similarities between past and current problems serves as the basis for generalization. Nearest neighbor approaches and support vector machines (kernel machines) [67] are two representative categories of analogizers in machine learning.

The k-nearest neighbors algorithm (k-NN) is a non-parametric method widely used for classification. The input consists of the k nearest training examples in the formed feature space. The output is a class membership defined by a majority vote of identified neighbors, i.e. a new example is assigned to the class most common among its k nearest neighbors. If $k = 1$, the class label of that single nearest neighbor is assigned. k-NN is a type of instance-based learning, or so-called lazy learning. The classification function is only approximated locally. The computations (querying for k nearest neighbors and then deciding on the label) are perform at the classification, i.e. there is no training phase as such. The k-NN algorithm is among the simplest and most wide-spread of all machine learning algorithms. If we can define a suitable similarity computation, i.e. given two objects X_1 and X_2, determine a value of the similarity Sim(X_1,X_2) between the two objects, we can use k-NN for classification, clustering, ranking, outlier analysis, searching for top n closest matches, query by example, etc on various complex data types going beyond vector space representation and Euclidian distance. Luckily, lots of effective similarity measures have been introduced already. We can use them for measuring how alike are two nodes in a graph (SimRank [161], PathSim [234]), a pair of time-series shapes (DTW, LCSS), a pair of documents (cosine-semilarity), a pair of strings of text (Edit or Levenshtein) etc.

Support vector machines (SVM) represent training examples as points in space. During the training phase they are mapped such that the examples of one class are divided from examples of another class by as wide gap as possible. The gap is defined by corresponding support vectors from each class. New examples to be classified are mapped into the same space. A class label is assigned depending on which side of the gap a new instance fall.

Please note that each kind of such techniques can be adopted for all or most of the categories of the data mining tasks discussed in the previous section. We discussed the case of the nearest neighbour like approaches above. But many other approaches, e.g. SVMs can be used for

classification [1], clustering [31] and outlier detection [14]. Similarly, deep learning approaches exist for classification, clustering, and anomaly detection.

Besides learning an individual global model for classification, clustering or anomaly detection, it is common to consider ensemble learning – inducing a finite set of alternative models that are used together (e.g. simply vote) to produce a final decision. Many ensemble learning approaches including bagging, boosting and other ideas have proven to be effective in practice. Random forest [45] and modern boosting like XGBoost [58] techniques are among the current state of the art best performers.

Disregarding whether we consider individual learners or ensembles, all of them consist of three main components, each introducing bias to what kind of models they favor [82]:

Representation – some formal language in which model is described; its expressive power defines the hypothesis space and what concepts can be learnt (e.g. a linear classifier can model only linearly separable concepts; an ensemble of linear classifier can be already much more expressive).

Evaluation – a scoring function that says how good a model is that in turn help to choose one hypothesis over the other. In symbolic learning we use accuracy (fraction of correct predictions) or information gain. In neural networks we use a continuous error measure, e.g. squared error – the sum of the squares of the differences between the predicted and the true values. In Bayesian learning the posterior probabilities are used, in SVMs – the margin defining the width of the gap we try to maximize. The minimal description length principle can be used to control for model complexity.

Optimization – a mechanism defining how to search the hypothesis space such that we have a good chance to find the highest-scoring model efficiently. For many learners, it is common to start with the most simple hypothesis, relaxing it if needed to explain the data more accurately. In symbolic learning, characteristic search algorithm is inverse deduction, in neural networks – gradient descent, in evolutionary approaches – genetic search, including crossover and mutation. In SVMs – constrained optimization is used.

3.3 Data Mining Process

Recall the e-mail spam classification example introduced in Section 3.1. There are several issues to consider along the classifier training process – how to formulate the classification task, how to process e-mails to extract features and form training instances, how to eliminate noise and redundancy and keep the predictive features, how to choose a suitable classification technique, how to evaluate and optimize the classification model through (hyper-parameter) model selection. These different aspects of the problem brings us to the consideration of the data mining as a process.

The CRISP-DM model [55] describes the classical data mining process, where the life cycle of a data mining project spans over six phases: business understanding, data understanding, data preparation, modeling, evaluation and deployment. Reinartz's framework [221] follows CRISP-DM with some modifications, making modeling steps more explicit. The high-level process steps are summarized in Figure 1.

The business understanding phase aims at formulating business questions, and translating them into data mining goals. The data understanding phase aims at analyzing and documenting the available data and knowledge sources in the business according to the formulated goals, and providing initial characterization of data. The data preparation phase starts from target data selection that is

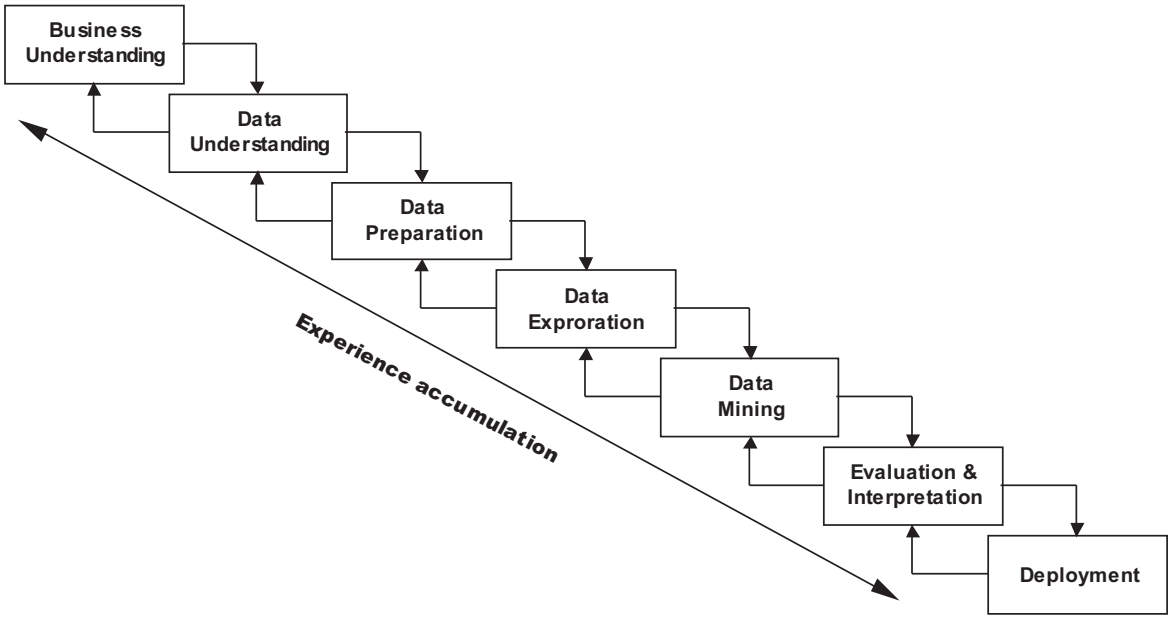


Figure 1: Knowledge discovery process: from problem understanding to deployment. Arrows indicate the most important and frequent dependencies between the phases (adapted from [221]).

often related to the problem of building and maintaining useful data warehouses. After selection, the target data is pre-processed in order to reduce the level of noise, pre-process the missing information, reduce data, and remove obviously redundant features. Next, the data exploration phase aims at providing the first insight into the data, evaluate the initial hypotheses, usually, by means of descriptive statistics and visualization techniques. The Data mining phase covers selection and application of data mining techniques, initialization and further calibration of their parameters to optimal values. The Evaluation phase typically considers offline evaluation on historical data. In predictive modeling, one would typically analyze simulated performance of the data mining system with respect to some suitable measures of accuracy (such as precision, recall, or AUC, among others), or utility (for instance, expressed as cost-sensitive classification). Finally, the most promising predictive model is deployed in operational settings, and the performance is regularly followed up.

Data mining techniques are aimed to be generic, i.e. being applicable on variety of data types and application settings. However, different application settings bring in certain constrains on the choice of possible data mining approaches. Next, we provide an overview of typical data mining tasks across different applications and then discuss the variety of application settings.

3.4 Data Mining Applications

Data mining already plays an important role, or has a high potential to be deployed in a number of applications. Several categories of such application domains have been summarized in [172]. Table 2 presents the summary of categorization of applications grouped into three application blocks: monitoring and control, information management, and analytics and diagnostics.

For a compact representation each industry (rows) is assigned a group of applications that share common supervised tasks. As it can be seen from the table, for each of the industries or groups of industries, more than one application type can be relevant. More generally, given an application use

case, several data mining tasks have to be defined and addressed to get a practical solution.

Table 2: Categorization of applications by type and industry.

Industrial Applications	**Monitoring and control**	**Information management**	**Analytics and diagnostics**
Healthcare and wellbeing	stress detection, anesthesia control	re-hospitalization prediction	medical diagnostics, antibiotic resistance prediction
Security, Police	fraud detection, insider trading detection, adversary actions detection	next crime place prediction	crime volume prediction
Finance, Banking, Telecom, Insurance, Marketing, Retail, Advertising	monitoring & management of customer segments, bankruptcy prediction	product or service recommendation, including complimentary, user intent or information need prediction	demand prediction, response rate prediction, budget planning
Production industry	controlling output quality	–	predict bottlenecks
Education (e-Learning, e-Health), Media, Entertainment	gaming the system, drop out prediction	music, VOD, movie, news, learning object personalized search & recommendations	player-centered game design, learner-centered education

3.5 Operational Settings and Learning Modes

Real application tasks can be mapped into several dimensions, including in particular; properties of the learning task, environment from which data comes, and (online) operational settings. This information is essential to determine the characteristics that the learning system needs to possess, the properties that must be prioritized when designing such a system and the evaluation criteria of the system performance. Table 3 summarizes some of the properties of the application tasks [172].

Operational settings determine availability of the ground truth in an online operation, such as, arrival of true labels in classification, or true target values in regression tasks. Labels may become known immediately in the next time step after casting the prediction (e.g. food sales prediction). Labels may arrive within a fixed or variable time lag (in credit scoring typically the horizon of bankruptcy prediction is fixed, for instance, to one year, thus true labels become known after one year has passed). Alternatively, the setting may allow to obtain labels on demand (e.g. in spam categorization we can ask the user the true status of a given message).

Requirements for the speed of decision making need to be considered when selecting, which algorithms to deploy. We shall recall in data mining we have training phase and application phase, and operational constrains may differ for each of these phases, i.e. time constrains for training a model vs. time constrains for casting predictions. In some applications prediction decisions may be required immediately (fraud detection), the sooner the better, while for other analytical decisions timing may

Table 3: Summary of properties of applications [172].

Data and task	***task***: prediction, classification, detection, clustering, itemset mining; ***input data type***: time series, relational, graph, bags or mix; ***incoming data***: stream, batches, collection iterations on demand; ***complexity***: volume; multiple scans; dimensionality; ***missing values***: unlikely, random, systematic;
Operational settings	***label availability***: real time, on demand, fixed lag, variable lag; ***decision speed***: real time, analytical; ***costs of mistakes***: balanced, unbalanced; ***true labels***: objective, subjective;

be more flexible (e.g. credit scoring decision may reasonably take one-two weeks).

The cost of errors is an aspect to consider when selecting an evaluation metric for monitoring of performance. In traditional supervised learning different types of errors (e.g. false positives, false negatives) may resolve to different losses. In some applications prediction accuracy may be the main performance metric (e.g. in online mass flow prediction), in other applications accurate and timely identification of changes as well as accurate prediction are important (e.g. in demand prediction). In the online setting, discrepancies in time may as well have associated error costs (for instance, too early prediction of a peak in food sales would still allow to sell the extra products later, but too late prediction would lead to throwing away the excess products).

Finally, the ground truth labels may be objective based on clearly defined and accepted rules (e.g. bankrupt or not bankrupt company) or subjective, based on a personal opinion (e.g. interesting or not interesting article). Alternatively, the true labels may not be available at all being impossible or too costly to measure or define in a direct way.

For the training data itseld it is possible to distinguish two high-level types of data, dependency- and nondependency-oriented data. E.g. when dealing with vector-space (feature-value) representation of examples (also called instances, tuples or data points) it is common to make i.i.d. (independent and identically distributed) assumptions about the data. However, when we consider time-series data or graph data, it is common to model corresponding dependencies.

We can distinguish two learning modes: *offline* learning and *online* learning. In offline learning the whole training data must be available at the time of model training. Only when training is completed the model can be used for predicting. In contrast, online algorithms process data sequentially. They produce a model and put it in operation without having the complete training data set available at the beginning. The model is continuously updated during operation as more training data arrives.

Less restrictive than online algorithms are *incremental* algorithms that process input examples one-by-one (or batch-by-batch) and update the decision model after receiving each example. Incremental algorithms may have random access to previous examples or representative/selected examples. In such a case these algorithms are called *incremental algorithms with partial memory* [177]. Typically, in incremental algorithms, for any new presentation of data, the update operation of the model is based on the previous one. *Streaming* algorithms are online algorithms for processing high-speed continuous flows of data. In streaming, examples are processed sequentially as well and can be examined in only a few passes (typically just one). These algorithms use limited memory and limited processing time per item. Such settings are discuss in the streaming computing model in Section 5.

For example considering our e-mail classification task, we can have only some e-mails labeled to

spam vs. relevant ones and many more unlabeled. This can encourage us to use semi-supervised learning approaches that make use of both labeled and unlabeled examples combined, e.g. using expectation-maximization approaches. Depending on our assumptions on data availability and their distribution we can use methods for incremental classification, online classification, active learning, adaptive learning, multi-task learning, one-class learning, or transfer learning.

Data mining on streaming data has two main perspectives or kinds of challenges to address:

- keeping representation and models uptodate, i.e. addressing the problem that data distribution and concepts change over time (cf. variability and volatility);
- on-the-fly (near) real-time processing settings assuming a streaming computing model (in particular, we can read data only once).

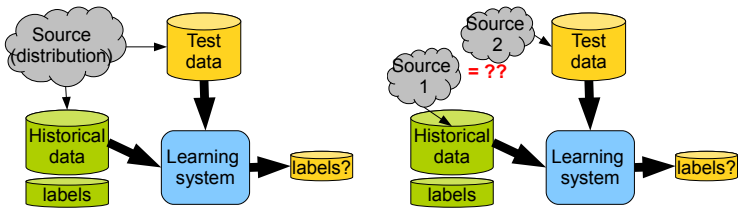
We overview there two perspectives one by one in the next sections.

4 Streaming: Learning over Evolving Data

In the era of big data, many data mining projects shift their emphasis towards the evolving nature of the data that requires us to study the automation of feedback loops more thoroughly. In the standard data mining and machine learning settings the majority of algorithmic techniques have been researched and developed under the assumption of identical and independent data distribution (i.i.d). In big data applications data arrives in a stream, and patterns in the data are expected to evolve over time, therefore, it is not practical, and often is not feasible to involve a data mining expert to monitor the performance of the models and to retrain the models every time they become outdated. Therefore, interest towards automating development and update of predictive models in the streaming data settings has been increasing.

