2021

Confidential Machine Learning on Untrusted Platforms: a Survey

Sharma Sagar

Keke Chen

Sagar and Keke *Cybersecurity* (2021) 4:30
https://doi.org/10.1186/s42400-021-00092-8

Cybersecurity

SURVEY Open Access

Confidential machine learning on untrusted platforms: a survey

Sharma Sagar[1,2†] and Chen Keke[1*†]

Abstract

With the ever-growing data and the need for developing powerful machine learning models, data owners increasingly depend on various untrusted platforms (e.g., public clouds, edges, and machine learning service providers) for scalable processing or collaborative learning. Thus, sensitive data and models are in danger of unauthorized access, misuse, and privacy compromises. A relatively new body of research confidentially trains machine learning models on protected data to address these concerns. In this survey, we summarize notable studies in this emerging area of research. With a unified framework, we highlight the critical challenges and innovations in outsourcing machine learning confidentially. We focus on the cryptographic approaches for confidential machine learning (CML), primarily on model training, while also covering other directions such as perturbation-based approaches and CML in the hardware-assisted computing environment. The discussion will take a holistic way to consider a rich context of the related threat models, security assumptions, design principles, and associated trade-offs amongst data utility, cost, and confidentiality.

Keywords: Confidential computing, Cryptographic protocols, Machine learning

Introduction

Data-driven methods, e.g., machine learning and data mining, have become essential tools for numerous research and application domains. With abundant data, data owners can build complex analytic models for areas ranging from social networking, healthcare informatics, entertainment, and advanced science and technology. However, limited in-house resources, inadequate expertise, or collaborative/distributed processing needs force data owners (e.g., parties that collect and analyze user-generated data) to depend on somewhat untrusted platforms (e.g., cloud/edge service providers) for elastic storage and data processing. As a result, cloud services for data analytics, such as machine-learning-as-a-service (MLaaS), have been rapidly growing during the past few years. While untrusted platforms refer to all non-in-house resources not directly owned by the data owner, we will use Cloud Services to represent them here forth.

When outsourcing sensitive data (e.g., proprietary, human-related, or confidential data), data owners have raised concerns in privacy, confidentiality, and ownership (Sharma et al. 2018; Duncan et al. 2012). On the one hand, cloud users cannot verifiably prevent the cloud provider from accessing their data; i.e., in practice, using public clouds often means one must fully trust the cloud provider. On the other hand, public cloud providers are not immune to security attacks leading to sensitive data breaches. Recent security incidents, including insider attacks (Chen 2010; Duncan et al. 2012) and external security breaches at the service providers (Mansfield-Devine 2015; Unger 2015), show the risks are aggravating by day. Researchers and practitioners have developed solutions to protect the confidentiality of cloud data at rest. For example, Google Cloud Platform has allowed users to include an external key manager to store encrypted data on the cloud with a third party (e.g., Fortanix) stores and manages keys off the cloud. However, it remains a critical challenge for data owners and cloud providers to protect confidentiality in computing, i.e., training models on

*Correspondence: keke.chen@marquette.edu
†Sharma Sagar and Chen Keke contributed equally to this work.
[1]Northwestern Mutual Data Science Associate Professor Director of Trustworthy and Intelligent Computing Lab Department of Computer Science Marquette University Milwaukee, Wisconsin, USA
Full list of author information is available at the end of the article

the cloud, while protecting the confidentiality of both the training data and the learned models.

In the past few years, researchers have made some progress in developing novel confidential machine learning (CML) approaches for model training with encrypted data. A successful CML approach is not straightforward. Unlike traditional machine learning approaches, a practical CML framework wrestles in balancing security (confidentiality) guarantees, costs, and model quality, while allocating appropriate workload distributions between cloud and client. Direct application of cryptographic and privacy-protection methods such as fully homomorphic encryption (FHE) (Gentry 2009) and garbled circuits (GC) (Yao 1986) in a homogeneous fashion do not usually meet the criteria for practical CML approaches. Most efficient approaches have been using hybrid methods that combine multiple primitives instead of a homogeneous composition. Recent studies (Nikolaenko et al. 2013; Nikolaenko et al. 2013; Demmler et al. 2015; Mohassel and Zhang 2017; Sharma et al. 2019; Sharma and Chen 2019) have followed the hybrid direction to effectively reduce performance bottlenecks and other practicality issues in developing CML solutions. However, the underlying techniques in these studies scatter among several papers making the basic principles unclear. The purpose of this survey is to uncover these basic principles and accurately organize the existing techniques under a unified framework so that researchers and practitioners can quickly grasp the development and challenges in this new area of research.

Contributions and Organization Overview. Capturing a comprehensive view of a complex and new topic like confidential machine learning is challenging. We primarily focus on frameworks for *model training* using cryptographic techniques that guarantee strong (semantic) security with practical cost overburden. A complete machine learning service usually includes a model application (or *model inference*) component that applies the trained model to generate a prediction for new input data, equivalent to secure function evaluation. The confidential model inference is much simpler and in a more mature state than confidential model training, therefore, not covered in this survey. Interested readers may refer to the related studies about confidential inference with pretrained models, such as Gilad-Bachrach et al. (2016), Bost et al. (2015), Hesamifard et al. (2017), and Rouhani et al. (2018).

This survey paper presents a unified perspective on designing and implementing different CML model learning methods with state-of-the-art cryptographic approaches. Despite numerous machine learning methods (Hastie et al. 2001), the studies on CML methods have focused on only a few specific machine learning methods. On the other hand, researchers have applied several cryptographic methods to realize CML frameworks. We observe that many clever CML techniques apply to specific machine learning algorithms without clear guidance or framework on extending these basic principles to broader machine learning algorithms. To systematically understand the set of developed techniques in CML, we summarize them under a general framework, the decomposition-mapping-composition (DMC) procedure + design and selection of crypto-friendly algorithms. The DMC procedure involves: decomposing the target machine learning algorithm into several components, mapping these components to their cryptographic constructions, and finally composing the CML solution with the confidential component counterparts. Moreover, several CML approaches adopting the DMC process development exhibit a unique additional feature: they use "crypto-friendly" alternative machine learning algorithms or components to achieve more efficient protocols. Keeping these observations in mind, we develop a systemization framework to summarize the design principles, strategies, cryptographic techniques, and optimization measures, which have been applied to solve the challenging problems in confidential machine learning over protected data.

We organize the survey based on underlying design principles of CML rather than any specific machine learning problems. As part of the survey, we summarize the experiences and learnings in each category of CML topics as *insights* and *gaps*. This work promotes practical aspects of applying cryptographic primitives in CML at their current level of maturity. Focuses will be on how different frameworks balance the associated trade-offs amongst cost, confidentiality, and data utility or model quality in different threat models and privacy settings. The survey, however, does not cover the orthogonal line of research that aims to optimize fully expressive primitives such as FHE and GC schemes. This survey will be a great resource for researchers to adopt and advance privacy-enhancing technologies in solving novel research questions and for practitioners to learn the best practices and avoid common pitfalls.

In the following sections, first, we will include the necessary background knowledge, notations, definitions, and the targeted threat model in Preliminaries of CML approaches section. Then, in Systematization framework section, we present the systematization framework along with the basic principles and methodologies in the CML development. After that, we briefly discuss the homogeneous approaches that aim to translate any machine learning algorithm into a confidential one with a single cryptographic primitive (Homogeneous cryptographic approaches section). Next, we move to the main theme of this survey: the compositional hybrid approaches (Hybrid composition section), which have resulted in more efficient protocols for complex

machine learning algorithms. We will also cover several topics, such as security proofs and common evaluation methods for cryptographic protocols in Security proofs, attacks, and correctness and Evaluation methods sections. Finally, we briefly review other non-cryptographic-protocol approaches, including the perturbation methods and hardware-assisted (e.g., SGX) methods in Other CML approaches section.

Related work

A few survey papers are related to the topic of this paper. Shan et al. (2018) focus on techniques for practical secure outsourced computation, using machine learning as a sample application. However, it does not comprehensively cover the major approaches as we do. Attacks on the integrity of machine learning models have also raised serious concerns due to the wide applications of machine learning in real-life scenarios such as self-driving cars (Grigorescu et al. 2019). Different from our survey focusing on the confidentiality of the model learning process, Papernot et al. (2018) focus on the integrity of training data, learning process, models, and model application.

There are also several survey papers on a specific category of cryptographic primitives. Since the first fully homomorphic encryption scheme was published in 2009 (Gentry 2009), it has been an active research area during the past decade. Acar et al. (2018) have a comprehensive review about the current development of homomorphic encryption schemes. Secure multi-party computation methods, including the garbled circuits and secret sharing methods, have been actively developed for the past two decades. Readers may find more information from other sources (Lindell 2020; Evans et al. 2018).

Differentially private machine learning frameworks are somewhat related to CML but hold a distinct thread model that aims to share data and models. They assume that the data consumer (i.e., model developer or model users) is not trusted and may try to, who may try to reveal private information in the training data shared by data owners or data contributors. Differential Privacy does not protect the ownership of data and models as the purpose is to share them without breaching individuals' privacy in the training data. Along with recent developments on differentially private deep learning such as Abadi et al. (2016) and Shokri and Shmatikov (2015), Ji et al. (2014) and Sarwate and Chaudhuri (2013) also provide excellent surveys on this topic. Other studies in privacy-preserving data mining (PPDM) (Aggarwal and Yu 2008; Matwin 2013; Aldeen et al. 2015; Sachan et al. 2013) also aim to share the data (and the models) while preserving individual's privacy, thus excluded from our survey.

Preliminaries of CML approaches

In this section, we review the terms and concepts used in the literature. First, we look at the representative system architectures considered in the published confidential machine learning (CML) approaches based on cryptographic protocols. Then, we examine how different threat models, associated confidential assets, and considered attacks affect CML designs. Finally, we briefly describe prevailing cryptographic and privacy primitives that serve as the skeleton of most CML approaches.

System architectures

The CML research is motivated by the cloud computing paradigm and then extended to additional scenarios, such as edge computing and services computing. Thus, we use "Cloud" as the representative of untrusted platforms in CML system architectures henceforth. Such a system may involve cloud providers, optional cryptographic service providers, data owners or application service providers, and data and model consumers. Figure 1 shows an architecture with a data owner outsourcing its data and computation to a single cloud provider. The data owner must ensure the cloud provider does not compromise any proprietary and privacy-sensitive data. A few homomorphic-encryption-based frameworks, e.g., Graepel et al. (2012) and Lu et al. (2016), present protocols for training machine learning models over encrypted data outsourced to a cloud provider without almost any engagement of the data owner. However, the associated cost makes these protocols unrealistic in real-life scenarios. An alternate strategy would involve the data owner in minimal tasks intermediately to simplify the single cloud architecture framework (Sharma et al. 2019). As long as the cloud takes the majority of the workload and the client's cost is practical, e.g., linear or sublinear to

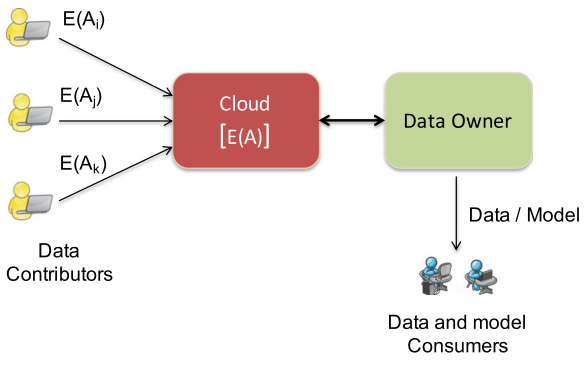

Fig. 1 A data owner outsourcing to an untrusted cloud provider for learning a model. The data contributors directly submit their encrypted data to the cloud. The cloud carries out the major expensive computations over the encrypted data and data owner can assist with some lightweight work

the number of records, more efficient protocols can be possible.

As some protocols become too expensive for the data owner to assist in cloud-centric learning, the architecture was evolved to a multi-server(cloud) setting. A data owner may choose to rely on two or more cloud providers to reduce the overall expense of learning. The second party may be as equally capable as the first party (Mohassel and Zhang 2017), or in the case of a cryptographic service provider (CSP), which manages keys and assists the cloud with intermediate decryption operations and light-weight computations (Nikolaenko et al. 2013; Nikolaenko et al. 2013; Sharma and Chen 2019). The two untrusted parties in such an architecture carry out secure multi-party computations without any of the parties learning the training data or the trained model. This setting also assumes that the two parties do not collude with each other, thus is slightly more vulnerable than the client-cloud two-party setting. Figure 2 shows such a framework that uses a garbled circuit.

Threat models

In this section, we examine the widely accepted threat models in the context of CML. We focus on the following aspects: the assumptions on the adversaries and the related confidential assets in CML.

Assumptions on Adversaries. Most CML approaches (Sharma and Chen 2019; Mohassel and Zhang 2017; Nikolaenko et al. 2013; Nikolaenko et al. 2013; Graepel et al. 2012) adopt the honest-but-curious (or semi-honest) adversary model to describe the untrusted cloud provider. Honest-but-curious parties, by definition, perform their share of tasks obediently, i.e., guarantee data and model integrity and follow the pre-defined protocols exactly. However, they might clandestinely snoop the storage, interactions, and computations to learn private information. Data owners and data contributors' concerns about data and model leakages, even when the infrastructure platforms are reputed, are alleviated by preserving the confidentiality of data and models. Many CML approaches also use an honest-but-curious cryptographic service provider to design more efficient protocols.

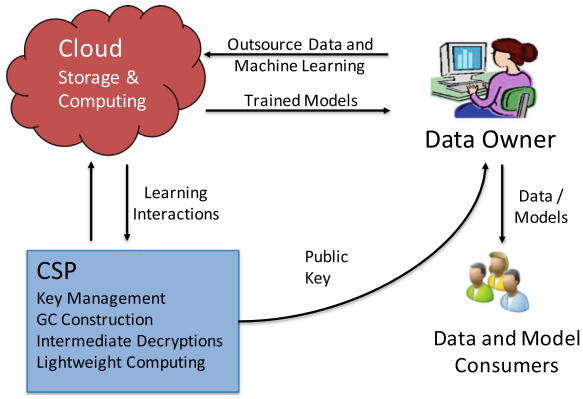

Fig. 2 A data owner outsources data storage and machine learning tasks to the Cloud. The Cryptographic Service Provider (CSP) manages the keys, decrypts intermediate results, and assists the Cloud with other relatively lightweight computations

Some CML approaches additionally address an adversary which actively seeks to compromise data and model confidentiality by performing additional probing tasks, e.g., by inserting crafted records or secretively running the algorithms on a selected record set offline. Sharma and Chen (2019) address the possibility of an adversary who may actively track identifiable training records to the datasets and follow the computations to infer the information about other training records. Nikolaenko et al. (2013) consider an adversary that selectively runs the machine learning protocol over an individual's data to draw personal inferences from the learned models.

Nevertheless, with either passive or active adversaries, CML approaches assume that the data and model integrity are not compromised at the end of the training. This assumption distinguishes CML from other studies such as attacks on machine learning by polluting training data or modifying learned models (Liu et al. 2018).

Moreover, the CML approaches often assume non-collusion between the involved parties, for example, between the cloud provider and the CSP (Nikolaenko et al. 2013; Nikolaenko et al. 2013; Sharma and Chen 2019) or the two cloud providers (Mohassel and Zhang 2017) in the two-server architecture. Collusion between the two parties in these frameworks directly compromises the privacy of the training data and learned models.

Most CML approaches assume that data and model consumers are trusted, which is orthogonal to the applications of differential privacy (Shokri and Shmatikov 2015; Abadi et al. 2016) that specifically targets untrusted data and model consumers. Furthermore, CML approaches assume properly secured infrastructures and communication channels to exclude external attacks and focus on the CML-specific challenges.

Confidential Assets at Risk. An adversarial party may be interested in the confidentiality of *sensitive data* and the *generated models*. All CML methods protect the training data feature vectors. Some methods designed for supervised learning (Graepel et al. 2012; Nikolaenko et al. 2013) expose the training data labels to simplify their secure modeling algorithms with the assumption that knowing the labels will not leak significantly more information to adversaries, which might be false for some applications. Some CML studies also expose unprotected models (Graepel et al. 2012; Nikolaenko et al. 2013; Lu et al. 2016). However, recent studies (Fredrikson et al. 2014; Fredrikson et al. 2015; Hitaj et al. 2017; Shokri et al. 2017;

Song and Shmatikov 2019) have shown that an adversary may use crafted data derived from trained models to infer sensitive training data or use the advanced features in deep learning models to breach data privacy. Furthermore, the intermediate results of outsourcing computations in the setting of federated learning, for example, the intermediate representation in a convolutional neural network learning, may reveal information about the private training data (Shokri and Shmatikov 2015). Thus, CML methods must protect both data and model confidentiality.

Cryptographic primitives

The cryptographic primitives are the fundamental building blocks for CML approaches. Some of these primitives are more expressive – meaning they can implement more types of functions or higher-level functions. On the other hand, some primitives are more cost-efficient than others. To make this survey self-contained, in this section, we briefly cover the most frequently-used primitives in existing CML approaches.

Additive Homomorphic Encryption (AHE). AHE schemes (e.g., Paillier encryption (Paillier 1999)) allow the additive operation over encrypted messages without decryption. For any two integers α and β, an AHE scheme allows the additive homomorphic operation: $E(\alpha + \beta) = f(E(\alpha), E(\beta))$ where the function f works on encrypted values. Conceptually, with one of the operands unencrypted, a "pseudo-homomorphic" multiplication between two messages can be expressed as a series of additions[1], i.e.,$E(\alpha\beta) = E(\sum_{i=1}^{\beta} \alpha)$. With homomorphic addition and pseudo-homomorphic multiplication, one can derive pseudo-homomorphic dot-product of vectors, matrix-vector multiplication, and matrix-matrix multiplication. However, the unencrypted operands in these operations either need to be non-sensitive information or protected with some masking and de-masking mechanism (Sharma et al. 2019; Sharma and Chen 2019). ElGamal, Goldwasser-Micali, Benaloh, and Okamoto-Uchiyama cryptosystems are some additional examples of AHE schemes (Acar et al. 2018).

Somewhat Homomorphic Encryption (SHE). There are many encryption schemes in this category (e.g., BV, BGV, NTRU, GSW, BFV, and BGN (Acar et al. 2018) and their variations such as TFHE (Chillotti et al. 2020) and CKK (Cheon et al. 2017)). SHE schemes allow both homomorphic additions and multiplications over encrypted messages, while the number of consecutive multiplications is limited to a few. A popular SHE scheme used in CML is the ring learning-with-error (RLWE) scheme that relies on the intractability of the learning-with-errors (LWE) problem on polynomial rings (Brakerski et al. 2014). Theoretically, RLWE supports arbitrary levels of multiplications. Therefore, it is considered to be fully homomorphic. However, due to the associated high cost for deeper levels of multiplications, RLWE is more suitable as a SHE scheme only (i.e., 1-3 levels of multiplications). A ciphertext in RLWE is represented as a two-tuple (c_0, c_1), where c_0 and c_1 are polynomials. Let $C_i = (c_{0,i}, c_{1,i})$ and C_j be the ciphertext of any two values. The encrypted addition of the two values is simply $(c_{0,i} + c_{0,j}, c_{1,i} + c_{1,j})$. The encrypted multiplication is translated to a series of polynomial operations on the ciphertext elements. RLWE allows multiple levels of multiplication at a certain cost. For details, please refer to the paper (Brakerski et al. 2014). *Message packing* (Brakerski et al. 2014) enables packing multiple ciphertexts into one polynomial, which considerably reduces RLWE's ciphertext size and optimizes linear algebra operations (Garay and Gennaro 2014). HELib library (Garay and Gennaro 2014) is a popular implementation of the RLWE scheme.

Garbled Circuits (GC). Garbled Circuits (GC) (Yao 1986) allow two parties, each holding an input to a function, to securely evaluate a function without revealing any information about the respective inputs. GC can express arbitrary functions using several basic gates such as AND and XOR gates in a secure two-party computation setting (usually with a Cryptographic Service Provider (CSP)). One party constructs the circuit, whereas the other evaluates it. Despite several GC cost optimization techniques, such as Free XOR gates (Kolesnikov and Schneider 2008), Half AND gates (Zahur et al. 2015), and OTExtensions (Asharov et al. 2013), GC still incurs high communication costs. Therefore, one must carefully examine its use in composing CML frameworks. FastGC (Huang et al. 2011) and ObliVM (Liu et al. 2015) are two popular GC libraries.

Randomized Secret Sharing (SecSh). The randomized secret sharing method (Demmler et al. 2015) protects data by splitting it into two (or multiple) random additive shares outsourced to two (or more) non-colluding untrusted parties. The two parties compute on the respective shares and return the results also as random shares. Addition is straightforward as $\alpha+\beta = (\alpha_0+\beta_0)+(\alpha_1+\beta_1)$ with α and β distributed between two parties 0 and 1. Multiplication, however, is expensive as it depends on the beaver triplet generation method (Demmler et al. 2015; Mohassel and Zhang 2017), which further depends on expensive AHE or Oblivious Transfer (OT) schemes to exchange the intermediate results securely.