Because data is expected *to evolve over time* - especially in dynamically changing environments, where non-stationarity is typical, its underlying distribution can change dynamically over time. The general assumption in the concept drift setting is that the change happens unexpectedly and is unpredictable, although in some particular real-world situations the change can be known ahead of time in correlation with the occurrence of particular environmental events. But solutions for the general case of drift entail the solutions for the particular cases. Moreover the change may take different forms, i.e. the input data characteristics or the relation between the input data and the target variable may change. We briefly summarize the problem of concept drift in the following section.

The traditional supervised learning assumes that the training and the application data come from the same distribution, as illustrated in Figure 2 (a). An online setting brings additional challenges, since it may be expected for the data distribution to change over time. Thus, at any point in time the testing data may be coming from a different distribution than the training data has come, as illustrated in Figure 2 (b).



(a) stationary (b) changing concepts

Figure 2: Supervised learning in (a) stationary and (b) non-stationary environments [172].

In the streaming settings, it is common to expect changes in data and model applicability. Therefore, monitoring of model performance and model update or relearning becomes a natural and core part of the data mining process (Figure 3). The main difference with the standard process is that now data preparation, mining, and evaluation steps are automated, there is no manual data exploration, and there is automated monitoring of performance, including change detection and alert services, after deployment.

4.1 Concept Drift

Concept drift is used as a generic term to describe computational problems with changes over time. These changes may be of countless different types and there are different types of applications that call for different adaptation techniques. Characteristics of change can be summarized with respect to:

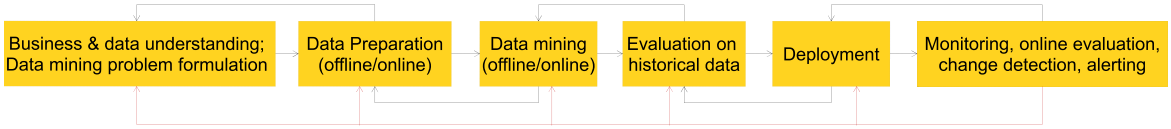


Figure 3: Towards CRISP for Adaptive Data Mining.

- *change source*: adversary activities, changes of preferences, population change, complex environment;
- *change type*: sudden, incremental, gradual, reoccurring;
- *change expectation*: unpredictable, predictable, identifiable (meta);
- *change visibility*: direct/indirect; visual inspection, ground truth.

One of the most illustrative cases, is learning against an adversary (e.g. spam filters, intrusion detection). A predictive model aims at identifying patterns characteristic of the adversary activity, while the adversary is aware that adaptive learning is used, and tries to change the behavior. Another context is learning in the presence of hidden variables. User modelling is one of the most popular learning tasks, where the learning system constructs a model of the user intentions, which are not observable and may change over time. Drift also occurs in monitoring tasks and predictive maintenance. Learning the behaviour of a system (e.g. the quality of products in industrial process) where degradation of equipment occurs over time.

It is not surprising that the problem of concept drift has been studied in several research communities including but not limited to pattern mining, machine learning and data mining, data streams, information retrieval, and recommender systems. Different approaches for detecting and handling concept drift have been proposed in research literature, and many of them have already proven their potential in a wide range of application domains.

In machine learning, data mining and predictive analytics unexpected changes in underlying data distribution over time are referred to as concept drift [112, 235, 192, 247]. In pattern recognition the phenomenon is known as covariate shift or dataset shift [192]. In signal processing the phenomenon is known as non-stationarity [126]. Formally *concept drift* between time point t_0 and time point t_1 can be defined as

$$\exists X : p_{t_0}(X,y) \neq p_{t_1}(X,y),$$

where p_{t_0} denotes the joint distribution at time t_0 between the set of input variables X and the target variable y. Changes in data can be characterized as changes in the components of this relation [148, 114]. In other terms, the prior probabilities of classes $p(y)$ or the class conditional probabilities $p(X|y)$ may change, and as a result, the posterior probabilities of classes $p(y|X)$ *may* change affecting the prediction.

We are interested to know two implications of these changes. First, we are interested to know (i) whether the data distribution $p(y|X)$ changes and affects the predictive decision and (ii) whether the changes are visible from the data distribution without knowing the true labels, i.e. $p(X)$ changes. From a predictive perspective only the changes that affect the prediction decision require adaptation.

We can distinguish the following types of drifts:

1. *Real concept drift* refers to changes in $p(y|X)$. Such changes can happen either with or without change in $p(X)$. Real concept drift has been referred to as *concept shift* in [227] and *conditional change* in [114].

2. *Population drift* refers to changes in the population from which future samples will be drawn compared the design/training sample was drawn[148].

3. *Virtual drift* happens if the distribution of the incoming data changes (i.e., $p(X)$ changes) without affecting $p(y|X)$ [246].

4.2 Adaptive Learning Strategies

Different strategies for updating learning models have been developed. Many classification techniques have been redesigned for streaming settings. Hoeffding trees [84] would be a popular example of decision tree learning on streaming data.

Two main strategies for adaptive learning can be distinguished. Learning models may evolve continuously, for instance, models can be periodically retrained using a sliding window of a fixed size over the past data [247].

Alternatively, learning models may use trigger mechanisms, to initiate a model update. Typically, statistical change detection tests are used as triggers. Incoming data is continuously monitored, if changes are suspected, the trigger issues an alert, and adaptive actions are taken. When a change is signalled, the old training data is dropped and the model is updated using the latest data.

Learning systems can use single models or ensembles of models. Single model algorithms employ only one model for decision making at a time. Once the model is updated, the old one is permanently discarded. Ensembles, on the other hand, maintain some memory of different concepts. The prediction decisions are made either fusing the votes casted by different models or nominating the most suitable model for the time being from the pool of existing models.

Ensembles can be evolving or have trigger mechanisms as well. Evolving ensembles build and validate new models as new data arrives, the rule for model combination is dynamically updated based on the performance (e.g. [188]). Ensembles with triggers proactively assign the most relevant models for decision maxing based on the context (e.g. [236]).

Surveys and categorization of different approaches for handling concept drift can be found in [112, 235, 192].

A closely related settings is *transfer learning*: similarly to the problem of concept drift we anticipate that the data used for model training and data in the application settings come from different distributions and we can observe some of the new data (e.g. we get a new batch of data and can compute how different the data distributions are). A comprehensive overview of transfer learning strategies can be found in [213].

In this section we focused on discussion the data mining approaches that deal with the evolving nature of data disregarding the streaming computing model, i.e. the efficiency of model (re)training and application was not in the spotlight. In the next section we discuss the streaming computing model and data mining approaches that follow it.

5 Streaming: Small-Space Algorithms

5.1 Streaming as Model of Computation

A *data stream* is an ordered sequence of data items, possibly of infinite length, whose data can only be accessed sequentially in forward direction, typically one item at a time. A stream optionally admits multi-pass data traversal, in this case the stream should obviously have finite length. Streams are useful in settings where the entire data set does not fit in memory of the computer used to process it. In more formal words, let $A = (a_1, \ldots, a_m)$ denote a finite-length stream of length m, where each $a_i \in [n]$. Then, the goal in streaming is to process data in space[7] much less than $m \log n$, ideally sub-linear or even logarithmic in n and m.

The streaming model may be viewed as an impediment for a computational task, i.e., certain computations that yield an exact answer in a random-access batch setting are not possible in the streaming setting; approximations have to be used. The challenge in designing algorithms that operate on streams, *streaming algorithms*, is to obtain good approximations while requiring minimal number of passes (ideally just one) and minimal storage of state information (and for practical purposes, minimal computational work, although this is not the primary concern in the streaming setting). The state information in the context of streaming algorithms is commonly called a *sketch* of the data stream.

In the literature, one encounters two models of streaming: the *cash-register* model and the *turnstile* model. To explain these models, it is easiest to generalize our notion of a stream to a sequence of pairs (a_i, k_i), where $k_i \in \mathbb{Z} \setminus \{0\}$ represents the "multiplicity" of a_i. In the cash-register model, which is more common, every k_i is positive. In this case, the pair may just be viewed as a more concise representation (a "runlength encoding") of a number of consecutive identical values in the stream. Furthermore, we can always flatten a cash-register stream (i.e., apply the corresponding runlength decoding) into the representation of a stream defined at the beginning of this section. In the turnstile model, however, the k_i can be *negative* (hence, we wrote multiplicity in quotes). Hence, in the turnstile model the events are either "arrivals" or "departures", depending on the sign of k_i. Just think of an actual turnstile that counts people who either enter or leave a building. In this case, for example, one could determine the number of people that are currently (at time t) inside the building by computing the cumulative sum of the events, i.e., $\sum_{i=1}^{t} a_i k_i$ (of course, under the assumption that the building was empty when the stream started). We will be working in the cash-register model unless we mention otherwise.

5.2 History

The streaming model of computation goes back to papers from the late 1970s by Munro and Paterson [195] and Morris [193], and was motivated by hardware constraints like the particular data-access properties of storage media used at that time (tape) and the limited amount of memory in computers [195], and low-precision arithmetic in embedded systems [193]. The latter work (by Morris) focused on approximate counting of a number of events, denoted by N, using ℓ-bit registers, with $\ell \ll \log_2 N$, hence, a non-trivial problem. Morris' solution involves storing a carefully chosen approximation to the logarithm of the number of counts. Note that due to the scope of this document, we cannot give an exhaustive history of the field of streaming computation; we will highlight some seminal works instead. Flajolet and Martin (FOCS, 1983) [105] were the first to study approximate counting of *distinct* elements in the streaming model. In fact, many other works in streaming focus on various forms of approximate counting. Alon, Matias and Szegedy [13] formalized counting in the streaming

[7] Here we mean *space* in the complexity-theoretic sense.

setting, for which they received the Gödel Prize in 2005. Their results are about the frequency moments of a stream, which are defined as follows. Let $m_i := |\{j \in [m] : a_j = i\}|$ denote the frequency of occurrence of the value i in the stream A, and define for any real number k the *frequency moments* of A as

$$F_k := \sum_{i=1}^{n} m_i^k.$$

Note that using this notion, we can now classify the work of Morris [193] as one that estimates F_1, while Flajolet and Martin [105] estimate F_0. In particular, the work of Flajolet and Martin gave rise to a substantial line of research into approximating F_0, which was finally closed in 2010 by Kane et al. [144]. The number F_2 is known under various names like Gini's index of homogeneity, repeat rate, and self-join size.

Another extensively studied problem in the streaming setting is to estimate the frequencies m_i themselves (rather than the frequency moments). Well-known algorithms in the context of this problem are the *Count Sketch* [56], and the *Count-Min Sketch* [63].

5.3 Lower Bounds for Frequency Moments

Definition 1 *We say that an algorithm $\mathscr{A}$ (ε, δ)-*approximates *X if $\mathscr{A}$ outputs a number $\hat{X}$ such that*

$$\Pr[|\hat{X} - X| > \varepsilon X] < \delta.$$

Another key quantity defined in [13] is

$$F_\infty^* := \max_{i \in [n]} m_i = \lim_{k \to \infty} \sqrt[k]{F_k},$$

i.e., the number of occurrences of the most frequent element. The following result tells us that there is no sub-linear algorithm for approximating F_∞^*

Proposition 1 ([13]) *Any randomized algorithm that, given a sequence A of at most $2n$ elements from the set $[n]$, $(1/3, \delta)$-approximates F_∞^* for some fixed $\delta < 1/2$, must use $\Omega(n)$ bits of space.*

The following result emphasizes the importance of probabilistic (instead of deterministic) algorithms in the streaming setting.

Proposition 2 ([13]) *For any nonnegative integer $k \neq 1$, any deterministic algorithm that, given a sequence A of $n/2$ elements from the set $[n]$, outputs a number Y such that $|Y - F_k| \leq 0.1F_k$ must use $\Omega(n)$ bits of space.*

Furthermore, in the context of frequency moments, approximation rather than exact computation is necessary to achieve sub-linear space:

Proposition 3 ([13]) *For any nonnegative integer $k \neq 1$, any randomized algorithm that, given a sequence A of at most $2n$ elements from the set $[n]$, outputs a number Y such that $Y = F_k$ with probability at least $1 - \varepsilon$ for some fixed $\varepsilon < 1/2$ must use $\Omega(n)$ bits of space.*

5.3.1 Lower Bounds for F_0, F_1, F_2 and F_k

Proposition 4 ([13]) *Let A be a sequence of at most m elements from the set* $[n]$.

(i) Any randomized algorithm for $(0.1, 3/4)$*-approximating* F_0 *must use at least* $\Omega(\log n)$ *bits of space.*

(ii) Any randomized algorithm for $(0.1, 3/4)$*-approximating* F_1 *must use at least* $\Omega(\log\log n)$ *bits of space.*

(iii) Any randomized algorithm for $(0.1, 3/4)$*-approximating* F_2 *must use at least* $\Omega(\log n + \log\log m)$ *bits of space.*

Proposition 5 ([13]) *For any fixed integer* $k > 5$ *and non-negative real number* $\delta < 1/2$*, any randomized algorithm that, given an input sequence A of at most n elements from the set* $[n]$*,* $(0.1, \delta)$*-approximates* F_k *uses at least* $\Omega(n^{1-5/k})$ *bits of space.*

Proposition 6 ([249]) *Any one-pass randomized algorithm that, given a sequence A with elements from the set* $[n]$*,* (ε, δ)*-approximates the k-th frequency moment* F_k *for any real* $k \neq 1$ *and any* $\varepsilon = \Omega(\frac{1}{\sqrt{n}})$ *for some fixed* $\delta < 1/2$*, must use* $\Omega\left(\frac{1}{\varepsilon^2}\right)$ *bits of space.*

5.4 Basic Examples of Streaming

Let us briefly recall the symbols defined in Section 5.1 and 5.2: $A = (a_1, \ldots, a_m)$ represents a stream of elements, where each element $a_i \in [n]$, and m_i denotes the number of occurrences of a_i in A.

5.4.1 Majority

Here we present a simple deterministic, single-pass algorithm by Boyer and Moore [42] for finding the *majority element*, i.e., an element i for which $m_i > m/2$, given the *promise* that there exist such an element. The algorithm maintains a state that consists of the current candidate element and a single counter, hence the algorithm has $O(\log n + \log m)$ space complexity. The algorithm is as follows.

If the promise is absent, then finding the majority element in one round requires linear space. Nonetheless, if one can afford to use a second pass over the stream, then it is possible to determine whether a majority element exists in sub-linear space, by using this second pass to count exactly how many times the candidate element (found in the first pass, using the above algorithm) occurs in the stream.