Random Additive Masking. A data owner may generate a random mask to hide the sensitive data, which will be stripped off at a certain step in the CML protocol to recover the desired result. Due to its low cost, it frequently serves as an auxiliary tool for a complex protocol, for instance, in CML for spectral clustering (Sharma

[1] Some methods like Paillier encryption (Paillier 1999) allow more efficient pseudo-homomorphic multiplication.

et al. 2019), boosting (Sharma and Chen 2019), and matrix factorization (Nikolaenko et al. 2013).

Systematization framework

It is challenging to have a clear understanding of the whole body of CML model training methods due to the following reasons. First, the number of machine learning models is huge (Hastie et al. 2001) and even the most used ones are around tens (Wu et al. 2007). They are so different that no unified framework can be used to describe them. Second, security researchers are often more interested in a specific utility-preserving cryptographic primitive method and pick the machine learning algorithms they are most familiar with. As a result, the important developments are scattered with focuses on either a specific machine learning model or the application of a novel cryptographic primitive. There is no thorough understanding of which primitive method (or framework) is best for a specific machine learning method or whether a CML method can be extended to other machine learning models. The fundamental principles are missing for solving all (or most) CML model training methods.

Categories of CML approaches. We believe this survey is the first effort to systematically organize and analyze the whole body of most representative CML approaches. We focus on the major category of methods: the pure software-based *cryptographic protocols*, while also briefly reviewing the *perturbation-based* approaches and the *hardware-assisted* approaches. Figure 3 shows the systematization framework. The fundamental features of the three categories are as follows.

- The cryptographic protocols are the focus of this survey, which can be further divided into two categories: those using one cryptographic primitive homogeneously and those employing novel hybrid compositions of multiple primitives. The homogeneous approaches take one of the homomorphic encryption (HE) schemes or garbled circuits to develop the solution. The hybrid approaches involve multiple primitives and often a clever composition strategy to achieve lower overall costs. We will analyze them in more detail.
- The perturbation-based CML approaches depend on novel data transformations to preserve a certain type of data utility, e.g., Euclidean distance, that is critical to one or multiple machine learning methods. The security of perturbation-based CML approaches mainly depends on secret transformation parameters and random noise addition, holding a different and somewhat weaker security notion compared to cryptographic protocols. However, they are often much more efficient and thus appealing for many applications that seek better protection than plaintext-based approaches while not taking significantly more overhead.
- The third category depends on *trusted execution environment*, such as Intel SGX (Costan and Devadas 2016), which demands hardware-level supports and are thus distinct from the former two categories of pure software approaches. The hardware-level features enforce *secure enclaves*, in which the adversaries cannot observe the running programs and data.

Common CML Development Strategies. We look into a unified framework to analyze both the homogeneous and hybrid approaches. Fundamentally, most approaches aim to design an efficient and secure transformation of the specific (or a class of) machine learning algorithms for

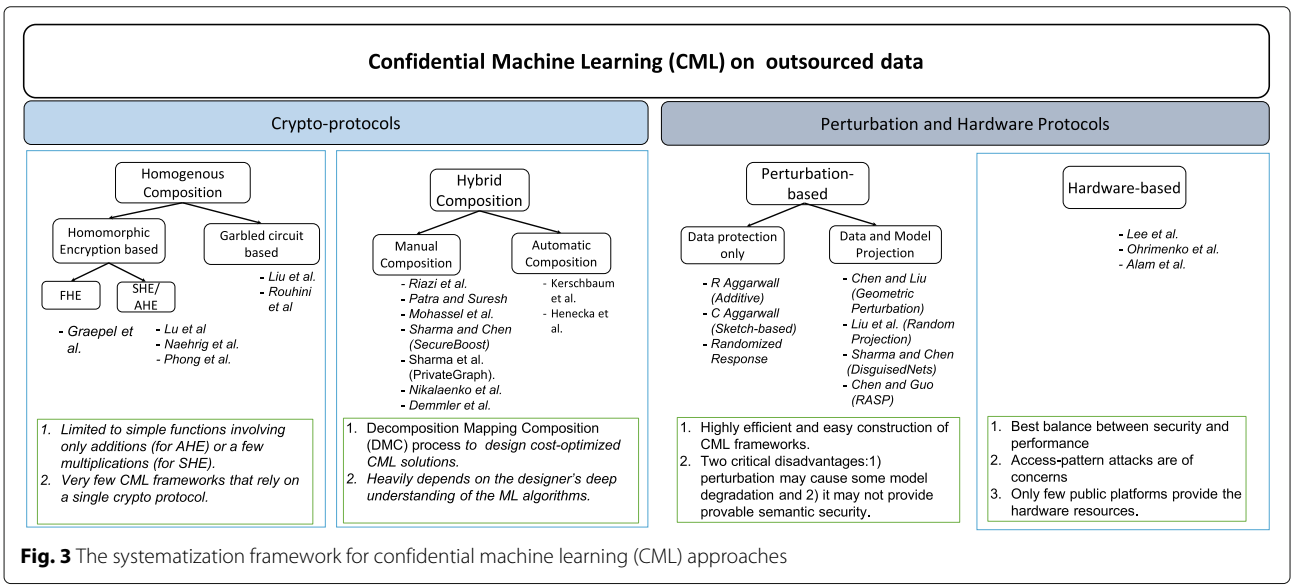

Fig. 3 The systematization framework for confidential machine learning (CML) approaches

the setting of two or three distributed parties (see System architectures section). To make the transformation easier, researchers often implicitly use the Decomposition-Mapping-Composition (DMC) procedure depicted in Fig. 4: decomposing the target algorithm into different subcomponents, mapping the sub-components to crypto-primitives, and composing the CML framework with the confidential sub-components. Many approaches skip the description of this whole procedure and only present the final composition, which creates difficulties for newcomers to fully appreciate the fundamental ideas scattered in several approaches.

Beyond the straightforward DMC procedure, we have also noticed a unique feature (Sharma and Chen 2019) specific to the CML development: finding "crypto-friendly" alternative machine learning algorithms or components. This feature is unique to machine learning algorithms because all machine learning algorithms essentially try to find an approximate model fitting the training data, and there is no unique model for a specific problem, only better or worse ones. In general, machine learning methods can be roughly categorized into two types: supervised learning that depends on labeled datasets and unsupervised learning (Hastie et al. 2001). For each type, there are numerous algorithms working under the same setting but performing differently for specific applications or datasets. Even for the same algorithm, there are many variants. For example, different base classifiers can be used to make ensemble classifiers (Schapire 1999), and different activation functions can be used for neural networks (LeCun et al. 2015). Among so many machine learning algorithms, some are more crypto-friendly, i.e., they can be converted to more efficient CML solutions.

With all these features in mind, we reassemble the common development framework behind most CML approaches (Procedure 1).

Note that most of the steps in this procedure cannot be automated, and thus each specific approach represents a result of enormous efforts behind the scene. Next, we analyze the homogeneous and hybrid approaches under this unified procedure.

Homogeneous cryptographic approaches

Homogeneous approaches rely on a single primitive to construct the framework protocols. The primitives used in the homogeneous composition of CML are broadly in

Procedure 1 A common procedure for developing CML methods

procedure GENERALIZED PROCEDURE FOR CML DEVELOPMENT(A)
  A: the target algorithm
  Identify the desired architecture and involved parties.
  Identify a list of alternative algorithms of A
  for Each candidate algorithm **do**
    decompose the algorithm to basic components
    for Each component **do**
      identify possible approximate/equivalent solutions
      for Each solution **do**
        identify candidate crypto-primitive mappings
      end for
      select the best solution and mapping.
    end for
    find the best composition method.
  end for
  evaluate the candidate alternative algorithms and identify the best one.
end procedure

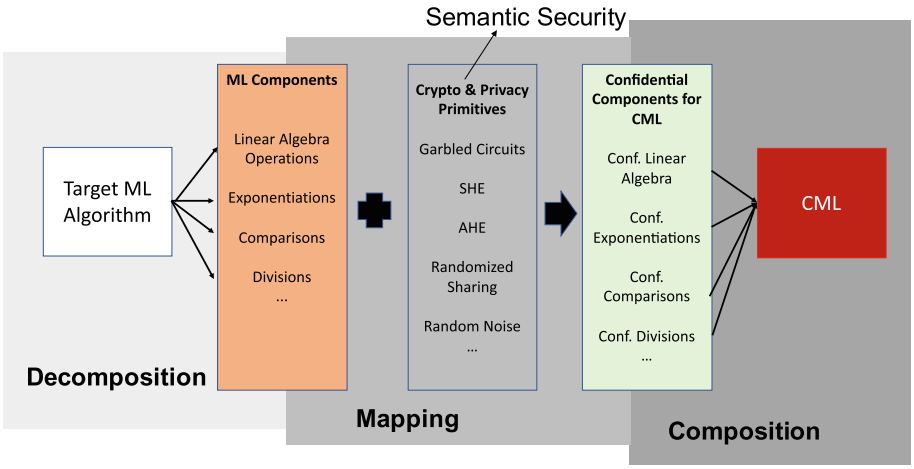

Fig. 4 The decomposition-mapping-composition (DMC) process for constructing hybrid CML solutions

two categories: (1) Fully Homomorphic Encryption (FHE) and Garbled Circuits (GC) and (2) Additively Homomorphic Encryption (AHE) and Somewhat Homomorphic Encryption (SHE). Since FHE implements arbitrary levels of homomorphic addition and multiplication and GC implements the boolean gates, in theory, they can individually construct all CML algorithms. FHE and GC are, therefore, the most expressive privacy primitives. However, both FHE and GC are too expensive to be practical when mapped to for training complex CML models. Oppositely, AHE and SHE schemes provide limited support for encrypted operations, therefore, less expressive and can only enable relatively simple algorithms. Most approaches we discuss next are relatively simple, and thus AHE or SHE scheme is sufficient. The decomposition and mapping steps of the DMC procedure described in the last section are still at play in the homogeneous approaches, but the composition step is trivial.

AHE and SHE are widely used to construct homogeneous solutions for applications involving only one or a few multiplications, including the elementary statistical aggregation functions, such as average, sum, and variance. Graepel et al. (2012) present a SHE-based framework for learning Fisher's linear discriminant analysis and Linear Means Classifier models on encrypted data. However, the implemented models are limited to linearly separable datasets. Lu et al. (2016) apply SHE for more sophisticated principal component analysis, and linear regression training (Hastie et al. 2001). However, due to the limited message space of the selected SHE implementation (60-bits in HELib) and the limited number of possible multiplications, only low data dimensionality (about 20) and a few training iterations were used in their evaluation. Such restrictions, however, resulted in only sub-optimal models.