5.4.2 Distinct Elements

In this section we aim to find a (ε, δ)-approximation to F_0, the number of distinct elements in the stream. As we have seen in Section 5.3, it is impossible to compute F_0 exactly ($\varepsilon = 0$) or accurately estimate F_0 using a deterministic algorithm ($\delta = 0$) in sub-linear space. We will present a randomized algorithm for approximating F_0 from Alon et al. [13]. The algorithm's main building block is a 2-universal hash function.

Definition 2 *A family of hash functions* $\mathscr{H} := \{h_i\}$ *where* $h_i : \mathscr{X} \to \mathscr{Y}$ *for all i, is called* 2-universal *if*

$$\Pr_H\left[H(x) = H(x')\right] \leq \frac{2}{|\mathscr{Y}|} \quad \forall x, x' \in \mathscr{X}$$

where the random variable H is uniformly distributed over the family $\mathscr{H}$.

Data: the stream A, and the promise that A contains a majority element
Result: the majority element (stored in the variable named "candidate")
counter := 0;
candidate := $\perp$;
for *i = 1:m* **do**
  if *counter == 0* **then**
    counter := 1;
    candidate := a_i;
  else
    if *candidate* == a_i **then**
      counter += 1 ;
    else
      counter -= 1 ;
    end
  end
end

Algorithm 1: A Deterministic Algorithm for Majority

Furthermore, let $\text{zeros}(x)$ for any $x \in \mathbb{N}$ be the function that returns the number of trailing zeros in the bit representation of x, i.e.,

$$\text{zeros}(x) = \max\{i : 2^i \text{ divides } x\}.$$

The algorithm is shown as Algorithm 2 and runs in space $O(\log n)$.[8] With respect to the accuracy of the estimate, it can be shown (by applying the Markov inequality and the Chebyshev inequality) that for any $c > 0$, it holds that

$$\Pr\left(\frac{\hat{F}_0}{F_0} \notin \left[\frac{1}{c}, c\right]\right) \leq \frac{2}{c}.$$

This rather weak bound can be easily improved to a $(O(1), \delta)$ approximation, for $\delta > 0$ arbitrary, by running $k = \Theta(1/\delta)$ instances in parallel, and taking the *median* of their outputs. This gives a $O(\log n \cdot \log(1/\delta))$-space algorithm.

Data: the stream A
Result: $\hat{F}_0$, being an estimate of F_0 (the number of distinct elements in A)
$z := 0$;
choose $h : [n] \to [n]$ from a 2-universal family;
for *i = 1:m* **do**
  if $\text{zeros}(h(a_i)) > z$ **then**
    $z := \text{zeros}(h(a_i))$
end
$\hat{F}_0 := 2^z$.

Algorithm 2: The AMS algorithm for estimating F_0 [13]

[8] Here, the $\log n$ factor comes from computing the hash function, storing z only requires $O(\log \log n)$ space.

5.4.3 The Heavy Hitters Problem

In the Heavy Hitters (or frequent items) problem, parameterized by integer $k \geq 0$, the task is to output the set of elements

$$\mathscr{F}_k := \{i \in [n] : m_i > m/k\}.$$

One deterministic two-pass algorithm is based on the *Misra–Gries algorithm*, shown in Algorithm 3, which is a generalization of Algorithm 1 to $k-1$ counters. In the algorithm description, M is an associative array, where $M[x]$ represents the value indexed by key x, and we write keys(M) for the set of keys of M. For positive integer k, the Misra–Gries algorithm returns a set of elements $\mathscr{S}_k$ such that

$$\mathscr{S}_k := \{i \in [n] : m_i - \frac{m}{k} \leq \hat{m}_i \leq m_i\},$$

where $\hat{m}_i := M^{\text{final}}[a_i]$, where M^{final} represents the state of M after completion of the algorithm. It follows that $\mathscr{F}_k \subseteq \mathscr{S}_k$. Hence, in a second pass, we can construct $\mathscr{F}_k$ by counting the frequency of the elements in $\mathscr{S}_k$ exactly, in order to determine if they should be included in $\mathscr{F}_k$.

```
Data: the stream A, parameter k
Result: a set of elements S_k, such that F_k ⊆ S_k.
for i = 1:m do
    if a_i ∈ keys(M) then
        M[a_i] += 1;
    else if |keys(M)| < k − 1 then
        Create M[a_i];
        M[a_i] := 1;
    else
        for x ∈ keys(M) do
            M[k] -= 1 ;
            if M[x] == 0 then
                Remove M[x];
            end
        end
    end
end
S_k := keys(M).
```

Algorithm 3: The Misra–Gries Algorithm

One-pass probabilistic algorithms: the Count Sketch and the Count-Min Sketch To solve the Heavy Hitters problem in one pass over the data (recall that the Misra–Gries-based solution requires two passes), we require probabilistic algorithms. Two well-known hash-function-based algorithms for this problem are the Count-Sketch [56], and the Count-Min Sketch [63]. In fact, these algorithms can estimate the frequency of any symbol appearing in the stream. The performance of both algorithms is compared in Table 4, where $\mathbf{m}_{\bar{i}}$ denotes the vector $(m_j)_{j \in [n] \setminus i}$. Although the Count-Min Sketch is a somewhat simpler algorithm, its analysis yields an error bound in terms of the ℓ^1 norm, whereas the analysis of the Count-Sketch algorithm gives an error bound in terms of the ℓ^2 norm. The latter is more desirable because $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2$ holds for any vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. For more information about these methods, we refer to [52] as well as the original papers.

5.4.4 Frequent Itemset Mining

Given a sequence $B = (b_1, \ldots, b_m)$, where $b_i \subseteq [n]$ for all $i \in [m]$, the *Frequent itemset mining* problem is to find the collection of non-empty subsets

$$\mathscr{G}_t := \{g_j \subseteq [n] : |g_j| \geq 1, |\{i : g_j \subseteq b_i\}| \geq t\}$$

for some threshold $t \in \mathbb{N}$.

A canonical application of frequent itemset mining is in the retail business, where it is used to discover non-trivial (by which we mean: non-common-sense) combinations of products that are often bought together, like diapers and beer.[9] In this retail setting, the items of the universe $[n]$ represent products in a shop, and a set b_i represents a transaction, i.e., the contents of a customer's shopping cart. As an example, Table 5 shows the items bought by four customers, for which, for example, $\mathscr{G}_3 = \{\text{milk}, \text{chocolate}, \text{bread}, \{\text{chocolate}, \text{bread}\}\}$, where we remark that the labels "milk", "bread", etc. are in one-to-one correspondence with the elements of the universe $\{1, \ldots, 5\}$.

In the context of privacy-preserving data mining, another interesting application could be the discovery of new "biomarkers" (like genes and proteins in a patient's blood), which are indicators for particular diseases.

A related problem is *association rule mining*, which are, informally speaking, rules like "if a customer buys milk and bread, then he is likely to buy chocolate as well". Although we will not go into the details of association rule mining, we wish to remark that in order to mine such association rules, one typically first needs to solve the frequent itemset mining problem.

Computing $\mathscr{G}_t$ A key observation that is useful for computing $\mathscr{G}_t$, is the *monotonicity property* for itemsets:

Definition 3 (Monotonicity Property) *If a set $S \in \mathscr{G}_t$, then every non-empty subset of S is also an element of $\mathscr{G}_t$.*

This property is the basis for the *Apriori* algorithm [8]. In particular, the Apriori algorithm requires t passes over the data to compute $\mathscr{G}_t$.

Various other algorithms have been proposed in the literature to decrease the number of passes, and to make use of approximations to lower the space requirements. For an overview, we refer to [168].

Frequent itemset mining in infinite streams When the length of a stream grows with time, or, in other words, the stream never ends, frequent item sets will typically vary over time, hence in the infinite-length-stream setting it may make sense to redefine the meaning of "frequent" relative to a

[9] An explanation for the purchasing-dependence between diapers and beer is that parents of young children typically have their beer at home instead of in the bar.

Table 4: Comparison of the Count Sketch and Count-Min Sketch (reproduced from [52])

Algorithm	Error	Space complexity
Count Sketch	$\Pr(\hat{m}_i - m_i \in [-\varepsilon\|\mathbf{m}_{\bar{i}}\|_2, \varepsilon\|\mathbf{m}_{\bar{i}}\|_2]) \geq 1-\delta$	$O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}(\log m + \log n)\right)$
Count-Min Sketch	$\hat{m}_i \geq m_i, \quad \Pr(\hat{m}_i \leq m_i + \varepsilon\|\mathbf{m}_{\bar{i}}\|_1) \geq 1-\delta$	$O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}(\log m + \log n)\right)$

Table 5: An example to explain frequent itemset mining: customers buying products

	milk	bread	cheese	meat	chocolate
Alice	•			•	
Bob		•	•		•
Charly	•	•		•	•
Daniel	•	•	•		•

given time window. Technically, this usually comes down to letting every counter used by some algorithm *decay* over time, i.e., by multiplying each counter value in every iteration with $1 - \tau$ for some small $\tau > 0$.

5.5 Secure Streaming

Melis et al. [183] apply the count-min and count sketch for counting aggregate quantities in a privacy-preserving manner. They target three different applications: (i) content recommendation systems, (ii) aggregated location monitoring (i.e., aggregated counts of the number of users in each partition of a given area, without revealing the position of individual users) and (iii) securely computing median statistics from servers that run the TOR network. For the first two applications, the aggregation comes down to securely adding values, where users mask their private input in a special way, such that the sum over the masks of all users vanishes. For the third application, a threshold public-key encryption scheme is used to aggregate the sketches. Their constructions are secure in the semi-honest model. In addition to approaching privacy from a cryptography perspective, the authors also analyze the aggregation of sketches from a differential-privacy perspective.

Corrigan-Gibbs and Boneh [64] outline how the work of Melis et al. [183] can be extended to the malicious-adversary setting, using their "Prio" framework.

Burkhart et al. [47] focus on aggregation of network events in a privacy-preserving manner. In particular, they propose a method to count distinct elements, but using a simple (not small-space) algorithm. Their framework, called SEPIA, is based on Shamir secret sharing and is secure in the semi-honest setting.

Bogdanov et al. [35] compute histograms from secret-shared data in Sharemind. The histogram's frequencies are computed securely from secret-shared input data, after which the histogram is opened publicly. The histogram is computed in a straight-forward (non small-space) manner. Additionally, the authors implement the *Apriori* and *Eclat* algorithms for frequent itemset mining in Sharemind.

5.6 Further Reading

We refer to Chakrabarti [52] and Roughgarden [225] for excellent lecture notes on the topic of streaming computation, as well as to the data mining book by Leskovec et al. [168], which also includes a chapter about frequent itemset mining in the streaming setting.

Lower bounds can be found in Alon et al. [13], Bar Yossef [22], Indyk and Woodruff [133] and Woodruff [249].

6 Privacy-Preserving Data Mining

6.1 Multiparty Computation in the Privacy-Preserving Data Mining Landscape

In this section, we give a general overview of the field of privacy-preserving data mining, and the role of multiparty computation in this landscape. All privacy-preserving data mining approaches have the common aim to protect sensitive data used in data mining algorithms. In particular, what these approaches have in common is that they somehow involve data owners, who have data they want to protect, and other parties who want to learn a data mining outcome.

Several variants of privacy-preserving data mining can be identified based on who performs the data mining. In traditional *privacy-preserving data mining*, there is one data owner, and the data mining itself is performed by a different party (Section 6.1.1). In *distributed privacy-preserving data mining*, there are multiple data owners whose combined data needs to be mined in a collaborative fashion, either by the owners themselves or by one or more third parties (Section 6.1.2). The MPC techniques to be developed in the SODA project fall into this category; the state-of-the-art in MPC is given later so in this section we focus on non-MPC techniques. Finally, *privacy-preserving querying* considers the setting where the data mining takes place by the data owner itself, in which case measures need to be taken that the data mining output itself does not reveal too much information about the underlying sensitive dataset. As discussed, this problem can also be considered in setting with multiple data owners, for instance in combination with MPC as the SODA project aims to do with its leakage control component (Section 6.1.3).

Although the SODA techniques fall mostly in the second category of distributed privacy-preserving data mining, awareness of techniques from the two other categories is useful for a succesful execution of the project. Specifically, ideas from privacy-preserving data mining to "transform" sensitive data into a form that can be processed without leaking information to the processor, when implemented well, can help to avoid performing computations fully under encryption, which would be prohibitively slow. Similarly, ideas from protecting computation output can help understand to what extent the decryption of intermediate results in an encrypted computation (such as intermediate versions of a classifier in iterative training techniques) degrades privacy.

6.1.1 Privacy-preserving data mining

In privacy-preserving data mining (PPDM), a data owner transforms its sensitive dataset in such a way that sensitive data is removed, but data mining on the transformed dataset is still possible. Many transformation techniques are designed specifically with a particular kind of analysis in mind. However, there are also techniques that remove sensitive information regardless of what mining algorithm is to be run on the transformed data; this is also known as *privacy-preserving data publishing* (PPDP).

PPDM techniques that are commonly identified in the literature (e.g, [240, 4]), include perturbation, blocking, merging/aggregation, swapping, and sampling. In *perturbation*, noise is added to records in the dataset in such a way that it hides values for individual records, but most of it "cancels out" when performing data mining. The kind of noise to be added and the amount of utility it can preserve, clearly depend on the data mining algorithm at hand; for instance, some data mining algorithms require additive noise while others require multiplicative noise. Applications include classification based on decision trees [7], mining of association rules [102], and clustering [207]. In *blocking*, the idea is not to change sensitive values, but to remove them entirely [54]; this approach can for instance be applied for association rule mining [230]. In *merging/aggregation*, the coarseness of attribute values is increased by combining several values into one category. This technique is generally applicable

but in general obviously reduces accuracy. In *swapping*, different values for attributes from different records are swapped, preserving the overall per-attribute probability distribution but obviously not protecting rare attribute values or relations between different attributes [191, 104]. In *sampling*, rather than releasing data for the full population, only a subset is released.

In PPDP, the goal is to anonymize a dataset regardless of the data mining algorithm to be executed on it. (This is also sometimes referred to as "tabular data protection".) The most well-known PPDP framework is k-anonymity [228]. The core observation of k-anonymity is that not just identifiers can be used to uniquely identify a record, but also combinations of quasi-identifiers such as age, ZIP code, and gender. To remedy this, values for these attributes are *suppressed* (i.e., deleted) or generalized (i.e., replaced by a coarser value such as year of birth instead of date of birth) until the dataset satisfies the k-anonymity criterium: for each combination of quasi-identifiers that occurs in the dataset, there must be at least k entries in the dataset having exactly that combination of values. k-anonymity is a criterion, not a method: different algorithms exist that each try to transform a dataset to meet this criterion while removing as little information as possible. However, doing this optimally is known to be an NP-hard problem [186].

It is well-known that the privacy provided by k-anonymity is limited [180], in response to which several variants such as l-diversity and t-closeness have been developed. A more fundamental, mathematical approach is differential privacy. While primarily developed for privacy-preserving querying, it can also be applied to data publishing, but as discussed in [59], it is not clear if the amount of data that needs to be removed to satisfy this definition still allows the resulting datasets to be used in practice.

6.1.2 Distributed privacy-preserving data mining

Distributed PPDM considers the problem to perform data mining on sensitive datsets from multiple parties when there is no single party that is trusted to hold all of the data. Note that in some cases, aggregates over multiple datasets can be computed from the aggregates of the individual datasets (e.g., the average over multiple datasets is itself a weighted average of the averages of its sub-datasets), but in general this is not the case (as a simple example, think of computing a correlations between two vectors where one data provider has one vector and another data provider has the other vector.

Multiple deployment models can be considered based on who performs the mining. In one model, there is a single miner who collects data from many clients, each of which holds an individual record about itself (e.g., association rule mining in [224]). In another model, a relatively small number of parties each hold a sub-database, and they perform joint data mining by each contributing computing power (e.g., [252] for support vector machines). In a final model, there can be a larger number of parties holding a sub-database, but the joint data mining is outsourced to a smaller number of computing parties (that themselves may or may not provide input) (e.g., [37] for general statistical operations). In each case, the data can be *horizontally partitioned*, meaning that each data provider holds all attribute values for one or more records; or *vertically partitioned*, meaning that there is a common set of records for which each data provider holds a number of attribute values.

The MPC techniques developed in SODA fit into the second and third deployment model by providing cryptographic techniques by which to perform joint data mining without learning the underlying inputs. In theory, every algorithm, so in particular every data mining algorithm can be performed with MPC in order to fully protect the privacy of sensitive inputs. However, because of the large performance penalty that this induces, in practice, often a "hybrid approach" is used in which MPC is combined with local computation by the data providers and part of the computation is performed in-the-clear by opening intermediate computation results that are deemed "sufficiently" aggregated (see Section 6.1.4).

Such hybrid approaches range from on the one hand approaches where the local computation is complex and the MPC aggregation very easy; to on the other hand approaches where the local computation is easy and the bulk of the work takes place inside MPC. Because of efficiency, most approaches are from the first category. For instance, approaches of this kind include the SVM classification protocol of [252] (where each party locally computes a some values that are securely summed up MPC and then opened, so the MPC computation is trivial), the clustering approach from [154] based on local density estimation, and the recent protocol for deep learning by gradient descent from Google [38]. As an example of an approach on the other side of the spectrum, consider the linear programming solver of [73], where the computation is performed fully on encrypted data and only the check whether the stopping criterium has been met is performed in the clear (so the plaintext computation is simple). (Hybrid approaches with a non-trivial MPC step are discussed later in this chapter.)

Several common patterns can be identified in the literature on distributed privacy-preserving data mining. Above, we discussed the combination of local computation on sensitive data, secure aggregation, and in-the-plain processing of the aggregate (in multiple iterations). Instead of using MPC to aggregate the local computation results, it is also possible to instead add noise to (e.g., [101, 224, 89]) or mask (e.g, [217]) the input data or local computation results and then perform aggregation based on this in the plain. Differential privacy can be used to rigorously bound the privacy leakage from opening these intermediate computation results [137, 198]. Many variations on this idea exist. For instance, it is possible to securely perform aggregation and add noise prior to opening the aggregate. Another idea that has been successfully applied to clustering [185] is to use generative models: instead of sharing real records, the records are summarized into a "generative model": parameters to an algorithm to generate representative synthetic records. The recipient then generates synthetic records and uses these in its data mining algorithm instead of the real records. Finally, it is possible to locally train a detailed model but share just a small part of it and combine those parts [239].

An alternative paradigm that has been applied with some success is to use MPC to transform one instance of a problem into a random other instance; solve this other instance in the plain; and transform this back into a solution to the original problem using MPC. This approach has for instance been applied to linear programming [237], in which context also its main weakness has been exposed, namely that it usually impossible to perform the transformation without preserving some of the (sensitive) structure of the original problem [28, 164].

Also the PPDP problem has been considered in the distributed context. For instance, for the main PPDP approach, k-anonymity, the goal is to produce a k-anonymous version of the combined dataset of multiple data providers without actually having to share these datasets; this can be solved with [138, 259] or without encryption [244].

6.1.3 Privacy-Preserving Querying

As discussed, the above approaches focus on protecting sensitive information from the data processor. Clearly, if sensitive data has been removed by the data owner prior to providing the data for processing, as with the approaches from section 6.1.1, then any data mining output will also not contain sensitive information and hence can be disclosed. On the other hand, if mining happens by the data owner without prior deletion of sensitive data, then care needs to be taken that the data mining output does not contain sensitive information. For example, suppose patients are clustered and averages are given over all patients in a cluster. In the extreme case, a patient may end up in a cluster by himself, in which these averages are actually directly personal information. However, also in the less extreme case when a cluster contains only few patients, the averages of the cluster are still statistically revealing a lot of information about each patient in the cluster. The same problem occurs if the mining happens on

encrypted data that is then decrypted for the recipient, as is the case with SODA. To address this problem, a range of techniques for "privacy-preserving querying" (also known as "query editing and inference control" [4]) are available. In SODA, these techniques are relevant for the Leakage control component that aims to address exactly this issue.

The general setting for privacy-preserving querying is where a recipient can perform multiple queries on the same dataset, and tries to learn as much sensitive data as possible by choosing queries that are as revealing as possible. Main approaches to prevent the recipient from doing this include modifying the query result (e.g., by adding noise or removing details) or underlying dataset on which queries are run; and denying particular (combinations) of queries.

Numerous approaches for modifying query results are known. One case that has been studied early on in the literature is association rule mining: in this case, data values in the output association rules can be randomly perturbed [208] or details can be left out [230]. Aproaches like k-anonymity [228] that can be applied to the disclosure of a full dataset, can also be applied in the setting when the outout of a data mining algorithm consists of a number of records.

The main foundational technique that is adopted nowadays for protecting privacy leakage from query results is differential privacy [93]. In this framework, the privacy leakage due to the result of a query is formalized by stating that the probability that the query gives a certain outcome, can not depend too much on the presence of any one particular record in the dataset. Mechanisms satisfying this privacy notion typically achieve it by adding noise to each individual query result. For instance, if the query is the mean age in a population, then the mechanism will first compute the actual age, and then add noise based on a supposed, known distribution of ages. Note that clearly, as the same query is executed multiple times on the sama dataset, each time giving a new value containing some noise, then the overall uncertainty about the result will decrease. This is why differential privacy can applied in the context where there is a "privacy budget": a maximal amount of leakage that is allowed, such that few accurate or many inaccurate queries can be performed before the maximum is reached. Differential privacy has been applied in many data mining applications, including recommender systems [181], decision tree classification [135], and clustering [233]; see [94] for a survey.

When MPC is used to protect sensitive data during processing (as proposed in SODA), it may need to be combined with privacy-preserving query approaches to protect the sensitivity of the computation results coming out of the multi-party computation. In SODA, this is handled by the leakage control component. There is some prior work in this direction. In Rmind, a system to compute statistics over sensitive data using MPC, results are protected by generalization, e.g., by using heat maps instead of showing individual points in plots [34]. (Rmind also implements a form of query auditing by only allowing queries that satisfy a user-supplied policy.) Several other works propose to combine MPC with differential privacy as in SODA [29, 95, 255]: work that SODA aims to improve by applying it to arbitrary corruption models and applying it to the particular data analytics algorithms to be developed in this project.

Another possibility is to modify the underlying dataset prior to answering queries. One simple mechanism to make it harder to infer information about the underlying dataset is to anwser each query on a new random subset of the dataset [78]. Other approaches add noise [9], or represent the data by means of a model such as a pseudo-random sketch and answer queries based on this model [189].

Finally, leakage of sensitive information can be prevented by simply not answering queries that would reveal too much sensitive information, an approach referred to as query auditing [4]. The first works on this topic already appeared in the seventies, where leakage due to sum, mean and median queries [81, 222] were considered. More recent work from the previous decade also considered the problem that the fact that a query was denied itself leaks information, something that can be prevented

by means of so-called simulatable auditing [149]. In this setting, apart from basic statistics, also select-project-join queries on a database have been considered [187]. A further extension to the model also considers queries that may modify the database [196].

6.1.4 PPDM and the Privacy-Utility Trade-Off

In this section, we sketch some results relating to the privacy-utility trade-off often seen in privacy-preserving data mining. Fundamentally, approaches that aim to protect sensitive information by modifying datasets or query results, have the problem that these modifications make the result of data mining less accurate, i.e., they reduce data utility. Adding to this problem is the fact that it is hard to decide how much information needs to be removed before a dataset can be considered sufficiently anonymized, especially given that background information may be available to help, or may become available later.

Several recent examples illustrate the latter point. A famous example is the Netflix dataset, that was anonymized and released to stimulate people to experiment with improvements to Netflix' recommendation algorithm; however, using background information many of these anonymized records could be linked back to Netflix users [199]. Other examples include a dataset of search queries release by AOL that could be traced back to AOL users [23], and a dataset of tennis matches containing suspicious betting patterns that could be unintentionally traced back to the suspect players [79].

More generally, as discussed above, several ad-hoc privacy measures have been criticized for not protecting privacy sufficiently in all cases. For instance, k-anonymity was found to be insufficient in some cases [180], leading to several variants, including l-diversity and t-closeness and others. Similarly, transformation-based approaches to privacy-preserving data mining in the case of linear programming turned out to suffer from fundamental problems [28, 164]. Other examples include [206, 100, 145].

Although differential privacy [93] is a more "fundamental" privacy notion with a rigorous mathematical background, it too is not immune from the privacy-utility trade-off. In fact, depending on the data mining algorithm and data it is applied to, differential privacy may require adding so much noise that the data mining output is not usable anymore [229, 109]. For instance, when computing the mean income per of a US county, differential privacy requires hiding the presence of any individual value from the dataset, including very high incomes. In one case, the true value was \$16,708, but a differentially private query mechanism would deviate from this value by at least \$10,000 88% of times [229]. Apart from this, understanding exactly what privacy differential privacy offers is subtle [150], and there is no broad consensus on how privacy budgets for differential privacy should be chosen [167].

In SODA, the privacy-utility trade-offs posed by opening up intermediate values and applying differential privacy to computation results, make the above discussion particularly relevant. Both imply that, even if computations are perfectly secured with multiparty computation, still some leakage of personal information is unavoidable. Therefore, controlling this leakage and assessing its severity in real-world challenges are important parts of understanding the privacy guarantees of our overall approach.

6.2 Privacy-preserving Data Mining using Multiparty Computation

In this section, we survey the state of the art in the MPC approach to privacy-preserving data mining. For a proper introduction of this field, we refer to Lindell and Pinkas [171], who provide a tutorial-like introduction to MPC-based privacy-preserving data mining with an emphasis on formal security analysis. Further, they discuss specialized constructions for some primitives in the two-party setting, like

set intersection and computing the median. (The motivation for using such specialized constructions is to reduce overhead compared to generic constructions of the same primitive.) Finally, the paper emphasizes that MPC solves only a part of the problem in the context of privacy: the task of deciding which function(s) may or may not be applied to privacy-sensitive data should not be omitted.

One might say that over the last decade a "Cambrian Explosion" has taken place in the field of MPC-based privacy-preserving data mining: many papers emerged, (in particular, too many to cover them all!), and of strongly varying quality. To arrive at a list of candidate papers for our survey, we have started off by making keyword searches in research databases. We then further expanded this list of search results by following references in those papers as well as by adding papers found in different ways. Next, we have filtered this list by removing obviously weak papers. Instead of following the categorical division presented in Section 3, we have created a categorization based on a manual clustering of the main (data-mining-related) topics in the papers. These clusters seem to emerge around applications where privacy is a natural aspect to consider, like classifiers (such as decision trees) in medical applications, as well as recent world-wide hypes (like *deep learning*) or combinations thereof (content recommendation systems, used by Internet firms like Amazon and YouTube). Roughly speaking, our categorization distinguishes between *supervised* methods (classification, regression, etc), *unsupervised* methods (clustering, pattern mining), search/querying/matching problems (e.g., biometric matching, genomic sequence matching), recommendation techniques (matrix factorization) and auxiliary methods from statistics (computing the mean, median, statistical tests, etc.). As a disclaimer, the review should not be regarded as being exhaustive, yet we did our best to cover most of the applied research directions in the field of MPC-based privacy-preserving data mining by means of discussing some of their exemplary papers.

6.2.1 Classification and Regression

Decision Trees Lindell and Pinkas [170] propose a method for securely learning an ID3 decision tree on data that is horizontally partitioned between two parties, such that each party's input remains secret to the other party. An interesting part is a method for securely computing the natural logarithm of a given value, which is accomplished through a Taylor approximation, computed via secure polynomial evaluation. The approach is based on garbled circuits, and the paper presents a formal security analysis in the semi-honest security model.

The work by Du and Zhan [90] is cited in some other papers as one of the early works in the context of private decision-tree learning in the three-party setting. However, their "scalar-product protocol" is described over the reals (completely unrealistic for the purpose of an implementation) and does not come with a security proof.

De Hoogh et al. [74] also focus on the problem of learning decision trees and propose to use a variant of ID3 based on the Gini index, which circumvents the expensive natural logarithmic computation from [170], and hence admits a faster implementation. The authors implement their algorithm in VIFF and and assume honest majority in the semi-honest security model. They apply their oblivious classifier to a real-world healthcare-related prediction problem.

Wu et al. [250] propose protocols for the private *evaluation* (as opposed to learning) of decision trees and random forests of those trees in the two-party setting, with an application to medical diagnosis. The protocols are based on additively-homomorphic encryption and obvious transfer. They cover both the semi-honest and malicious setting.

Bost et al. [41] focus on the private evaluation of several machine-learning primitives in the two-party setting: hyperplane-decision-based primitives, the naive Bayes classifier, and decision trees. A hyperplane-decision classifier consists of a set of k length-d vectors $\mathbf{w}_i \in \mathbb{R}^d$, $i \in [k]$ and computes for

a length-d input $\mathbf{x} \in \mathbb{R}^d$

$$k^* := \arg\max_{i \in [k]} \langle \mathbf{w}_i, \mathbf{x} \rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Many machine learning algorithms can be viewed as instances of hyperplane-decision classifiers, like the perceptron (a basic building block for neural networks), linear discriminant analysis, support-vector machines, and logistic regression. Evaluating the Naive Bayes classifier (under the maximum a-posteriori rule) comes down to computing the arg max over a list of probabilities. To evaluate a decision tree (ID3 or C4.5), the authors first represent the tree as a polynomial, and then evaluate this polynomial. The authors observe that these three machine-learning primitives can be implemented using the following three cryptographic primitives: secure comparison, secure argmax, and secure inner product. The authors use Paillier encryption and leveled homomorphic encryption to implement their protocols, and avoid floating-point arithmetic by scaling all values using a large constant to integers.