More sophisticated machine learning algorithms often result in expensive homogeneous solutions. Phong et al. (2018) employ LWE and Paillier encryption in encrypting the gradients in their privacy-preserving deep learning framework. The framework, however, takes over 2.5 h to complete one iteration of a simple neural network training for 20,000 MNIST images. Researchers also aim to provide libraries for homogenous learning based on Garbled Circuits (GC). However, their uses are limited in practicality due to huge costs (Liu et al. 2015). Liu et al. (2015) present a GC-based KMeans learning framework that involves two untrusted servers. The associated cost overburden, however, is far from efficient in real-world settings. For example, the KMeans implementation required over 2,000 million AND gates and more than 200 GB communication for clustering just 6,000 data points. Rouhani et al. (2018) propose a deep learning model inference frameworks using garbled circuits to protect both the model's parameters and test data samples. Similar to other homogoenous frameworks, the costs are staggeringly high.

Insight. *Homogeneous solutions are often limited to simple functions involving only additions (for AHE), a few multiplications (SHE), or a few comparisons (GC). Individually, these crypto primitives are not practical to construct complex CML algorithms. However, even the less expressive primitives can be valuable components for hybrid solutions, as we will see later.*

Hybrid composition

As discussed above, depending on a single cryptographic primitive to compose a sophisticated CML algorithm is impractical. However, each primitive has its unique strengths and shortcomings (e.g., performance, storage, bandwidth advantage, etc.) in attaining certain operations. This realization leads to an interesting strategy: can we combine different primitives in such a manner to compose secure yet more optimized protocols? The idea of hybrid composition is thus, mixing and switching amongst several privacy primitives to avoid the associated cost bottlenecks and restrictions of any individual primitive.

This section will look into the details of specific steps of the DMC procedure. First, we dissect the common sub-components and underlying operations in machine learning algorithms. We examine the various ways to implement these sub-components and operations confidentially. Then, we explore the different switching and mixing strategies, including some recent automated ones, essential to hybrid CML frameworks in practice. Finally, we discuss the unique feature or desired requirement of CML development: designing crypto-friendly machine learning algorithms or sub-components for cost-efficient and practical CML solutions.

Basic operations

We devote this subsection to inspecting the mapping of the foundational sub-components of the target machine learning algorithms to their confidential versions. We observe that some of these mappings are practical or crypto-friendly, whereas others may face cost bottlenecks and limitations. The understanding of the different implementations of basic operations will affect the composition strategies.

Simple Arithmetic Operations With AHE or a SHE encryption scheme, one can conveniently add two encrypted integers. Adding two b bit integers with the Paillier cryptosystem involves modular multiplication with $O(b^2)$ complexity. Additions with an RLWE-like scheme involve polynomial additions linear to the number of bits for the given polynomial degree (Chakarov and Papazov 2019). With a specific integer encoding, subtraction becomes trivial expressed as encrypted

additions. SHE schemes allow homomorphic multiplications over encrypted integers. RLWE-like crypto-systems allow several rounds of multiplications and additions. However, with each additional multiplications, the ciphertext noise, cipher size, and cost increase. Generally, multiplying two b bit integers with RLWE-like crypto-systems involves homomorphically computing $O(b^2)$ AND circuits (Chakarov and Papazov 2019). On the other hand, the AHE scheme requires one of the operands to be unencrypted to realize multiplication expressed as summations. With Paillier encryption, multiplication is modular exponentiation of encrypted b-bit message by the unencrypted b-bit operand with a cost complexity of $O(b^3)$. The only caveat of using AHE-multiplication is that if the unencrypted operand is privacy-sensitive, a mechanism to mask it needs to be augmented, the masking recoverable after the multiplication is complete (Sharma et al. 2019; Nikolaenko et al. 2013).

Additions and subtractions are trivial with randomized secret sharing in the multi-party setting with constant time complexity. Each party performs additions and subtractions on respective shares of data and shares the results for recovery. A GC protocol for addition requires two parties to construct $O(b)$ many AND gates and carry out $O(b)$ communication, encryptions, and decryptions along with $O(b)$ oblivious transfers when adding two b bit integers. Multiplication with randomized secret sharing involves a costly multiplicative triplet generation scheme that relies on Oblivious transfer or AHE (Demmler et al. 2015; Mohassel and Zhang 2017). For example, the AHE-based scheme incurs transmission of two encrypted integers between the parties and performing two homomorphic encryptions, multiplications, additions, and decryptions by each party. Multiplying two integers of b bits with GC, on the other hand, requires construction and evaluation of $O(b^2)$ AND gates involving $O(b^2)$ communication, encryption, and decryption.

Comparison. Comparison is essential in many operations, such as sorting vectors and applying activation functions in training neural networks. Unfortunately, comparing two encrypted or protected integers is not trivial. Graepel et al. (2012) pose the complexity of comparison as the reason to avoid algorithms like perceptrons and logistic regression in their SHE-based confidential ML framework. Veugen (2014) presents a client-server interactive comparison protocol for two encrypted integers based on the AHE scheme, which involves computation and transfer of b many AHE encrypted bits. Each comparison incurs $O(b)$ homomorphic multiplications for both client and server. Lu et al. (2016) use the technique of "greater than" protocol (Golle 2006) optimized with the message packing of the RLWE scheme for comparing two encrypted messages in a two-party setting. However, the associated complexity is an astonishing $O(2^b/h)$ of homomorphic additions when comparing two b-bit integers while packing h messages in a ciphertext. With GC, a comparison between two b-bit integers is possible with $O(b)$ AND gates and $O(b)$ communication, encryption, and decryption by two parties. Since GC-based comparison for full integers is expensive, one may use an efficient one-bit sign checking protocol (Mohassel and Zhang 2017; Sharma and Chen 2019) by encoding negative integers as two's complement, making the comparison cost is constant to the number of bits. Note that the GMW protocol of Goldreich et al. (1987) can perform comparisons just as garbled circuits but with $O(b)$ rounds. A similar sign-checking protocol is possible with GMW. However, the GC-based comparison seems the popular choice in current solutions.

Division. Division can be essential to many analytics algorithms, e.g., from the computation of mean to the implementation of complex algorithms such as K-means (Bunn and Ostrovsky 2007) and Levenshtein distance (Rane and Sun 2010). Despite its prevalence and importance, translating division to its confidential version is expensive and often results in a performance bottleneck (Lazzeretti and Barni 2011). Veugen (2014) presents a protocol for exact division in a client-server scenario, using the AHE scheme and additive noise masking. However, the protocol requires the divisor to be public knowledge. On top of that, the protocol requires $O(b)$ homomorphic comparisons and $O(b)$ encrypted communication for division between two b-bit integers. Dahl et al. (2012) present two AHE-based division schemes that rely on Taylor approximation in a secure multi-party setting. The schemes brought expensive $O(b)$ encrypted communication. It is possible to perform integer divisions with GC when the two parties hold the numerator and denominator respectively in a 2-party setting (Lazzeretti and Barni 2011; Nikolaenko et al. 2013). However, even with several optimizations, a division between two b-bit integers involves the construction and evaluation of a circuit with $O(b)$ non-XOR gates (Lazzeretti and Barni 2011). A more practical solution would be to decrypt the operands at a crypto-service provider and conduct division on plaintext before finally encrypting the result.

Linear Algebra Operations. Linear algebra operations, such as vector dot products, matrix-vector multiplication, and matrix-matrix multiplications, are the core operations for many machine learning algorithms. They are commonly implemented with the cryptographic versions of additions and multiplications with some tricks in RLWE-based SHE for improved efficiency. Among all available methods, the AHE and SHE-based implementations are the most efficient ones.

A dot product $x_k^T y_k$ involves $O(k)$ element-wise homomorphic multiplications and additions. Similarly, a matrix-vector multiplication $A_{n\times k}x_k$ involves $O(nk)$

homomorphic multiplications and additions, and a matrix-matrix multiplication $A_{n\times k}B_{k\times m}$ involves $O(nkm)$ multiplications and additions. With the AHE scheme, one of the operands must remain unencrypted for these multiplicative operations. Therefore, the unencrypted operand needs some level of protection, e.g., novel randomized masking (Sharma et al. 2019) with a minimized cost. With the message packing feature for the RLWE-like SHE scheme, one can easily vectorize the vector and matrix operations with message packing to gain more efficiency (Garay and Gennaro 2014). With such facilities, Jiang et al. (2018) can optimize matrix-matrix multiplication with only $O(k)$ complexity for symmetric matrices of k dimensions.

Randomized secret sharing enables linear algebraic operations with the multiplicative triplet generation approach in a multi-party setting. However, this involves the expensive AHE or OT-based multiplicative triplet generation schemes as used in Mohassel and Zhang (2017); Demmler et al. (2015). In computing a matrix-vector multiplication Ab, each party is responsible for $O(n + k)$ encryptions and upload, $O(nk)$ homomorphic multiplications, $O(nk + n)$ homomorphic additions, and $O(n)$ decryptions.

One can easily map linear algebra operations to garbled circuits. GC-based vector and matrix addition/subtraction require $O(kb)$ and $O(nkb)$ AND gates where b is the number of bits in the vector and matrix elements. They also result in $O(kb)$ and $O(nkb)$ communication, encryption, and decryption operations, respectively. GC-based dot product for two b bit vectors with k dimensions is a collection of sub-circuits for multiplication and additions, which consist of $O(kb^2)$ AND gates. The cost also involves $O(b^2)$ encryption and decryption, and $O(b^2)$ encrypted communication. The GC-based dot product can easily extend to matrix-vector and matrix-matrix multiplication. However, GC-based linear algebra solutions are more expensive than HE-based ones.

Empirical cost comparison

We have formally analyzed different crypto implementations for each of the major operations. However, some of them look close in terms of bigO complexity levels. To have a better idea how the cost differences look like for the different implementations of the same operator, we also prepare Table 1. Since this comparison rests on a specific hardware configuration and software implementation, readers should only focus on the relative differences rather than the actual numbers. After a careful study of available AHE and SHE implementations, we choose the most efficient one for each category: we use the HELib library (Halevi and Shoup 2013) for the RLWE encryption scheme and implement the Paillier cryptosystem (Paillier 1999) for the AHE encryption scheme. We adopt the ObliVM (oblivm.com) library for the garbled circuits. We also take the AHE scheme for the multiplicative triplet generation when using the randomized secret sharing (SecSh) method. We pick cryptographic parameters[2] corresponding to 112-bit security. All schemes allow at least 32-bit messages-space overall. The RLWE parameters allow one full vector replication and at least two levels of multiplication. Note that the GC and SecSh costs are for the two-party setting, which has to involve communication costs between the two parties. Thus, we also include the bytes of exchanged messages for these methods. We run the experiments on an Intel i7-4790K CPU running at 4.0 GHz using 32 GB RAM with Ubuntu 18.04.