Support Vector Machines Vaidya et al. [238, 252] consider an MPC-based privacy-preserving version of a support vector machine (SVM) classifier. The authors cover the cases of horizontally, vertically and arbitrary partitioned data. In the case of horizontally partitioned data, a secure sum is computed using a ring-shaped communication pattern between the players: the accumulator value is passed along the ring, and during the first cycle each player adds a one-time-pad-protected summand to the accumulator. In the second cycle, each player removes the random pad, which finally yields the summed value. In the case of vertically and arbitrary partitioned data, the protocol is based on, respectively, the secure dot product from Goethals et al. [118], and some additively-homomorphic encryption scheme.

Regression Nikolaenko et al. [202] present a method for privacy-preserving linear regression. (More precisely, they focus on *ridge regression*, which is an $\mathscr{L}_2$-regularized variant of linear regression.) Their model of $(n+2)$-party computation closely resembles that of [201] (discussed in Sec. 6.2.8): there are n users, one evaluator and a helper-party called "crypto service provider". Each user i has one datum $(\mathbf{x}_i, y_i)$ where $\mathbf{x} \in \mathscr{A}^d$ is a vector and $y_i \in \mathscr{A}$, for some finite alphabet $\mathscr{A}$. Each user precomputes $A_i := \mathbf{x}_i \mathbf{x}_i^\mathsf{T}$ and $\mathbf{b}_i := y_i \mathbf{x}_i$, encrypts those element-wise using an additively-homomorphic encryption scheme, and submits the resulting ciphertexts to the evaluator. The remaining part of the regression procedure is then executed as a garbled circuit between the evaluator and the helper. This remaining part comes down to summing the ciphertexts that correspond to the same element in the matrix A or vector $\mathbf{b}$, obliviously decrypting the summed ciphertexts, and (obliviously) solving the resulting linear system,[10] for which the authors choose the Cholesky decomposition. They implement the scheme using FastGC [129] and use fixed-point arithmetic for the computations.

In [141], Marc Joye extends on the ideas presented by Nikolaenko et al. [202] by replacing the garbled circuits phase by a blinding with uniformly random $\mathscr{R}, \mathscr{S} \in \mathbf{GL}(d, \mathbb{Z}/N\mathbb{Z})$ by utilizing the homomorphic properties of the encryption. After which the crypto service provider opens the blinded values and solves $\mathscr{R}\mathscr{A}\mathscr{S}\beta' = \mathscr{R}\mathbf{b}$ in the clear. The crypto service provider returns the result β' to the evaluator who extracts the solution to $\mathscr{A}\beta = \mathbf{b}$ as $\beta = \mathscr{S}\beta'$.

Gascón et al. [116] study the same ridge regression problem but consider the data to be horizontally partitioned instead. The main difference with [202] lies in the first phase of the protocol in which user participation during the online phase is required, while the second phase remains largely the

[10] Here, we omitted the fact that the evaluator adds $\lambda \cdot I$ to A before solving the system, where λ is a positive scalar and I is the $d \times d$ identity matrix.

same. For the first phase, each pair of parties performs a 2-party secure dot product protocol to compute (2-party) shares of $a_{i,j} = \mathbf{x}_i^T \mathbf{x}_j$ and $b_i = \mathbf{x}_i^T \mathbf{y}$. Note that this generally involves $\mathcal{O}(d^2)$ interactive secure dot product computations (where d is the number of features). Therefore, no scalability issues occur when n increases to one million records. For the second phase, Gascón et al. study *fixed point conjugate gradient descent* (FP-CGD) as potential alternatives to Cholesky's decomposition, which is used in [202]. They conclude that their FP-CGD algorithm, tailored for fixed-point computation, scales better in the number of features than Cholesky.

Graepel et al. [122] consider machine learning tasks computed by a cloud service provider on confidential data. The confidentiality of the data is protected by a leveled homomorphic encryption (LHE) [43, 44], which is a version of somewhat homomorphic encryption (SHE) for which noise only grows polynomial in the levels (circuit depth) of multiplications. This approach portraits some of the difficulties commonly encountered in the field of secure computation. It should be noted that the authors decide against the use of fully homomorphic encryption (FHE) [117], which allows an arbitrary number of operations on ciphertexts, because it is known to come with a rather costly bootstrapping procedure. They define the notion of a D-polynomial learning/prediction algorithm, by which they refer to machine learning functions (including a training phase) which are polynomial of low-degree D. The resulting construction is limited however, since it disallows the use of certain common primitives, e.g. secure comparison (unless the inputs are already bit-wise encrypted), and secure division. Instead they focus on approximation (e.g. truncated Taylor series) to accommodate for the limitations of the setting. Graepel et al. present concrete measurements of implementation of Linear Means classification and approximate Fisher's Linear Discriminant classifier. Communication cost are excluded from their experiments and real numbers are converted to decimal fixed-point with scaling factor of 10^2 and encoded as binary polynomials. The experiment results, as is expected with homomorphic encryption, show a significant cost in computation time, even for a limited number of features and entries. For example, the approximate Fisher's Linear Discriminant classifier computed on 5 features and 100 entries for both training and testing takes roughly 5.6 hours to train, 10 seconds to classify, and 27 minutes for input encoding and encryption. While in the clear these numbers are reported as roughly 2.6E-3 seconds for training and 1.72E-6 seconds for classification.

Bos et al. [39] focus on the private evaluation of logistic regression models in the two-party setting with leveled homomorphic encryption. The models are evaluated by computing Taylor approximations. Arithmetic over real numbers is simulated by working over the rationals.

6.2.2 Neural Networks

Barni et al. [25] focus on evaluating neural networks in the two-party semi-honest-security setting. They rely on three primitives, namely scalar-product, comparison and polynomial evaluation, and instantiate those, respectively, based on a homomorphic encryption scheme, garbled circuits, and oblivious transfer.

Liu et al. [174] propose MiniONN (Mini Oblivious Neural Network), an approach for making oblivious predictions using a existing neural-network model. The paper is in the two-party setting; it considers a client and a server in the cloud. When evaluating the network to obtain predictions, the client's input as well as the prediction output remain secret to the server, while the model remains secret to the client, beyond what is revealed by the prediction output. I.e., the prediction output inevitably leaks some information about the model. The authors assume that the network has already been trained (as their objective is to perform oblivious prediction with existing neural-network models), and do not consider the problem of training the network in an oblivious way. MiniONN employs fixed point computation (computations over the integers with truncations). The authors make extensive use

of "dot product"-triples and propose a homomorphic-encryption-based dot-product-triple-generation protocol suitable for SIMD (single-instruction multiple data) batch processing. This protocol is proven secure against semi-honest adversaries (either client or server). The authors have made a test implementation is based on ABY [76], which offers a "hybrid" MPC framework (a mix of secret-sharing and garbled circuits), and YASHE [40] and SEAL [87] (for homomorphic encryption).

Rouhani et al. [226] also focus on the evaluation (rather than training) of a neural network in a client–server setting. They focus on evaluating convolutional and deep neural nets using garbled circuits, and design circuits for basic neural-net functionalities like convolution (implemented as a secure weighted sum), max- and mean-pooling (implemented resp. as secure max and mean computation) and for computing non-linearities: hyperbolic tangent, sigmoid, softmax (implemented with a CORDIC) and ReLu. Security holds in the honest-but-curious model (but can be extended to malicious secure with known techniques.)

Bonawitz et al. [38] design a constant round protocol to train a neural network by means of federated learning. Here, federate learning means that each user trains a model locally (on a local dataset), after which the trained model parameter vector x_u of each user u is aggregated to a centralized server and summed to obtain a joint model $\sum_{u\in U} x_u$. The aggregation should protect the privacy of the users' data: this means that the server must not learn anything about the model parameters from individual users, beyond what can be deduced from the aggregated model, i.e. the sum $\sum_{u\in U} x_u$. This privacy is achieved by the use of pairwise one-time-pads $p_{u,v}$ between each pair of users u, v such that $p_{u,v} + p_{v,u} = 0 \pmod R$ [2]. Specifically, the authors consider a setting in which users participate in the protocol via a mobile device. In this setting they require robustness against users dropping out during the protocol. As in [121], they achieve this robustness property through the use of threshold secret sharing so that masks from dropped users can be reconstructed as long as a minimal number of users remains. Unlike [121], Bonawitz et al.'s recovery method is robust even if additional users drop out during the recovery phase. The authors present a security proof against semi-honest adversaries in the plain model, and a proof for the malicious case in the random oracle model. Privacy against malicious adversaries requires an additional round in order to perform a consistency check.

6.2.3 Search Problems

Osadchy et al. [210] propose a system for privacy-preserving face recognition in the two-party client–server setting. The ideal functionality that is realized by the system takes as input from the client a single photo p of a face, and from the server a list L of faces, and outputs a bit telling whether there exists a photo in L that closely resembles p. A key idea from the paper is to propose a novel representation for a facial image. For this representation, a face is divided into a number of predefined parts (nose, eyes, mouth, etc.) and for each part there is a predefined list of features, called *words* in the paper, where each such word is either present or absent in a given facial image. Then, a facial image can then be represented as a set of words from various image parts that are present. Under this representation, and by representing sets as binary vectors, the task of comparing two images reduces to computing a Hamming distance—an operation that can be quite easily implemented as a secure computation. Osadchy et al. use additively-homomorphic encryption and oblivious transfer. Their construction is secure against semi-honest adversaries.

Erkin et al. [96] also propose a semi-honest, privacy-preserving, face recognition system for the two-party client-server setting. The server owns m database images represented as n dimensional vectors. They assume that the server performs Principal Component Analysis (in the clear) beforehand on the training images and pre-computes the (orthonormal) eigenvectors $u_1, \ldots, u_k$ corresponding to the $k(\ll m)$ largest eigenvalues. The server also pre-computes (again in the clear) the feature vec-

tor projections $\Omega_1, \ldots, \Omega_M$ of the database images into the subspace $span\{u_1, \ldots, u_k\}$. In the online phase, the authors use the homomorphic properties of Paillier encryption [212] to securely project the (encrypted) input face image into $\bar{\Omega} \in span\{u_1, \ldots, u_k\}$. Note that the u_i are considered sensitive data, as they represent a significant portion of the information in the database. After projection, the square Euclidean distance between $\bar{\Omega}$ and every database image Ω_i is computed. The protocol finishes with secure comparison, based on DGK encryption [68, 69], to determine whether any of the distances is shorter than a certain threshold T, which will be considered a match. The authors implemented their approach with a database size of $m = 320$ images with vector length $n = 10304$. The two parties are implemented within the same machine as separate threads, so there is no network latency in their experiments. Querying one image is reported to take roughly 40 seconds. When the (computationally heavy) randomization factors of the homomorphic encryptions are pre-computed, the online execution time drops to 18 seconds.

The iDASH workshops [131] are annual competitions for privacy and security of genomic data. One of the challenges of 2016's iDASH competition was to devise a privacy-preserving approximation algorithm for the *k-closest match problem*, defined as follows. Given a database with genomic sequences $S_1, \ldots, S_m$ and a user query sequence Q, find (an approximation to) the indices of the k database sequences with smallest edit distance to Q. Edit distance computation between two sequences of length n typically runs in $\mathcal{O}(n^2)$ time [62]. This motivates the use of approximations, such as sequence partitioning methods, which approximate edit distance by comparing short sequence blocks. In [15] such a sequence partitioning approximation is proposed, which runs in near-linear time. Nonetheless, for the k-closest-match problem, this still doesn't scale well enough in terms of the database size m. Asharov et al. [17] won this particular iDASH challenge with an approximation algorithm secure against semi-honest adversaries. The straightforward solution of fixed-length blocks results in a poor approximation, because insertions and shifts in the sequence (especially at the start of the sequence) can cause major misalignment issues. Hence, Asharov et al.'s solution starts by splitting the genomic sequences S_i and query Q into blocks $\{S_{i,1}, \ldots, S_{i,n}\}$ and $\{Q_1, \ldots, Q_n\}$ of *varying block size*, determined via the Wagner-Fischer algorithm [241] with the help of a public reference sequence R. They then approximate the edit distance by $\sum_l \chi_l \cdot \mathrm{ED}(Q_l, S_{i,l})$ where $\chi_l = 1$ if $Q_l \in T_l := \{S_{1,l}, \ldots, S_{m,l}\}$ and 0 otherwise. The approximation only adds a block edit distance $\mathrm{ED}(Q_l, S_{i,l})$ to the sum if the block Q_l lies in the column T_l, this allows them to pre-compute edit distances (in the clear) within a column to speed up the computation. This also means that for a good approximation, it is important for R to be sufficiently close to the database queries, which implies that such a choice for R inevitably leaks some information about the database sequences. In addition, a chosen-plaintext attack can fully recover the block values T_l. Hence, the approximation is not a *secure approximation* in the sense of Feigenbaum et al. [103], because the approximation leaks more information than the original computation. The authors report real-world performance in terms of accuracy and speed. They consider databases with higher variability ($\approx 5\%$) than a previous solution by Wang et al. [245] ($\approx 0.5\%$).

6.2.4 Pattern Mining

In [36], Bogdanov et al. implement four privacy preserving algorithms for frequent itemset mining, as a proof of practical applicability of their *Sharemind* framework. In all algorithms they search for all sets χ which are frequent (i.e. $\mathrm{support}(\chi) \geq t$), while the database rows/columns (depending on horizontal or vertical partitioning) are secret shared. A key observation is that every subset of a frequent set is frequent. This observation leads to the breadth-first *Apriori* algorithm [8, 178] and the depth first *Eclat* algorithm [253]. For both aforementioned algorithms they also implement a hybrid algorithm, *Hyb-Apriori* performs better in terms of memory footprint at a very slight cost

of performance and *Hyb-Eclat* has slightly increased performance at the cost of a small increase in memory consumption. The Sharemind framework is equipped with three nodes/parties for the multi-party computation, which is proven to be secure in the semi-honest security setting separately in [37].

6.2.5 Clustering

Jagannathan and Wright [136] introduce the (two-party) concept of *arbitrary partitioned data* serving as a generalization of vertical and horizontal partitioned data. Arbitrary data partitioning, in the two party setting, means that every database entry $x_{i,j}$ belongs to either Alice or Bob. So every transaction/object $t_i = \{x_{1,j}, \ldots, x_{i,l}\}$ is split between the two parties: $t_i = t_i^A \cup t_i^B$. Note that if $t_g^B = \emptyset$, then $t_g = t_g^A$ belongs completely to Alice, and Bob is not aware of the existence of transaction g. The authors present a two-party privacy-preserving k-means clustering protocol on arbitrary partitioned data in the semi-honest setting. In each iteration, the protocol assigns each database object to each candidate cluster and new candidate cluster means are computed accordingly. The cluster means μ_i are (additive) shared between the two parties. The shared distance between an item and cluster mean $[d(t_i, \mu_j)]$ is largely computed locally, after which the index of the minimal mean distance cluster is located via a garbled circuit. Cluster means are updated in a similar fashion: the two parties compute local sums of cluster object entries and input the result in a garbled circuit which outputs the new shared cluster mean $[\mu_j]$, alternatively they can compute the new shared cluster mean via a secure inversion and a secure multiplication using Beaver triples [21]. The k-means clustering algorithm stops when there are no more substantial improvements. So after each iteration, the protocol checks whether the cluster centers have significantly changed via another garbled circuit secure comparison. the protocol leaks information (as the authors remark in their performance analysis) because it reveals intermediate information, namely the assigned cluster number of the objects after every iteration. This information leaks the amount of objects and can also leak individual party data.

In [46], Bunn and Ostrovsky also present a two-party privacy-preserving k-means clustering protocol on arbitrary partitioned data. Their main contribution consists of a construction which does not leak intermediate information. Their protocol refrains from releasing the object cluster numbers at the end of an iteration. Bunn and Ostrovsky take the single database k-means algorithm as described in [211], which has provable correctness, and extend it to the privacy-preserving two party setting in the semi-honest security model. Bunn and Ostrovsky hide the intermediate information by computing the new cluster means μ_j under a homomorphic encryption.

Clifton et al. [169] instantiate expectation–maximization (EM) mixture clustering [139, 77], in a privacy-preserving and distributed way, by making use of Benaloh's secure sum protocol [32]. EM mixture clustering is a form of probabilistic clustering by means of maximum likelihood parameter estimation. They assume the data is horizontally partitioned. As in [136], every iteration reveals assigned cluster number of the objects. So intermediate information is leaked within the protocol. In practice this means that the protocol requires at least three parties and no collusion to be secure.

6.2.6 Secure Statistics

Kiltz et al. [151] discuss securely computing the mean in the two-party setting, and moreover they point out a flaw in a protocol for securely computing the mean by Du and Attalah [88]. Aggarwal et al. [5] propose a method for securely computing the median (or, more generally, the k-the ranked element. DeBenedetto et al. [75] focus on the secure computation of standard deviation and Chi-squared tests. They implement their methods in PICCO [257], a threshold-linear-secret-sharing-based MPC compiler framework.

6.2.7 Oblivious Data Mining on Graphs

Meng et al. [184] study the problem of answering shortest-distance queries on large *encrypted* graphs (size in the order of millions of vertices) in the client–server model, where the encryption is of a special kind such that it allows for privately querying the distance between two vertices. Actually, the authors focus on the related problem of answering shortest-distance queries *approximately* (for example, within a constant factor), using the concept of *distance oracles*. The paper uses a particular class of distance oracles called *sketch-based* distance oracles (recall that we already encountered sketches in Section 5), which precompute a sketch sk_v for each node v, such that in the online phase, the approximate shortest distance between two nodes u and w can be efficiently computed from their sketches sk_u and sk_w. The authors adapt an existing adaptive semantic security notion to the setting of distance-oracle encryption and formally define the types of leakage that can occur. They also propose three distance-oracle encryption schemes: one scheme with low computational complexity through the use of symmetric-encryption primitive, one scheme based on somewhat-homomorphic encryption having optimal communication complexity, and a third scheme that enjoys low computational as well as low communication complexity at the cost of some privacy leakage. The security of all schemes is analyzed formally. Finally, the author have implemented the schemes and run some experiments. We give a rough characterization of the performance of the third scheme: (a) sketch construction time (per node): 20ms, (b) sketch size (per node) 20 KB, (c) communication per query: 34 bytes.

6.2.8 Recommender Systems

Nikolaenko et al. [201] present an $(n+2)$-party protocol for an oblivious recommender system based on matrix factorization, where the $n+2$ parties consist of n users, one recommender and one "helper party" (named crypto service provider). In this paper, the work "oblivious" means that the recommender does not learn the ratings of the users, nor which items each user has rated. The protocol is based on (two-party) garbled circuits, executed between the recommender and the helper, and on a *partially homomorphic* encryption scheme, used by the users to encrypt their ratings. A partially homomorphic encryption scheme supports the addition of (public) constants under the encryption; note that this is a strictly weaker notion than an additively-homomorphic encryption scheme. The main body of the paper assumes the honest-but-curious security model, but outlines how the protocol can be extended to achieve active security. The matrix factorization method is regularized least-squares minimization, via gradient descent. An important contribution of the paper is a trick to prevent $O(nm)$ complexity (where m are the number of items) in each gradient-descent iteration by the use of Batcher's sorting network. With this technique, the asymptotic complexity becomes $(m+M)\log^2(m+M)$ per iteration, where M is the total number of ratings (from all users). The protocol has been implemented in FastGC [129], and uses fixed-point arithmetic. The authors perform simulations to measure execution times and explore the trade-off between accuracy and bitsize of the fixed-point representation.

7 Conclusion

SODA's Work Package 2 will focus on privacy-preserving data mining *from the perspective of cryptography*, or more precisely, *secure multiparty computation* (MPC), as opposed to the (commonly studied) alternative perspective of data perturbation, such as differential privacy. The present document attempts to survey the current state of the art in this field.

To put this survey on solid grounds, we have started this document by reviewing primitives and preliminaries from the field of secure multiparty computation. In that part, it already becomes clear that there are several mismatches between results from the field of MPC (and features provided by MPC frameworks), and the typical requirements from data mining methods. Arithmetic in MPC is provided naturally over a finite ring or finite field, while in data mining (or, if you wish, numerical mathematics like computational linear algebra) computations are often assumed to be performed in floating point arithmetic. Many results in the MPC literature about linear algebra are over some finite field, instead of over (some approximation of) the reals. Furthermore, in MPC the relative ordering of operations (addition, multiplication, comparison, etc.) in terms of their *strength* (a term from compiler technology, by which we mean a quantity that is proportional to the execution time of an operation, see also [60]) is very different from an ordinary CPU. In MPC, the strength of an operation is not only determined by its computational complexity, but also depends on the MPC paradigm (garbled circuits, arithmetic secret-sharing, homomorphic encryption) that is used, which in turn has a strong influence on the round complexity of basic operations. Moreover, latency and throughput depend on the particular type, topology and geographical aspects of the network that interconnects the players. Hence, design decisions deeply rooted in (data mining) algorithms might need to be revisited when such an algorithm is adapted to be run as a multiparty computation, because those decisions were originally made with the goal in mind to run the algorithm on an ordinary computer.

In Section 3, we have introduced data mining and machine learning, by first giving an overview of the main data mining tasks. With the help of Domingos' categorization [83], we have aimed to connect those tasks to the wide variety of existing data mining methods described in the literature. We have covered the "data mining process", i.e., all steps between a high-level question about data and, ultimately, the practical deployment of a data-processing system. And we discussed classes of applications and their important aspects for data mining.

A particular topic in the field of data mining is the study of *data streams*, which we discussed in Sections 4 and 5. A main theme in this subfield is to design algorithms that are capable of processing lots of data while only having a small memory footprint and, typically, having a low computational complexity as well. Especially the latter aspect aligns well with the requirements of running an algorithm as a multiparty computation.

Finally, in Section 6, we have surveyed the state of the art in the MPC approach to privacy-preserving data mining. We did our best to review relevant papers from the recent "Cambrian Explosion" of papers in the literature, and to cover most of the applied research directions within the field by means of discussing some of their exemplary papers. Nonetheless, an exhaustive review of all papers would have been impossible within the scope of this survey, simply because there are by now already too many papers to mention. Interestingly, there seem to some gaps in the literature in the direction of secure ensemble methods (like secure boosting) and secure variants of Bayesian inference[11] like Markov-Chain Monte Carlo methods and variational inference techniques. Also, we have only seen a few works in the literature that consider the streaming setting in the context of MPC. Hence, we see ample opportunities in the SODA project to perform further research into these directions!

[11] Here, we do not mean the "Naive Bayes" classifier.

References

[1] Shigeo Abe. *Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[2] Gergely Ács and Claude Castelluccia. *I Have a DREAM! (DiffeRentially privatE smArt Metering)*, pages 118–132. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[3] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. *SIGMOD Rec.*, 28(2):61–72, June 1999.

[4] Charu C. Aggarwal and Philip S. Yu, editors. *Privacy-Preserving Data Mining - Models and Algorithms*, volume 34 of *Advances in Database Systems*. Springer, 2008.

[5] Gagan Aggarwal, Nina Mishra, and Benny Pinkas. Secure computation of the k th-ranked element. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 40–55. Springer, 2004.

[6] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data. *Data Min. Knowl. Discov.*, 11(1):5–33, July 2005.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.*, pages 439–450. ACM, 2000.

[8] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[9] Rakesh Agrawal, Ramakrishnan Srikant, and Dilys Thomas. Privacy preserving olap. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 251–262, New York, NY, USA, 2005. ACM.

[10] Miklós Ajtai, János Komlós, and Endre Szemerédi. An 0 (n log n) sorting network. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 1–9. ACM, 1983.

[11] Mehrdad Aliasgari and Marina Blanton. Secure computation of hidden markov models. In *Security and Cryptography (SECRYPT), 2013 International Conference on*, pages 1–12. IEEE, 2013.

[12] Mehrdad Aliasgari, Marina Blanton, Yihua Zhang, and Aaron Steele. Secure computation on floating point numbers. In *NDSS*, 2013.

[13] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 1996.

[14] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, ODD '13, pages 8–15, New York, NY, USA, 2013. ACM.

[15] Alexandr Andoni and Krzysztof Onak. Approximating Edit Distance in Near-linear Time. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 199–204, New York, NY, USA, 2009. ACM.

[16] Yuriy Arbitman, Moni Naor, and Gil Segev. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 787–796. IEEE, 2010.

[17] Gilad Asharov, Shai Halevi, Yehuda Lindell, and Tal Rabin. Privacy-Preserving Search of Similar Patients in Genomic Data. Cryptology ePrint Archive, Report 2017/144, 2017.

[18] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 535–548. ACM, 2013.

[19] Big Data Value Association. Big Data Value Strategic Research and Innovation Agenda version 3.0. bdva.eu, 2017.

[20] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 671–680, New York, NY, USA, 2014. ACM.

[21] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, PODC '89, pages 201–209, New York, NY, USA, 1989. ACM.

[22] Ziv Bar-Yossef and Christos H Papadimitriou. *The complexity of massive data set computations*. PhD thesis, University of California, Berkeley, 2002.

[23] M. Barbaro and T. Zeller. A face is exposed for aol searcher no. 4417749. The New York Times, http://select.nytimes.com/gst/abstract.html?res=F10612FC345B0C7A8CDDA10894DE404482, 2006.

[24] Erwin H Bareiss. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of computation*, 22(103):565–578, 1968.

[25] M. Barni, C. Orlandi, and A. Piva. A Privacy-preserving Protocol for Neural-network-based Computation. In *Proceedings of the 8th Workshop on Multimedia and Security*, MM&Sec '06, pages 146–151, New York, NY, USA, 2006. ACM.

[26] Kenneth E Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 307–314. ACM, 1968.

[27] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 479–488. ACM, 1996.

[28] Alice Bednarz, Nigel Bean, and Matthew Roughan. Hiccups on the road to privacy-preserving linear programming. In Ehab Al-Shaer and Stefano Paraboschi, editors, *Proceedings of the 2009 ACM Workshop on Privacy in the Electronic Society, WPES 2009, Chicago, Illinois, USA, November 9, 2009*, pages 117–120. ACM, 2009.

[29] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 451–468. Springer, 2008.

[30] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 585–591, Cambridge, MA, USA, 2001. MIT Press.

[31] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *J. Mach. Learn. Res.*, 2:125–137, March 2002.

[32] Josh Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Advances in Cryptology-CRYPTO'86*, pages 251–260. Springer, 1987.

[33] Marina Blanton and Everaldo Aguiar. Private and oblivious set and multiset operations. *International Journal of Information Security*, 15(5):493–518, 2016.

[34] D. Bogdanov, L. Kamm, S. Laur, and V. Sokk. Rmind: a tool for cryptographically secure statistical analysis. PP(99):1–1.

[35] Dan Bogdanov, Roman Jagomägis, and Sven Laur. Privacy-preserving Histogram Computation and Frequent Itemset Mining with Sharemind. Technical Report T-4-8, Cybernetica, 2009.

[36] Dan Bogdanov, Roman Jagomägis, and Sven Laur. A Universal Toolkit for Cryptographically Secure Privacy-Preserving Data Mining. In Michael Chau, G. Alan Wang, Wei Thoo Yue, and Hsinchun Chen, editors, *Intelligence and Security Informatics - Pacific Asia Workshop, PAISI'12, Kuala Lumpur, Malaysia, May 29, 2012. Proceedings*, volume 7299 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2012.

[37] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. *Computer Security-ESORICS 2008*, pages 192–206, 2008.

[38] 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. Cryptology ePrint Archive, Report 2017/281, 2017.

[39] Joppe W Bos, Kristin Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. *Journal of biomedical informatics*, 50:234–243, 2014.

[40] Joppe W Bos, Kristin E Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In *IMA Int. Conf.*, pages 45–64. Springer, 2013.

[41] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, 2015.

[42] Robert S Boyer and J Strother Moore. Mjrty-a fast majority vote algorithm. In *Automated Reasoning*, pages 105–117. Springer, 1991.

[43] Zvika Brakerski. Fully Homomorphic Encryption Without Modulus Switching from Classical GapSVP. In *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, pages 868–886, New York, NY, USA, 2012. Springer-Verlag New York, Inc.

[44] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[45] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[46] Paul Bunn and Rafail Ostrovsky. Secure Two-Party k-Means Clustering. Cryptology ePrint Archive, Report 2007/231, 2007.

[47] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. *Network*, 1:101101, 2010.

[48] Octavian Catrina and Sebastiaan de Hoogh. Improved primitives for secure multiparty integer computation. In *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, pages 182–199, 2010.

[49] Octavian Catrina and Sebastiaan de Hoogh. Secure multiparty linear programming using fixed-point arithmetic. In *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings*, pages 134–150, 2010.