Table 1 compares the related costs of arithmetic operations over integers. We have observed that the AHE scheme has the most efficient arithmetic additions and multiplications. However, for comparison and division, the 2-party garbled circuits are the only viable option. The table also shows the costs for the linear algebraic operations. The observation is consistent with the simpler arithmetic operation of additions and multiplications. As we can fit multiple messages in a ciphertext when using the RLWE scheme, the vectorized additions and multiplications are much more efficient than the non-vectorized additions and multiplications. The RLWE with message packing realizes homomorphic additions more efficiently when compared to the Paillier scheme. The RLWE costs for dot product and matrix-vector multiplication involve the ciphertext replication costs. Although better than without message packing, the RLWE scheme with the vectorized linear algebraic operation is still slower than the Paillier solutions. Randomized secret sharing is almost free for vector addition but involves higher computation and communication costs for the dot product and matrix-vector multiplication. Garbled circuits appear to be the worst solution for the confidential versions of the linear algebraic operation with higher computation and communication costs between the two parties. Although the Paillier implementation shows performance advantages over RLWE on arithmetic operations, it requires one operand to be plaintext. Paillier's encryption and decryption costs, however, are higher than that of RLWE (Sharma et al. 2019). When CSP is involved in a solution, encryption and decryption costs will become a critical performance factor. These cost comparisons on the basic operations will be useful for readers to analyze and compare a pair of CML protocols, especially when not all CML methods are open-source.

[2] The Paillier cryptosystem uses a 2048-bit key size. We set the degree of the corresponding cyclotomic polynomial in the RLWE scheme to $\phi(m) = 12,000$ and c = 7 modulus switching matrices, which gives us h = 600 slots for message packing.

Table 1 Real cost comparison for confidential arithmetic and linear algebra operations at 112-bit security, $v_{100\times1}$ and $M_{100\times100}$

	AHE (Paillier)	SHE (RLWE)	Garbled Circuits		Secret Sharing	
	Comp	Comp	Comp	Comm	Comp	Comm
Addition/Subtraction	0.01 ms	0.2 ms	37 ms	2 KB	0.0 ms	0.0 KB
Multiplication	0.05 ms	39 ms	138 ms	40 KB	1 s	2 KB
Comparison	429 h	$10^5 h$	37 ms	2 KB	-	-
Division	-	-	208 ms	46 KB	-	-
Vector Addition	0.6 ms	0.2 ms	36 ms	192 KB	0.0 ms	0.0 KB
Dot Product	6 ms	39 ms	5 s	4 MB	7 s	195 KB
Matrix-vector Multiplication	1 s	3 m	8 m	396 MB	7 s	290 KB

We do not experimentally compare complete CML approaches because 1) different approaches often solve different ML problems, which makes the comparison difficult, and 2) not all approaches have open-sourced their implementation or shared executable binaries. However, we hope the empirical comparison between different implementations for basic operators gives an intuitive understanding of the rationales behind different CML design strategies and optimization methods. We refer readers to the papers describing CML approaches that often contain detailed performance comparisons between selected CML approaches.

Insight. *Based on most studies, the most efficient constructions for confidential comparison are GC-based, while SHE and AHE are better candidates for linear algebra operations. Since most division schemes are too expensive, one should consider transforming the functions/algorithms with divisions to the equivalent (often approximately) ones that involve no division.*

Switching and composing strategies

When composing the confidential versions of operations implemented with different primitives, there is an important step: switching computation flows between the primitives. This switching often requires a second party in the CML frameworks, i.e., either the data owner, the second non-colluding cloud, or a cryptographic service provider (CSP) to achieve better performance.

HE to/from GC. Switching from a HE component to a GC component involves a second server (e.g., a CSP) in the framework. A straightforward approach would be including a data decryption circuit inside a garbled circuit to be evaluated by the two parties. However, such an approach is super-expensive (Nikolaenko et al. 2013). A more practical strategy (Nikolaenko et al. 2013; Nikolaenko et al. 2013; Sharma and Chen 2019) is to have the party holding the encrypted data, denoted P_A, mask it homomorphically before sending it to the second party, P_B for decryption. The second party constructs the desired garbled circuit, where the first step of the garbled circuit is de-masking the data with inputs: the decrypted masked data from P_B and the mask from P_A.

SecSh to/from GC. Switching from a SecSh component to a GC component is straightforward in a two-party architecture. The two random shares in possession of the two parties can be their respective private inputs to the desired garbled circuits (Mohassel and Zhang 2017; Sharma and Chen 2019; Riazi et al. 2018). Similarly, switching from GC to SecSh involves evaluating the GC and randomly distributing the output to two parties (Riazi et al. 2018).

SecSh to/from HE. A switch from randomized secret sharing to a HE component needs two involved parties to encrypt their respective shares. Then, one of the parties homomorphically reconstructs the protected value from the shares. Similarly, a switch from a HE component to a randomized secret sharing protocol includes a masking mechanism (homomorphic noise addition) similar to the HE-to-GC switch discussed above. These two switches are relevant in the AHE-based multiplicative triplet generation protocol for randomized secret sharing (Mohassel and Zhang 2017; Demmler et al. 2015).

Table 2 provides some examples of switching between cryptographic primitives in well-known CML approaches. These switchings lead to simplification of the CML framework and cost optimizations, as explained in the "Justification" column of the table. The ABY framework (Demmler et al. 2015) covers different adapter-like switching protocols for the multi-party computation settings, where two servers hold the training data as arithmetic, boolean, or Yao's garbled shares. The ABY3 (Mohassel and Rindal 2018) and BLAZE(Patra and Suresh 2020) framework extend the switches to 3-party scenarios. These works, however, do not cover the switching from and to the homomorphic encryption schemes.

Manual vs. Automated Composition. Most existing CML approaches using the hybrid composition strategy (Mohassel and Zhang 2017; Sharma and Chen 2019; Sharma et al. 2019; Nikolaenko et al. 2013) are manually

Table 2 Examples for primitive switching strategies in hybrid composition of CML frameworks

Framework	Primitive Switch	Operation Switch	Justification
Sharma and Chen (2019)	SHE → GC	Matrix vector multiplication → Sign Check	Sign checking is impractically expensive with SHE whereas tolerable with GC.
Nikolaenko et al. (2013)	AHE → GC	Matrix Additions → Cholesky's decomposition	The operations of division and square root in Cholesky's decomposition were not feasible with the AHE scheme.
Nikolaenko et al. (2013)	AHE → GC	Matrix Additions → Gradient Descent	Gradient descent involved multiplications, additions, and subtractions not entirely feasible with the AHE scheme.
Mohassel and Zhang (2017)	SecSh → GC	Matrix-vector multiplication → Comparison	Comparison is impossible over randomly shared secrets leading the switch to the garbled circuits.
Mohassel and Zhang (2017)	GC → SecSh	Comparison → Vector Subtraction	Use of garbled circuits for comparison was unavoidable however continuing GC on to vector subtraction would result in excessive cost overhead.
Demmler et al. (2015)	SecSh → AHE/OT	Data at rest → Multiplication	Multiplication with random shares required switching to either AHE or OT protocol involving the two parties in the frameworks.
Riazi et al. (2018)	SecSh → GC	Matrix matrix multiplication → ReLu computation	Sign checking is impossible over randomly shared secrets leading the switch to garbled circuits.
Riazi et al. (2018)	GC →SecSh	ReLu → Matrix vector multiplication	Use of garbled circuits for matrix vector multiplication is impractical.

composed as there are myriads of problem-specific details to address. A line of research explores the possibility of automatically composing the CML frameworks (Dreier and Kerschbaum 2011; Henecka et al. 2010). Although promising, the automatic composition strategy of Dreier and Kerschbaum (2011) depends on the availability of an extensive performance matrix for the different confidential versions of the target algorithms' components. Henecka et al. (2010) propose the TASTY compiler that automatically compiles a given machine learning problem as a mixture of garbled circuits and homomorphic encryption in a secure two-party computation framework. However, the process is still not fully automated - it requires a privacy expert to design and specify the components as well as the recommended mappings.

Gap. *Due to the high complexity of formulating the component-wise costs and profiling the switching costs, the automated composition approaches are not yet fully mature. More importantly, as we will see in the next section, the construction of a practical CML solution involves one more crucial step that automated composition methods cannot help much. One must establish an in-depth understanding and analysis of the target ML algorithm to redesign a "crypto-friendly" algorithm.*

Crypto-friendly ML algorithms

So far, the DMC framework seems straightforward: one decomposes the target machine-learning algorithm to its sub-components and maps them to cryptographic constructions, and the final composition becomes almost trivial except that the primitive switching requires some clever steps. With enough experimentation, one can find an optimal set of confidential components for the target ML algorithm. However, this straightforward strategy may only work for some problems. Despite the best optimization of mapping and composition, one may still end up with an impractical protocol, although better than the homogeneous or other suboptimal compositions. The fundamental reason is that the original machine learning algorithms do not account for confidential computation. ML algorithms are optimized to achieve the best model prediction power rather than to be crypto-friendly. On the other hand, a less-known slightly-under-performing ML algorithm that attains the same learning goal might be more cost-effective to translate to its confidential version. Thus, an advanced design step critical to the DMC procedure is replacing or redesigning some of the underlying ML components or even the entire ML algorithm to find the most efficient CML protocols. Table 3 summarizes some example CML frameworks that incorporate strategies to make their protocols crypto-friendly and hence more cost-effective.

Mohassel and Zhang (2017), in their SecureML work, substitute the expensive softmax operation involving inverses with a ReLU-based function involving only one division. This way, the framework significantly reduces the cost bottlenecks in their protocol. Graepel et al. (2012) cleverly avoid division of encrypted data in the framework for confidential linear means classifier and Fisher's linear

Table 3 Example CML methods that replace the expensive algorithmic components with their crypto-friendly versions

Framework	ML Algorithm	Original Component	Crypto-friendly Component	Benefits
Mohassel and Zhang (2017)	Logistic Regression, Neural Networks	Sigmoid, Softmax	ReLu	Avoids inversion and limits expensive confidential divisions to one.
Graepel et al. (2012)	LMC, Fisher's LDA	Divisions	Multiplications with incorporated division factors	Avoids division costs and simplifies the protocol.
Nikolaenko et al. (2013)	Ridge Linear Regression	LU decomposition	Cholesky's decomposition	Reduces the cost complexity by half.
Nikolaenko et al. (2013)	Matrix Factorization	Cholesky's Decomposition	Sorting based matrix factorization	Reduces the overall complexity from quadratic to within a polylogarithmic factor of the complexity in the plaintext
Sharma and Chen (2019)	Boosting	Decision Stumps	Random Linear Classifiers	Reduced number of comparisons and simplicity in learning.
Naehrig et al. (2011)	Logistic Regression	Exponentiation	Taylor Expansion	Avoids costs involved in multiple levels of multiplications.
Sharma et al. (2019)	Spectral Clustering	Eigen decomposition	Eigen-approximation by Lanczos and Nystrom	Reduces complexity of the problem from $O(N^3)$ to $O(N^2)$.

discriminant analysis by replacing divisions with a multiplicative factor. Nikolaenko et al. (2013) use the more efficient Cholesky's decomposition instead of the expensive LU decomposition in solving a system of linear equations in their linear regression framework. Similarly, Nikolaenko et al. (2013) adopt the sorting-based matrix factorization solution to reduce the overall complexity of computing gradient descent with Cholesky's decomposition-based matrix factorization. Sharma and Chen (2019) propose to train a boosting classifier over encrypted data with an ensemble of random linear classifiers (RLC) instead of decision stumps. An RLC takes mere N encrypted comparisons, whereas a decision stump takes far too many comparisons. Naehrig et al. (2011) replace the exponential function (the sigmoid) in their logistic regression protocol with the Taylor approximation of exponentiation. Computing the exact exponential function would have led to the computation of many levels of multiplications over the encrypted message – which would have been intolerably expensive with SHE schemes. Similarly, Sharma et al. (2019) replace the inherently expensive eigendecomposition $O(N^3)$ with cheaper $O(N^2)$ approximation algorithms of Lancozs and Nystrom in their spectral clustering framework.

Data reduction techniques such as subsampling and preserving the sparsity of matrix are also critical to performance. Nikolaenko et al. (2013), in their matrix factorization framework, use a sorting network that optimizes the garbled circuit-based gradient descent algorithm by only updating it for the user ratings that are present in the training dataset. Similarly, Sharma et al. (2019) propose a differential privacy-based graph submission mechanism that reduced total storage by over 15 times and costs involving encryptions and the associated homomorphic operations by over 20 times on the graph drastically when running the secure Nystrom method for spectral clustering. To sum up, although the approximate algorithms introduce some degradation to the learned models, they deliver desired cost practicality justifying the tolerable quality sacrifice.

Insight. *For the same learning problem, there are numerous algorithms. Even for the same learning algorithm, there are many variants (Hastie et al. 2001). The search space for optimal composition can be quite large. More difficultly, most well-known ML algorithms are best known for model quality or learning efficiency and none specifically designed with optimal CML in mind. Even worse, some crypto-friendly alternatives might have been forgotten or become obsolete due to their suboptimal quality or efficiency. The design of a good CML solution heavily depends on the designer's deep understanding of the ML algorithms and even the history of ML algorithm development.*

Gap. *There is no systematic way to explore crypto-friendly alternative ML algorithms. The current practice is to heuristically design a problem/algorithm-specific crypto-friendly solution. Although the problem-specific design experiences and learnings can extend to a new solution design, there are no well-known rules or general frameworks for exploring such alternative ML algorithms yet.*

Security proofs, attacks, and correctness

In this section, we summarize the three aspects: security proofs, attack analysis, and correctness for existing CML approaches.

Security Proofs. Homogeneous approaches do not use complex protocols other than the cryptographic primitive they use. For example, homomorphic encryption-based approaches involve only simple interactions between the client and the cloud - the client submitting the data and the cloud computing and returning the result; the GC-based methods have two involved parties following the fundamental GC protocols. Thus, most such homogenous approaches simply skip the security proof step, fully depending on the proven security and privacy guarantees provided by the underlying primitives.

For hybrid approaches, it is more sophisticated to prove their security, as they may include complex interactions among parties. We have observed two security proof frameworks are in prevalence. SecureML (Mohassel and Zhang 2017) utilizes the Universally Composable Security (UC) framework (Canetti and Canetti 2001). The UC security framework defines security-preserving universal composition operation and allows for modular design and bottom-up analysis of complex cryptographic protocols from simpler building blocks. PrivateGraph (Sharma et al. 2019), SecureBoost (Sharma and Chen 2019), and Lu et al. (2016) adopt the simulation-based security proof (Lindell 2017). Both approaches need to show the existence of a *simulator* in the ideal scenario that corresponds to the adversary in the real scenario, such that it is impossible to distinguish the interactions in the ideal scenario from those in the real scenario. The assumption of semi-honest parties held by most CML approaches makes the security proofs much easier (Lindell 2017; Canetti and Canetti 2001). As a result, many CML approaches ignore the steps of security proof.

Attacks. To our knowledge, attacks on the confidentiality of cryptographic CML approaches have not been fully explored. Most works we covered in this category did not mention any potential attacks on their approaches, partially due to the well-known security guarantees provided by the underlying primitives or formal security proofs provided by a few approaches. While all approaches want to fully protect feature vectors in the training data, some approaches require the labels (in supervised learning) to be exposed for easier modeling (Graepel et al. 2012), and some even expose the final learned models (Nikolaenko et al. 2013; Lu et al. 2016). However, recent studies have shown that exposed models may lead to serious attacks, such as model inversion attacks (Fredrikson et al. 2015; Tramèr et al. 2016), and membership inference attacks (Shokri et al. 2017).

Correctness. Contrary to some cryptographic protocols and encryption systems that need to prove their correctness (e.g., encrypted values can be correctly decrypted), the correctness of CML protocols is attached to the correctness of the original machine learning algorithms. The DMC procedure honestly reassembles the original learning algorithm with the cryptographic components. Thus, as long as the primitives preserve the correctness and the composition strategy does not change the correctness (see Switching and composing strategies section), the correctness property is guaranteed. However, when researchers adopt a crypto-friendly alternative algorithm or component, they must justify whether the alternative methods warrant/attain the desired learning objective. SecureBoost (Sharma and Chen 2019) depends on the basic boosting theory (Schapire 1999) that states any weak base classifier, including random weak linear classifiers, can be used for the boosting framework. Naehrig et al. (2011) utilize the Taylor approximation of exponentiation to approximate the sigmoid function, which is a well-accepted mathematical method. While these alternative methods may affect the model quality, implying a potential trade-off between model quality and costs, they are all considered correct algorithms.

Gap. *Security proofs are missing for some existing CML approaches, which raise a concern that they may contain flaws leading to significant information leaks. Further studies are needed to rigorously analyze these approaches.*

Evaluation methods

Researchers evaluate their proposed CML methods primarily based on costs and model quality. Some CML methods also involve trade-offs between these two aspects.

Costs. CML researchers primarily concern about the costs of protocol, striving to find the most efficient secure protocols. Since multiple parties are involved, the costs for each party, i.e., the cloud provider, the client, and possibly the crypto-service provider or the second cloud provider, are all essential to the design of CML protocols. For a given CML method, each party's costs are the outcome of the cost for comparing the encryption/ decryption, data transmission, and other computation overhead. Because of the original motivation of outsourcing large-scale computation, a skewed cost distribution between the client and the cloud is fundamental, i.e., the client should take much lower overheads compared to the cloud (Sharma et al. 2019; Sharma and Chen 2019; Mohassel and Zhang 2017). However, the client may still take much higher costs when running CML protocols when compared to running the original non-secure ML solution. The cost of external storage and related I/O operations are also critical to the cloud-side components as they are responsible for storing the encrypted data, which often is much larger than the plaintext version and cannot reside in memory. It is also highly desired that the cloud-side computation can

be done parallelly with a popular processing framework such as MapReduce (Dean and Ghemawat 2008; Sharma et al. 2019). Besides, when GC is adopted as a primitive to implement some components, additional communication cost related to the GC protocol is also significant, including the cost of transmitting the circuit and one-party's input data obliviously to the other party (Liu et al. 2015; Huang et al. 2011). As a result, the use of GC is limited to a few operations, such as comparison (Demmler et al. 2015). The overall computation and communication costs of different approaches are frequently compared and used as a measure to show the novelty of a new method. For example, Mohassel and Zhang (2017) show their work is more computation efficient than the GC-based framework considered by Nikolaenko et al. (2013) by about two orders of magnitude. Similarly, Sharma and Chen (2019) show their boosting solution is about three times faster than the neural network CML in Mohassel and Zhang (2017).

Model Quality. Model quality, a unique feature of CML evaluation, is often tightly related to the cost of model training. Many machine learning algorithms are iterative, such as logistic regression, neural networks, and many clustering algorithms. As a result, model quality increases with the number of iterations until the process converges. However, a large number of iterations implies the increased overall costs. Some CML methods, e.g., Lu et al. (2016), may only report the overall costs for one/few iterations of a specific learning algorithm, which is insufficient unless the number of iterations necessary for optimal results is specified. More precisely, many works miss the requirement that model evaluation should be tied to the cost evaluation, i.e., how much cost is needed to reach a certain model accuracy (Mohassel and Zhang 2017; Sharma and Chen 2019). The discussion on crypto-friendly alternative algorithms also holds the assumption that model quality can be possibly traded off with costs, with the expectation that the crypto-friendly alternative may perform comparably or slightly worse than the original machine learning algorithm (Sharma and Chen 2019; Sharma et al. 2019; Mohassel and Zhang 2017; Graepel et al. 2012).

Other CML approaches

So far, we focused on cryptographic methods based on well-known primitives. To cover a panoramic view of development in the growing area of confidential machine learning, we briefly discuss two closely related approaches, the perturbation-based approach and the hardware-assisted approach in this section.

Perturbation methods

Most practical CML solutions that carefully follow the DMC process with some innovative uses of crypto-friendly ML algorithms still cost magnitudes more than the original plaintext algorithms. Especially if the learning algorithm is intrinsically expensive or relies on a massive-scale training dataset, the cryptographic primitives that provide semantic security may become impractically expensive, discouraging users from adopting the outsourcing paradigm. Another category of work: the perturbation-based approach offers much more efficient solutions with some weaker security notions. Often, these methods do not guarantee semantic security and may only be resilient to ciphertext-only attacks. Nevertheless, they can be interesting for users who are willing to make a practical trade-off between efficiency and the level of protection. We briefly discuss this body of work to extend readers' interests to this unique domain.

The basic idea of perturbation is injecting random noises into the outsourced data while (approximately) preserving some specific properties machine learning models rely upon. The most well-known properties are geometric and topological structures in the multidimensional space. Therefore, one can still train a model from the perturbed data on the untrusted platform with preserved confidentiality of both data and model. Typical perturbation methods include randomized response (Erlingsson et al. 2014; Du and Zhan 2003), additive perturbation (Agrawal and Srikant 2000), geometric perturbation (Chen and Liu 2011), random projection perturbation (Liu et al. 2006), and random space perturbation (Xu et al. 2012). They have been applied to decision tree learning (Du and Zhan 2003; Agrawal and Srikant 2000), clustering (Chen and Liu 2011; Liu et al. 2006), kNN classifier (Chen and Liu 2011), support vector machines (Chen and Liu 2011), linear classifier (Chen and Liu 2011; Chen and Guo 2018), and boosting (Chen and Guo 2018). The perturbation mechanisms can also disguise the training images in deep learning frameworks (Sharma and Chen 2018) to achieve much lower training costs than cryptographic protocols (Mohassel and Zhang 2017). Furthermore, the perturbation methods often do not involve expensive cryptographic primitives. Consequentially, one can observe significant cost savings in the entire life cycle of data analytics, including data submission, computation, and communication amongst the involved parties.