[50] Octavian Catrina and Claudiu Dragulin. Multiparty computation of fixed-point multiplication and reciprocal. In *Database and Expert Systems Applications, DEXA, International Workshops, Linz, Austria, August 31-September 4, 2009, Proceedings*, pages 107–111, 2009.

[51] Octavian Catrina and Amitabh Saxena. Secure computation with fixed-point numbers. In Radu Sion, editor, *Financial Cryptography and Data Security: 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers*, pages 35–50, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[52] Amit Chakrabarti. Cs49: Data stream algorithms: Lecture notes, fall 2011, 2011. Last Update: October 27, 2015.

[53] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[54] Liwu Chang and Ira S. Moskowitz. An integrated framework for database privacy protection. In Bhavani M. Thuraisingham, Reind P. van de Riet, Klaus R. Dittrich, and Zahir Tari, editors, *Data and Application Security, Development and Directions, IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security, Schoorl, The Netherlands, August 21-23, 2000*, volume 201 of *IFIP Conference Proceedings*, pages 161–172. Kluwer, 2000.

[55] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. CRISP-DM 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, 2000.

[56] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, ICALP '02, pages 693–703, London, UK, UK, 2002. Springer-Verlag.

[57] Shuo Chen, Rongxing Lu, and Jie Zhang. A flexible privacy-preserving framework for singular value decomposition under internet of things environment. In *Trust Management XI*, IFIP Advances in Information and Communication Technology, pages 21–37. Springer, Cham.

[58] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, New York, NY, USA. ACM.

[59] Chris Clifton and Tamir Tassa. On syntactic anonymity and differential privacy. *Trans. Data Privacy*, 6(2):161–183, 2013.

[60] John Cocke and Ken Kennedy. An algorithm for reduction of operator strength. *Commun. ACM*, 20(11):850–856, November 1977.

[61] William W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, ICML'95, pages 115–123, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[62] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[63] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, April 2005.

[64] Henry Corrigan-Gibbs and Dan Boneh. Prio: Private, robust, and scalable computation of aggregate statistics. *CoRR*, abs/1703.06255, 2017.

[65] Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In *Annual International Cryptology Conference*, pages 119–136. Springer.

[66] Ronald Cramer, Eike Kiltz, and Carles Padró. A note on secure computation of the moore-penrose pseudoinverse and its application to secure linear algebra. In *Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO'07, pages 613–630. Springer-Verlag.

[67] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000.

[68] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. *Efficient and Secure Comparison for On-Line Auctions*, pages 416–430. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[69] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. A correction to "Efficient and Secure Comparison for On-Line Auctions". Cryptology ePrint Archive, Report 2008/321, 2008.

[70] Ivan Damgård, Sigurd Meldgaard, and Jesper Nielsen. Perfectly secure oblivious ram without random oracles. *Theory of Cryptography*, pages 144–163, 2011.

[71] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *Asiacrypt*, volume 6477, pages 213–231. Springer, 2010.

[72] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography*, volume 10, pages 143–159. Springer, 2010.

[73] Sebastiaan de Hoogh. *Design of large scale applications of secure multiparty computation: secure linear programming*. PhD thesis, Eindhoven University of Technology, 2012.

[74] Sebastiaan de Hoogh, Berry Schoenmakers, Ping Chen, and Harm op den Akker. *Practical Secure Decision Tree Learning in a Teletreatment Application*, pages 179–194. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[75] Justin DeBenedetto and Marina Blanton. Optimizing secure statistical computations with PICCO. *CoRR*, abs/1612.08678, 2016.

[76] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[78] Dorothy E. Denning. Secure statistical databases with random sample queries. *ACM Trans. Database Syst.*, 5(3):291–315, September 1980.

[79] Nicholas Diakopoulos. Buzzfeed's pro tennis investigation displays ethical dilemmas of data journalism. Columbia Journalism Review, https://www.cjr.org/tow_center/transparency_algorithms_buzzfeed.php, 2016.

[80] Vassil Dimitrov, Liisi Kerik, Toomas Krips, Jaak Randmets, and Jan Willemson. Alternative implementations of secure real numbers. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 553–564. ACM, 2016.

[81] David P. Dobkin, Anita K. Jones, and Richard J. Lipton. Secure databases: Protection against user influence. *ACM Trans. Database Syst.*, 4(1):97–106, 1979.

[82] Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, October 2012.

[83] Pedro Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books, New York, 2015.

[84] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.

[85] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Mach. Learn.*, 29(2-3):103–130, November 1997.

[86] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 789–800. ACM, 2013.

[87] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Manual for using homomorphic encryption for bioinformatics. *Proceedings of the IEEE*, 105(3):552–567, 2017.

[88] Wenliang Du and M. J. Atallah. Privacy-preserving cooperative statistical analysis. In *Seventeenth Annual Computer Security Applications Conference*, pages 102–110, Dec 2001.

[89] Wenliang Du and Justin Zhijun Zhan. Using randomized response techniques for privacy-preserving data mining. In Lise Getoor, Ted E. Senator, Pedro M. Domingos, and Christos Faloutsos, editors, *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 505–510. ACM, 2003.

[90] Wenliang Du and Zhijun Zhan. Building Decision Tree Classifier on Private Data. In *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining - Volume 14*, CRPIT '14, pages 1–8, Darlinghurst, Australia, Australia, 2002. Australian Computer Society, Inc.

[91] R. Duda, P. Hart, and D. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[92] Wouter Duivesteijn, Ad J. Feelders, and Arno Knobbe. Exceptional model mining: supervised descriptive local pattern mining with complex target concepts. *DATA MINING AND KNOWLEDGE DISCOVERY*, 30(1):47–98, 2016.

[93] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[94] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Ding-Zhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2008.

[95] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006.

[96] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-Preserving Face Recognition. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*, PETS '09, pages 235–253, Berlin, Heidelberg, 2009. Springer-Verlag.

[97] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[98] Vernon Turner et al. The digital universe of opportunities. IDC Research, 2014.

[99] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[100] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting Privacy Breaches in Privacy Preserving Data Mining. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 211–222, New York, NY, USA, 2003. ACM.

[101] Alexandre V. Evfimievski. Randomization in privacy-preserving data mining. *SIGKDD Explorations*, 4(2):43–48, 2002.

[102] Alexandre V. Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. *Inf. Syst.*, 29(4):343–364, 2004.

[103] Joan Feigenbaum, Yuval Ishai, Tal Malkin, Kobbi Nissim, Martin J. Strauss, and Rebecca N. Wright. Secure Multiparty Computation of Approximations. *ACM Trans. Algorithms*, 2(3):435–472, July 2006.

[104] Stephen E. Fienberg and Julie McIntyre. Data swapping: Variations on a theme by dalenius and reiss. In Josep Domingo-Ferrer and Vicenç Torra, editors, *Privacy in Statistical Databases*, volume 3050 of *Lecture Notes in Computer Science*, pages 14–29. Springer, 2004.

[105] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[106] Ian Fox, Lynn Ang, Mamta Jaiswal, Rodica Pop-Busui, and Jenna Wiens. Contextual motifs: Increasing the utility of motifs using contextual data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 155–164, New York, NY, USA, 2017. ACM.

[107] Martin Franz. *Secure Computations on Non-Integer Values*. PhD thesis, TU Darmstadt, 2011.

[108] Martin Franz, Björn Deiseroth, Kay Hamacher, Somesh Jha, Stefan Katzenbeisser, and Heike Schröder. Secure computations on non-integer values. In *Information Forensics and Security (WIFS), 2010 IEEE International Workshop on*, pages 1–6. IEEE, 2010.

[109] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, San Diego, CA, 2014. USENIX Association.

[110] Michael J Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, volume 3378, pages 303–324. Springer, 2005.

[111] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. *Efficient Private Matching and Set Intersection*, pages 1–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[112] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37, 2014.

[113] Junhao Gan and Yufei Tao. Dynamic density based clustering. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1493–1507, New York, NY, USA, 2017. ACM.

[114] Jing Gao, Wei Fan, Jiawei Han, and Philip S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proc. of the 7th SIAM Int. Conf. on Data Mining*, SDM, 2007.

[115] Juan A. Garay, Berry Schoenmakers, and José Villegas. Practical and secure solutions for integer comparison. In *Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings*, pages 330–342, 2007.

[116] Adria Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. 4:248–267. bibtex: GSB+17.

[117] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[118] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. *On Private Scalar Product Computation for Privacy-Preserving Data Mining*, pages 104–120. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[119] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[120] Michael T. Goodrich. Randomized Shellsort: A Simple Oblivious Sorting Algorithm. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1262–1277, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[121] S. Goryczka and L. Xiong. A Comprehensive Comparison of Multiparty Secure Additions with Differential Privacy. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1–1, 2017.

[122] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML Confidential: Machine Learning on Encrypted Data. Cryptology ePrint Archive, Report 2012/323, 2012.

[123] Koki Hamada, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Oblivious radix sort: An efficient sorting algorithm for practical secure multi-party computation. *IACR Cryptology ePrint Archive*, 2014:121, 2014.

[124] Koki Hamada, Ryo Kikuchi, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Practically efficient multi-party sorting protocols from comparison sort algorithms. In *International Conference on Information Security and Cryptology*, pages 202–216. Springer, 2012.

[125] Shuguo Han, Wee Keong Ng, and Philip S. Yu. Privacy-preserving singular value decomposition.

[126] S. Haykin and Liang Li. Nonlinear adaptive prediction of nonstationary signals. *IEEE Trans. on Signal Processing*, 43(2):526–535, 1995.

[127] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Scalable multi-party private set-intersection. In *IACR International Workshop on Public Key Cryptography*, pages 175–203. Springer, 2017.

[128] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*, 2012.

[129] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, volume 201, 2011.

[130] Yan Huang, Chih-hao Shen, David Evans, Jonathan Katz, and Abhi Shelat. Efficient secure computation with garbled circuits. In *Information Systems Security*, Lecture Notes in Computer Science, pages 28–48. Springer, Berlin, Heidelberg.

[131] iDASH - integrating Data for Analysis, Anonimization, and SHaring, 2016. Webpage at https://idash.ucsd.edu/genomics. 2016 competition at http://www.humangenomeprivacy.org/2016/.

[132] IEEE Task P754. *ANSI/IEEE 754-1985, Standard for Binary Floating-Point Arithmetic*. August 1985. Also standardized as *IEC 60559 (1989-01) Binary floating-point arithmetic for microprocessor systems*.

[133] Piotr Indyk and David Woodruff. Tight lower bounds for the distinct elements problem. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 283–288. IEEE, 2003.

[134] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. Cryptology ePrint Archive, Report 2017/738, 2017.

[135] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N. Wright. A practical differentially private random decision tree classifier. *Trans. Data Privacy*, 5(1):273–295, 2012.

[136] Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 593–599, New York, NY, USA, 2005. ACM.

[137] Zhanglong Ji, Xiaoqian Jiang, Shuang Wang, Li Xiong, and Lucila Ohno-Machado. Differentially private distributed logistic regression using private and public data. *BMC Medical Genomics*, 7(1):S14, May 2014.

[138] Wei Jiang and Chris Clifton. *Privacy-Preserving Distributed k-Anonymity*, pages 166–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[139] Xin Jin and Jiawei Han. *Expectation Maximization Clustering*, pages 382–383. Springer US, Boston, MA, 2010.

[140] Kristján Valur Jónsson, Gunnar Kreitz, and Misbah Uddin. Secure Multi-Party Sorting and Applications. Cryptology ePrint Archive, Report 2011/122, 2011.

[141] Marc Joye. Privacy-Preserving Ridge Regression Without Garbled Circuits. Cryptology ePrint Archive, Report 2017/732, 2017.

[142] Seny Kamara, Payman Mohassel, Mariana Raykova, and Saeed Sadeghian. Scaling private set intersection to billion-element sets. In *International Conference on Financial Cryptography and Data Security*, pages 195–215. Springer, 2014.

[143] Liina Kamm and Jan Willemson. Secure floating point arithmetic and private satellite collision analysis. *International Journal of Information Security*, 14(6):531–548, 2015.

[144] Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 41–52. ACM, 2010.

[145] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, pages 99–106. IEEE Computer Society, 2003.

[146] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure ot extension with optimal overhead. In *Annual Cryptology Conference*, pages 724–741. Springer, 2015.

[147] Marcel Keller and Peter Scholl. Efficient, oblivious data structures for MPC. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 506–525. Springer, 2014.

[148] Mark G. Kelly, David J. Hand, and Niall M. Adams. The impact of changing populations on classifier performance. In *Proc. of the 5th ACM SIGKDD int. conf. on Knowl. disc. and dat. min.*, KDD, pages 367–371, 1999.

[149] Krishnaram Kenthapadi, Nina Mishra, and Kobbi Nissim. Simulatable auditing. In Chen Li, editor, *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*, pages 118–127. ACM, 2005.

[150] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 193–204, New York, NY, USA, 2011. ACM.

[151] Eike Kiltz, Gregor Leander, and John Malone-Lee. Secure Computation of the Mean and Related Statistics. Cryptology ePrint Archive, Report 2004/359, 2004.

[152] Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew Franklin. Secure linear algebra using linearly recurrent sequences. In *Theory of Cryptography Conference*, pages 291–310. Springer. bibtex: KMWF07.

[153] Lea Kissner, Dawn Song, et al. Privacy-preserving set operations. In *Crypto*, volume 3621, pages 241–257. Springer, 2005.

[154] Matthias Klusch, Stefano Lodi, and Gianluca Moro. Distributed clustering based on sampling local density estimates. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 485–490. Morgan Kaufmann, 2003.

[155] Donald E. Knuth. *The Art of Computer Programming: Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998.

[156] Yun Sing Koh and Sri Devi Ravana. Unsupervised rare pattern mining: A survey. *ACM Trans. Knowl. Discov. Data*, 10(4):45:1–45:29, May 2016.

[157] Vladimir Kolesnikov and Ranjit Kumaresan. Improved ot extension for transferring short secrets. In *Advances in Cryptology–CRYPTO 2013*, pages 54–70. Springer, 2013.

[158] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829. ACM, 2016.

[159] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, pages 1–20, 2009.

[160] Toomas Krips and Jan Willemson. Hybrid model of fixed and floating point numbers in secure multiparty computations. In *International Conference on Information Security*, pages 179–197. Springer, 2014.

[161] Mitsuru Kusumoto, Takanori Maehara, and Ken-ichi Kawarabayashi. Scalable similarity search for simrank. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 325–336, New York, NY, USA, 2014. ACM.

[162] Mikkel Lambæk. Breaking and fixing private set intersection protocols. *IACR Cryptology ePrint Archive*, 2016:665, 2016.

[163] S. Lang. *Algebra*. Graduate Texts in Mathematics. Springer New York, 2005.