Insight. *The key idea of perturbation approaches is to identify a certain high-level utility in training datasets and preserve it in secure randomized transformations. Similar ideas have also been explored in the cryptographic domain, such as order-preserving encryption (Boldyreva et al. 2011; Boldyreva et al. 2009; Kerschbaum 2015) and encrypted keyword search (Golle et al. 2004; Curtmola et al. 2011).*

Gap. *Despite their efficiency, perturbation approaches face two critical weaknesses. First, perturbation methods may cause significant degradation to the data quality and introduce significant trade-offs between utility and confidentiality. Second, there is no systematic framework for*

analyzing the protection level guaranteed by a perturbation method. Some of them are known not to provide provable semantic security (Chen and Liu 2011; Xu et al. 2012). However, under a clear, rigorous threat model definition and thorough analysis, these methods will have high practical values in the venues where users can accept the specific threat model.

Hardware-Assisted approaches

During the past few years, hardware-assisted trustworthy computing has made a significant breakthrough. In particular, several CPU manufactures have implemented the trusted execution environment (TEE) platforms, among which the most popular one is Intel's Software Guard Extensions (SGX) (Costan and Devadas 2016). We will take SGX as an example in the following. SGX defines a specific memory area (e.g., the *enclave*). Only the authorized owner can run programs and access data in the enclave via special instructions. Owners and users gain access rights via an *attestation* protocol. SGX minimizes the trust boundary to the enclave, which means even though the entire operating system is compromised, adversaries cannot access the enclave. The physical enclave memory is limited (less than 100MB is usable by users). When the enclave memory pages are swapped out/in by the virtual memory management subsystem of the OS[3], they are encrypted/decrypted by the SGX library functions implicitly. SGX uses AES encryption, and thus the encryption and decryption costs are much lower than the primitives we have discussed so far. Besides, since the enclave program works on decrypted data, there is no need to develop special CML algorithms for running inside the enclave, making SGX an appealing platform for developing CML solutions for complex algorithms working with large data.

However, there are a few challenges for migrating algorithms to the SGX environment. First, users need to learn the whole SGX working mechanism and learn to use special instructions and APIs, which can be inconvenient. A few efforts have simplified the migration of applications to SGX, among which the Graphene-SGX library OS (Tsai et al. 2017), SCONE (Arnautov et al. 2016), and Panoply (Shinde et al. 2017) are the most well-known. With a tool like Graphene-SGX, developing CML solutions becomes more straightforward. Lee et al. (2020) have tried to migrate machine learning algorithms to SGX based on Graphene-SGX. However, these methods do not address side-channel attacks.

Second, side-channel attacks are considered the primary threat to SGX-based applications. As TEEs have prevented many traditional attacks and the assumption is now changed to adversary-controlled OS, side-channel attacks are active research areas. Memory side channels and cache side channels are the two types that researchers mostly examined. Memory side-channel attacks are primarily access pattern attacks (Sasy et al. 2018; Ahmad et al. 2018; Shinde et al. 2016). As the encrypted data have to be loaded from the file to the untrusted area first and then accessed by the enclave, the access pattern attacks seem inevitable for data-intensive applications like CML. The well-known approach addressing this problem is the Oblivious RAM technique (Goldreich and Ostrovsky 1996), which has been applied to SGX by ZeroTrace (Sasy et al. 2018) and Obliviate (Ahmad et al. 2018). Ohrimenko et al. (2016) also uses oblivious access techniques for multi-party machine learning with SGX. Branching attacks (Shinde et al. 2016) utilize the branching statements and manipulate page faults to extract information, often addressable with oblivious branching instructions such as CMOV (Shinde et al. 2016; Sasy et al. 2018; Alam et al.). Cache side-channel attacks such as cache timing and transient execution state (Bulck et al. 2018; Ristenpart et al. 2009; Kocher et al. 2019; Lipp et al. 2018) utilize the unique CPU architectural features and thus depend on the manufacturers' firmware and software patches to fix. More studies are necessary to explore the full potential and unique problems with SGX-based CML.

Insight. *The TEE, e.g., SGX, techniques can significantly boost CML's performance on untrusted platforms, as the solutions do not involve expensive crypto primitives or protocols. We consider the SGX based CML as a promising direction because it achieves a strong confidentiality guarantee with significant performance benefits compared to other approaches.*

Gap. *The most critical challenge TEEs face is side-channel attacks, especially the access pattern attacks. Also, machine learning algorithms have unique features (e.g., data access, batching, etc.) that may lead to specific attacks that have not been fully explored yet. Another practical concern is that most recent Intel server CPUs still have not had SGX enabled. A few cloud platforms such as Microsoft Azure and IBM Cloud have started offering SGX-enabled instances, and thus we consider this gap of missing public SGX resources will be filled up soon.*

Conclusion

Despite the potential risk of data and model leakages, many resource-constrained data owners use untrusted platforms (e.g., clouds and edges) for training machine learning models. Researchers have been designing and developing confidential machine learning (CML) approaches for outsourced data using cryptographic primitives and various composition strategies. The overall goal of CML is to protect the confidentiality of data, model, and intermediate results from the untrusted

[3]The enclave virtual memory management is only enabled on the Linux system for early versions of SGX, which might be changed in newer versions of SGX

platforms while also preserving the trained model quality with acceptable costs.

We have reviewed the recent significant CML developments under a systemization framework, focusing on the cryptographic approaches. We have briefly described the cryptographic primitives that are the backbone of the CML approaches and compared their costs in implementing basic operations. While the homogeneous methods that rely on a single cryptographic primitive are straightforward, their solutions are too expensive to be practical. Thus, we focus on the primary design trend of the hybrid composition framework under the decomposition-mapping-composition (DMC) procedure and the selection of crypto-friendly alternative learning algorithms. We describe the critical issues such as the switching between multiple primitives and the principles of identifying crypto-friendly machine learning algorithms. Finally, we include a brief discussion of related approaches and new directions, including the perturbation and hardware-assisted methods. At the end of most sections, we have also included a concise summary area labeled with *Insight* and *Gap* for readers to get the section gist conveniently. We believe this survey can be valuable to both researchers and practitioners in building more complex and practical CML solutions in the future.

Acknowledgements
Not applicable.

Authors' contributions
The authors have contributed equally to this work. The author(s) read and approved the final manuscript.

Funding
This work is partially supported by the National Science Foundation under grant no. 1245847 and the National Institute of Health under grant no. 1R43AI136357-01A1.

Availability of data and materials
Not applicable.

Declarations

Competing interests
Not applicable.

Author details
[1]Northwestern Mutual Data Science Associate Professor Director of Trustworthy and Intelligent Computing Lab Department of Computer Science Marquette University Milwaukee, Wisconsin, USA. [2]HP Inc., USA.

Received: 14 January 2021 Accepted: 27 April 2021
Published online: 01 September 2021

References

Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security CCS '16. ACM, New York, NY, USA. pp 308–318. http://doi.org/10.1145/2976749.2978318.http://doi.acm.org/10.1145/2976749.2978318

Acar A, Aksu H, Uluagac AS, Conti M (2018) A survey on homomorphic encryption schemes: Theory and implementation. ACM Comput Surv 51(4). https://doi.org/10.1145/3214303

Aggarwal CC, Yu PS (2008) Privacy-preserving data mining: models and algorithms. Springer Science & Business Media

Agrawal R, Srikant R (2000) Privacy-preserving data mining. In: Proceedings of ACM SIGMOD Conference. ACM, Dallas, Texas. pp 439–450

Ahmad A, Kim K, Sarfaraz MI, Lee B (2018) OBLIVIATE: A Data Oblivious Filesystem for Intel SGX. In: NDSS, San Diego

Alam AKMM, Sharma S, Chen K Sgx-mr: Regulating dataflows for protecting access patterns of data-intensive sgx applications. Proc Priv Enhancing Technol 2021(1):5–20. https://doi.org/10.2478/popets-2021-0002. Accessed 01 Jan 2021

Aldeen YAAS, Salleh M, Razzaque MA (2015) A comprehensive review on privacy preserving data mining. SpringerPlus 4(1):694. https://doi.org/10.1186/s40064-015-1481-x

Arnautov S, Trach B, Gregor F, Knauth T, Martin A, Priebe C, Lind J, Muthukumaran D, O'Keeffe D, Stillwell ML, Goltzsche D, Eyers D, Kapitza R, Pietzuch P, Fetzer C (2016) Scone: Secure linux containers with intel sgx. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation OSDI'16. USENIX Association, Berkeley, CA, USA. pp 689–703

Asharov G, Lindell Y, Schneider T, Zohner M (2013) More efficient oblivious transfer and extensions for faster secure computation. In: 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013. pp 535–548. https://doi.org/10.1145/2508859.2516738

Boldyreva A, Chenette N, Lee Y, O'Neill A (2009) Order preserving symmetric encryption. In: Proceedings of EUROCRYPT Conference

Boldyreva A, Chenette N, O'Neill A (2011) Order-preserving encryption revisited: Improved security analysis and alternative solutions. In: Annual Cryptology Conference. Springer, Santa Barbara. pp 578–595

Bost R, Popa RA, Tu S, Goldwasser S (2015) Machine learning classification over encrypted data. In: NDSS, vol 4324, San Diego. p 4325

Brakerski Z, Gentry C, Vaikuntanathan V (2014) (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans Comput Theory (TOCT) 6(3):1–36

Bulck JV, Minkin M, Weisse O, Genkin D, Kasikci B, Piessens F, Silberstein M, Wenisch TF, Yarom Y, Strackx R (2018) Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In: 27th USENIX Security Symposium (USENIX Security 18). USENIX Association, Baltimore, MD. pp 991–1008. https://www.usenix.org/conference/usenixsecurity18/presentation/bulck

Bunn P, Ostrovsky R (2007) Secure two-party k-means clustering. In: Proceedings of the 14th ACM Conference on Computer and Communications Security CCS '07. ACM, New York, NY, USA. pp 486–497. https://doi.org/10.1145/1315245.1315306

Canetti R, Canetti R. (2001) Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp 136–145. https://doi.org/10.1109/SFCS.2001.959888

Chakarov D, Papazov Y (2019) Evaluation of the complexity of fully homomorphic encryption schemes in implementations of programs. In: Proceedings of the 20th International Conference on Computer Systems and Technologies CompSysTech '19. Association for Computing Machinery, New York, NY, USA. pp 62–67. https://doi.org/10.1145/3345252.3345292

Chen A (2010) Gcreep: Google engineer stalked teens, spied on chats. Gawker September

Chen G, Guo S (2018) RASP-Boost: Confidential Boosting-Model Learning with Perturbed Data in the Cloud. IEEE Trans Cloud Comput 6(2):584–597