[164] Peeter Laud and Alisa Pankova. On the (im)possibility of privately outsourcing linear programming. In Ari Juels and Bryan Parno, editors, *CCSW'13, Proceedings of the 2013 ACM Cloud Computing Security Workshop, Co-located with CCS 2013, Berlin, Germany, November 4, 2013*, pages 55–64. ACM, 2013.

[165] Sven Laur, Riivo Talviste, and Jan Willemson. From oblivious {AES} to efficient and secure database join in the multiparty setting. In *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, pages 84–101, 2013.

[166] Sven Laur, Jan Willemson, and Bingsheng Zhang. Round-efficient oblivious database manipulation. In *ISC*, volume 11, pages 262–277. Springer, 2011.

[167] Jaewoo Lee and Chris Clifton. How much is enough? choosing ε for differential privacy. In Xuejia Lai, Jianying Zhou, and Hui Li, editors, *Information Security, 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings*, volume 7001 of *Lecture Notes in Computer Science*, pages 325–340. Springer, 2011.

[168] J. Leskovec, A. Rajaraman, and J.D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

[169] Xiaodong Lin, Chris Clifton, and Michael Zhu. Privacy-preserving clustering with distributed EM mixture modeling. *Knowledge and Information Systems*, 8(1):68–81, 2005.

[170] Yehuda Lindell and Benny Pinkas. Privacy Preserving Data Mining. *J. Cryptology*, 15(3):177–206, 2002.

[171] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(5), 2009.

[172] Indre Žliobaite, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. In *Big Data Analysis: New Algorithms for a New Society*, pages 91–114. Springer, 2016.

[173] Helger Lipmaa and Tomas Toft. Secure equality and greater-than tests with sublinear online complexity. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 645–656, 2013.

[174] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious Neural Network Predictions via MiniONN transformations. Cryptology ePrint Archive, Report 2017/452, 2017. `http://eprint.iacr.org/2017/452`.

[175] Yun-Ching Liu, Yi-Ting Chiang, Tsan-Sheng Hsu, Churn-Jung Liau, and Da-Wei Wang. Floating point arithmetic protocols for constructing secure data analysis application. *Procedia Computer Science*, 22:152–161, 2013.

[176] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, volume 4. San Diego, CA, USA, 2004.

[177] M. Maloof and R. Michalski. Incremental learning with partial instance memory. *Artificial Intelligence*, 154:95–126, 2004.

[178] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient Algorithms for Discovering Association Rules. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, pages 181–192. AAAI Press, 1994.

[179] Moxie Marlinspike. The difficulty of private contact discovery. *Blog post https://whispersystems. org/blog/contact-discovery*, 2014.

[180] David J. Martin, Daniel Kifer, Ashwin Machanavajjhala, Johannes Gehrke, and Joseph Y. Halpern. Worst-case background knowledge for privacy-preserving data publishing. In Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis, editors, *Proceedings of the 23rd*

International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007, pages 126–135. IEEE Computer Society, 2007.

[181] Frank McSherry and Ilya Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki, editors, *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 627–636. ACM, 2009.

[182] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *Security and Privacy, 1986 IEEE Symposium on*, pages 134–134. IEEE, 1986.

[183] Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient private statistics with succinct sketches. *arXiv preprint arXiv:1508.06110*, 2015.

[184] Xianrui Meng, Seny Kamara, Kobbi Nissim, and George Kollios. Grecs: graph encryption for approximate shortest distance queries. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 504–517. ACM, 2015.

[185] Srujana Merugu and Joydeep Ghosh. Privacy-preserving distributed clustering using generative models. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, pages 211–218. IEEE Computer Society, 2003.

[186] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In Catriel Beeri and Alin Deutsch, editors, *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France*, pages 223–228. ACM, 2004.

[187] Gerome Miklau and Dan Suciu. A formal analysis of information disclosure in data exchange. *J. Comput. Syst. Sci.*, 73(3):507–534, 2007.

[188] L. L. Minku, A. P. White, and X. Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Trans. Knowl. Data Eng.*, 22(5):730–742, 2010.

[189] Nina Mishra and Mark Sandler. Privacy via pseudorandom sketches. In Stijn Vansummeren, editor, *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 143–152. ACM, 2006.

[190] Payman Mohassel and Enav Weinreb. Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries. In *Advances in Cryptology - CRYPTO 2008*, Lecture Notes in Computer Science, pages 481–496. Springer, Berlin, Heidelberg.

[191] Richard A. Moore. Analysis of the kim-winkler algorithm for masking microdata files – how much masking is necessary and sufficient? conjectures for the development of a controllable algorithm. https://www.census.gov/srd/CDAR/rr96-05_Analysis_of_Kim-Winkler.pdf, 1996.

[192] J. G. Moreno-Torres, T. Raeder, R. Alaiz-RodríGuez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. *Pattern Recogn.*, 45(1):521–530, 2012.

[193] Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, October 1978.

[194] Stephen Muggleton and Hiroaki Watanabe. *Latest Advances in Inductive Logic Programming*. Imperial College Press, London, UK, UK, 2014.

[195] J. Ian Munro and Mike Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980.

[196] Shubha U. Nabar, Bhaskara Marthi, Krishnaram Kenthapadi, Nina Mishra, and Rajeev Motwani. Towards robustness in query auditing. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, pages 151–162. ACM, 2006.

[197] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, and Nikita Borisov. Botgrep: Finding p2p bots with structured graph analysis. In *USENIX Security Symposium*, pages 95–110, 2010.

[198] Arjun Narayan. *Distributed differential privacy and applications*. University of Pennsylvania, 2015.

[199] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*, pages 111–125. IEEE Computer Society, 2008.

[200] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, volume 7417, pages 681–700. Springer, 2012.

[201] Valeria Nikolaenko, Stratis Ioannidis, Udi Weinsberg, Marc Joye, Nina Taft, and Dan Boneh. Privacy-preserving Matrix Factorization. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 801–812, New York, NY, USA, 2013. ACM.

[202] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-Preserving Ridge Regression on Hundreds of Millions of Records. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 334–348, Washington, DC, USA, 2013. IEEE Computer Society.

[203] Kobbi Nissim and Enav Weinreb. Communication efficient secure linear algebra. In *Theory of Cryptography Conference*, pages 522–541. Springer. bibtex: NW06.

[204] Petra Kralj Novak, Nada Lavrač, and Geoffrey I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *J. Mach. Learn. Res.*, 10:377–403, June 2009.

[205] G O'connor Daniel et al. Sorting system with nu-line sorting switch, April 10 1962. US Patent 3,029,413.

[206] Burcu Demirelli Okkalioglu, Murat Okkalioglu, Mehmet Koç, and Huseyin Polat. A survey: deriving private information from perturbed data. *Artif. Intell. Rev.*, 44(4):547–569, 2015.

[207] Stanley R. M. Oliveira and Osmar R. Zaïane. Privacy preserving clustering by data transformation. In *XVIII Simpósio Brasileiro de Bancos de Dados, 6-8 de Outubro, Manaus, Amazonas, Brasil, Anais/Proceedings.*, pages 304–318, 2003.

[208] Stanley R. M. Oliveira, Osmar R. Zaïane, and Yücel Saygin. Secure association rule sharing. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *Advances in Knowledge Discovery and Data Mining, 8th Pacific-Asia Conference, PAKDD 2004, Sydney, Australia, May 26-28, 2004, Proceedings*, volume 3056 of *Lecture Notes in Computer Science*, pages 74–85. Springer, 2004.

[209] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-n ot extension with application to private set intersection. In *Cryptographers' Track at the RSA Conference*, pages 381–396. Springer, 2017.

[210] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, and Boaz Moskovich. Scifi-a system for secure face identification. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 239–254. IEEE, 2010.

[211] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 165–176, Oct 2006.

[212] Pascal Paillier. *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, pages 223–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[213] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[214] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private Set Intersection Using Permutation-based Hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, Washington, D.C., 2015. USENIX Association.

[215] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on ot extension. In *USENIX Security Symposium*, pages 797–812, 2014.

[216] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *IACR Cryptology ePrint Archive*, 2016:930, 2016.

[217] H. Polat and Wenliang Du. Privacy-preserving top-n recommendation on horizontally partitioned data. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 725–731, Sept 2005.

[218] Pille Pullonen and Sander Siim. Combining secret sharing and garbled circuits for efficient private ieee 754 floating-point computations. In *International Conference on Financial Cryptography and Data Security*, pages 172–183. Springer, 2015.

[219] J. R. Quinlan. Improved use of continuous attributes in c4.5. *J. Artif. Int. Res.*, 4(1):77–90, March 1996.

[220] MO Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

[221] Thomas Reinartz. *Focusing Solutions for Data Mining: Analytical Studies and Experimental Results in Real-world Domains*. Springer-Verlag, Berlin, Heidelberg, 1999.

[222] Steven P Reiss. Security in databases: A combinatorial study. *Journal of the ACM (JACM)*, 26(1):45–57, 1979.

[223] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 235–259. Springer, 2017.

[224] Shariq Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 682–693. Morgan Kaufmann, 2002.

[225] Tim Roughgarden. Lecture notes cs369e: Communication complexity (for algorithm designers), 2015. Stanford University.

[226] Bita Darvish Rouhani, M. Sadegh Riazi, and Farinaz Koushanfar. DeepSecure: Scalable Provably-Secure Deep Learning. Cryptology ePrint Archive, Report 2017/502, 2017. http://eprint.iacr.org/2017/502.

[227] Marcos Salganicoff. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, 11(1-5):133–155, 1997.

[228] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, Technical report, SRI International, 1998.

[229] Rathindra Sarathy and Krishnamurty Muralidhar. Evaluating laplace noise addition to satisfy differential privacy for numeric data. *Trans. Data Privacy*, 4(1):1–17, 2011.

[230] Yücel Saygin, Vassilios S. Verykios, and Chris Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, 2001.

[231] Adi Shamir. On the power of commutativity in cryptography. *Automata, Languages and Programming*, pages 582–595, 1980.

[232] Donald L. Shell. A high-speed sorting procedure. *Communications of the ACM*, 2(7):30–32, 1959.

[233] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pages 26–37, New York, NY, USA, 2016. ACM.

[234] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *In VLDB' 11*, 2011.

[235] A. Tsymbal. The problem of concept drift: Definitions and related work. Technical report, Department of Computer Science, Trinity College Dublin, Ireland, 2004.

[236] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, 9(1):56–68, 2008.

[237] Jaideep Vaidya. Privacy-preserving linear programming. In Sung Y. Shin and Sascha Ossowski, editors, *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC), Honolulu, Hawaii, USA, March 9-12, 2009*, pages 2002–2007. ACM, 2009.

[238] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving svm classification. *Knowledge and Information Systems*, 14(2):161–178, Feb 2008.

[239] Adriano Veloso, Wagner Meira Jr., Srinivasan Parthasarathy, and Márcio de Carvalho. Efficient, accurate and privacy-preserving data mining for frequent itemsets in distributed databases. In *XVIII Simpósio Brasileiro de Bancos de Dados, 6-8 de Outubro, Manaus, Amazonas, Brasil, Anais/Proceedings.*, pages 281–292, 2003.

[240] Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yücel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[241] Robert A. Wagner and Michael J. Fischer. The String-to-String Correction Problem. *J. ACM*, 21(1):168–173, January 1974.

[242] Abraham Waksman. A permutation network. *J. ACM*, 15(1):159–163, 1968.

[243] Guan Wang, Tongbo Luo, Michael T. Goodrich, Wenliang Du, and Zutao Zhu. Bureaucratic Protocols for Secure Two-party Sorting, Selection, and Permuting. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 226–237, New York, NY, USA, 2010. ACM.

[244] Ke Wang, Benjamin C. M. Fung, and Guozhu Dong. Integrating private databases for data analysis. In Paul B. Kantor, Gheorghe Muresan, Fred S. Roberts, Daniel Dajun Zeng, Fei-Yue Wang, Hsinchun Chen, and Ralph C. Merkle, editors, *Intelligence and Security Informatics, IEEE International Conference on Intelligence and Security Informatics, ISI 2005, Atlanta, GA, USA, May 19-20, 2005, Proceedings*, volume 3495 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2005.

[245] Xiao Shaun Wang, Yan Huang, Yongan Zhao, Haixu Tang, XiaoFeng Wang, and Diyue Bu. Efficient Genome-Wide, Privacy-Preserving Similar Patient Query Based on Private Edit Distance. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 492–503, New York, NY, USA, 2015. ACM.

[246] Gerhard Widmer and Miroslav Kubat. Effective learning in dynamic environments by explicit context tracking. In *Proc. of the Eur. Conf. on Mach. Learn.*, ECML, pages 227–243, 1993.

[247] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Mach. Learn.*, 23(1):69–101, 1996.

[248] Hee-Sun Won, Sang-Pil Kim, Sanghun Lee, Mi-Jung Choi, and Yang-Sae Moon. Secure principal component analysis in multiple distributed nodes. *Security and Communication Networks*, 9(14):2348–2358, 2016. sec.1501.

[249] David Woodruff. Optimal space lower bounds for all frequency moments. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 167–175, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[250] David J. Wu, Tony Feng, Michael Naehrig, and Kristin Lauter. Privately evaluating decision trees and random forests. Cryptology ePrint Archive, Report 2015/386, 2015. http://eprint.iacr.org/2015/386.

[251] A. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pages 160–164. IEEE Computer Society, 1982.

[252] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-preserving SVM classification on vertically partitioned data. In Wee Keong Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang, editors, *Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conference, PAKDD 2006, Singapore, April 9-12, 2006, Proceedings*, volume 3918 of *Lecture Notes in Computer Science*, pages 647–656. Springer, 2006.

[253] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, May 2000.

[254] Bingsheng Zhang. *Generic Constant-Round Oblivious Sorting Algorithm for MPC*, pages 240–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[255] Ning Zhang, Ming Li, and Wenjing Lou. Distributed data mining with differential privacy. In *Proceedings of IEEE International Conference on Communications, ICC 2011, Kyoto, Japan, 5-9 June, 2011*, pages 1–5. IEEE, 2011.

[256] Yihua Zhang and Marina Blanton. Efficient secure and verifiable outsourcing of matrix multiplications. In *Information Security*, Lecture Notes in Computer Science, pages 158–178. Springer, Cham.

[257] Yihua Zhang, Aaron Steele, and Marina Blanton. PICCO: a general-purpose compiler for private distributed computation. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 813–826, 2013.

[258] Kaidi Zhao, Bing Liu, Jeffrey Benkler, and Weimin Xiao. Opportunity map: Identifying causes of failure - a deployed data mining system. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 892–901, New York, NY, USA, 2006. ACM.

[259] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. Privacy-enhancing k-anonymization of customer data. In Chen Li, editor, *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*, pages 139–147. ACM, 2005.