Chen K, Liu L (2011) Geometric data perturbation for privacy preserving outsourced data mining. Knowl Inf Syst 29(3):657–695. https://doi.org/10.1007/s10115-010-0362-4

Cheon JH, Kim A, Kim M, Song Y (2017) Homomorphic encryption for arithmetic of approximate numbers. In: Takagi T, Peyrin T (eds). Advances in Cryptology – ASIACRYPT 2017. Springer, Cham. pp 409–437

Chillotti I, Gama N, Georgieva M, Izabachène M (2020) TFHE: fast fully homomorphic encryption over the torus. J. Cryptology 33(1):34–91. https://doi.org/10.1007/s00145-019-09319-x

Costan V, Devadas S (2016) Intel sgx explained. IACR Cryptol ePrint Archive 2016:86

Curtmola R, Garay J, Kamara S, Ostrovsky R (2011) Searchable symmetric encryption: improved definitions and efficient constructions. J Comput Secur 19(5):895–934

Dahl M, Ning C, Toft T (2012) On secure two-party integer division. In: Keromytis AD (ed). Financial Cryptography and Data Security. Springer, Berlin, Heidelberg. pp 164–178
Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
Demmler D, Schneider T, Zohner M (2015) ABY - A framework for efficient mixed-protocol secure two-party computation. In: 22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015. https://www.ndss-symposium.org/ndss2015/aby---framework-efficient-mixed-protocol-secure-two-party-computation
Dreier J, Kerschbaum F (2011) Practical privacy-preserving multiparty linear programming based on problem transformation. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing. IEEE, Los Alamitos. pp 916–924
Du W, Zhan Z (2003) Using randomized response techniques for privacy-preserving data mining. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, DC. pp 505–510
Duncan AJ, Creese S, Goldsmith M (2012) Insider attacks in cloud computing. In: 2012 IEEE 11th international conference on trust, security and privacy in computing and communications. IEEE, Liverpool. pp 857–862
Erlingsson U, Pihur V, Korolova A (2014) Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security CCS '14. ACM, New York, NY, USA. pp 1054–1067. https://doi.org/10.1145/2660267.2660348
Evans D, Kolesnikov V, Rosulek M (2018) A Pragmatic Introduction to Secure Multi-Party Computation. Found Trends Priv Secur 2(2-3):70–246. https://doi.org/10.1561/3300000019
Fredrikson M, Jha S, Ristenpart T (2015) Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver. pp 1322–1333
Fredrikson M, Lantz E, Jha S, Lin S, Page D, Ristenpart T (2014) Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In: 23rd USENIX Security Symposium USENIX Security. USENIX Association, San Diego, CA. pp 17–32
Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Annual ACM Symposium on Theory of Computing. ACM, New York, NY, USA. pp 169–178
Gilad-Bachrach R, Dowlin N, Laine K, Lauter K, Naehrig M, Wernsing J (2016) Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In: International Conference on Machine Learning. PMLR, New York City. pp 201–210
Goldreich O, Micali S, Wigderson A (1987) How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing STOC '87. ACM, New York, NY, USA. pp 218–229. https://doi.org/10.1145/28395.28420
Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious ram. J ACM 43:431–473
Golle P (2006) A private stable matching algorithm. In: International Conference on Financial Cryptography and Data Security. Springer, Anguilla. pp 65–80
Golle P, Staddon J, Waters B (2004) Secure Conjunctive Keyword Search over Encrypted Data. In: Jakobsson M, Yung M, Zhou J (eds). Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings, vol 3089. Springer. pp 31–45. https://doi.org/10.1007/978-3-540-24852-1_3
Graepel T, Lauter K, Naehrig M (2012) ML confidential: Machine learning on encrypted data. In: International Conference on Information Security and Cryptology. Springer, Seoul. pp 1–21
Grigorescu S, Trasnea B, Cocias T, Macesanu G (2019) A survey of deep learning techniques for autonomous driving. J Field Robot 37:3
Garay JA, Gennaro R (2014) Algorithms in HElib. In: Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. Lecture Notes in Computer Science, vol 8616. Springer. pp 554–571. https://doi.org/10.1007/978-3-662-44371-2_31
Halevi S, Shoup V (2013) Design and implementation of a homomorphic-encryption library. IBM Res (Manuscr) 6(12-15):8–36
Hastie T, Tibshirani R, Friedman J (2001) The Elements of Statistical Learning. Springer, New York City, New York
Henecka W, K ögl S, Sadeghi A-R, Schneider T, Wehrenberg I (2010) Tasty: Tool for automating secure two-party computations. In: Proceedings of the 17th ACM Conference on Computer and Communications Security CCS '10. ACM, New York, NY, USA. pp 451–462. https://doi.org/10.1145/1866307.1866358
Hesamifard E, Takabi H, Ghasemi M (2017) Cryptodl: Deep neural networks over encrypted data. CoRR abs/1711.05189. http://arxiv.org/abs/1711.05189
Hitaj B, Ateniese G, Perez-Cruz F (2017) Deep models under the gan: Information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security CCS '17. ACM, New York, NY, USA. pp 603–618. https://doi.org/10.1145/3133956.3134012
Huang Y, Evans D, Katz J, Malka L (2011) Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium, vol 201. USENIX, San Francisco. pp 331–335
Ji Z, Lipton ZC, Elkan C (2014) Differential Privacy and Machine Learning: a Survey and Review. CoRR abs/1412.7584. http://arxiv.org/abs/1412.7584
Jiang X, Kim M, Lauter K, Song Y (2018) Secure outsourced matrix computation and application to neural networks. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security CCS '18. ACM, New York. pp 1209–1222. https://doi.org/10.1145/3243734.3243837
Kerschbaum F (2015) Frequency-hiding order-preserving encryption. In: Proceedings of ACM Conference on Computer and Communication Security
Kocher P, Horn J, Fogh A, Genkin D, Gruss D, Haas W, Hamburg M, Lipp M, Mangard S, Prescher T, et al. (2019) Spectre attacks: Exploiting speculative execution. In: 2019 IEEE Symposium on Security and Privacy (SP). IEEE, San Francisco. pp 1–19
Kolesnikov V, Schneider T (2008) Improved garbled circuit: Free XOR gates and applications. In: International Colloquium on Automata, Languages, and Programming, Springer, Reykjavik. pp 486–498
Lazzeretti R, Barni M (2011) Division between encrypted integers by means of garbled circuits. In: 2011 IEEE International Workshop on Information Forensics and Security. pp 1–6. https://doi.org/10.1109/WIFS.2011.6123132
LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
Lee D, Kuvaiskii D, Vahldiek-Oberwagner A, Vij M (2020) Privacy-preserving machine learning in untrusted clouds made simple. CoRR abs/2009.04390. http://arxiv.org/abs/2009.043902009.04390
Lindell Y (2017) How to Simulate It – A Tutorial on the Simulation Proof Technique. In: Lindell Y (ed). Tutorials on the Foundations of Cryptography (2017). Springer, Cham. pp 277–346
Lindell Y (2020) Secure Multiparty Computation (MPC). Cryptology ePrint Archive, Report 2020/300. https://eprint.iacr.org/2020/300
Lipp M, Schwarz M, Gruss D, Prescher T, Haas W, Fogh A, Horn J, Mangard S, Kocher P, Genkin D, et al. (2018) Meltdown: Reading kernel memory from user space. In: 27th {USENIX} Security Symposium ({USENIX} Security 18), Baltimore. pp 973–990
Liu K, Kargupta H, Ryan J (2006) Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. IEEE Trans Knowl Data Eng (TKDE) 18(1):92–106
Liu Q, Li P, Zhao W, Cai W, Yu S, Leung VCM (2018) A survey on security threats and defensive techniques of machine learning: A data driven view. IEEE Access 6:12103–12117. https://doi.org/10.1109/ACCESS.2018.2805680
Liu C, Wang XS, Nayak K, Huang Y, Shi E (2015) Oblivm: A programming framework for secure computation. In: 2015 IEEE Symposium on Security and Privacy. pp 359–376. https://doi.org/10.1109/SP.2015.29
Lu W, Kawasaki S, Sakuma J (2016) Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data. IACR Cryptol ePrint Arch 2016:1163
Mansfield-Devine S (2015) The Ashley Madison affair. Netw Secur 2015(9):8–16
Matwin S (2013) Privacy-Preserving Data Mining Techniques: Survey and Challenges. In: Custers B, Calders T, Schermer B, Zarsky T (eds). Discrimination and Privacy in the Information Society. Springer, Berlin. pp 209–221
Mohassel P, Rindal P (2018) ABY^3: A Mixed Protocol Framework for Machine Learning. In: Lie D, Mannan M, Backes M, Wang X (eds). Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications

Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. ACM. pp 35–52. https://doi.org/10.1145/3243734.3243760
Mohassel P, Zhang Y (2017) Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE, San Jose. pp 19–38
Naehrig M, Lauter K, Vaikuntanathan V (2011) Can homomorphic encryption be practical?. In: Proceedings of Cloud Computing Security Workshop. ACM, New York, NY, USA. pp 113–124
Nikolaenko V, Ioannidis S, Weinsberg U, Joye M, Taft N, Boneh D (2013) Privacy-preserving matrix factorization. In: ACM SIGSAC Conference on Computer and Communications Security. pp 801–812
Nikolaenko V, Weinsberg U, Ioannidis S, Joye M, Boneh D, Taft N (2013) Privacy-preserving ridge regression on hundreds of millions of records. In: IEEE Symposium on Security and Privacy. pp 334–348
Ohrimenko O, Schuster F, Fournet C, Mehta A, Nowozin S, Vaswani K, Costa M (2016) Oblivious multi-party machine learning on trusted processors. In: Holz T, Savage S (eds). 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016. USENIX Association. pp 619–636. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/ohrimenko
Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. Springer, Berlin. pp 223–238
Papernot N, McDaniel P, Sinha A, Wellman MP (2018) Sok: Security and privacy in machine learning. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, London. pp 399–414
Patra A, Suresh A (2020) BLAZE: Blazing Fast Privacy-Preserving Machine Learning. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020. The Internet Society, San Diego. https://www.ndss-symposium.org/ndss-paper/blaze-blazing-fast-privacy-preservingmachine-learning/
Phong LT, Aono Y, Hayashi T, Wang L, Moriai S (2018) Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans Inf Forensics Secur 13(5):1333–1345. https://doi.org/10.1109/TIFS.2017.2787987
Rane S, Sun W (2010) Privacy preserving string comparisons based on levenshtein distance. In: 2010 IEEE International Workshop on Information Forensics and Security. pp 1–6. https://doi.org/10.1109/WIFS.2010.5711449
Riazi MS, Weinert C, Tkachenko O, Songhori EM, Schneider T, Koushanfar F (2018) Chameleon: A hybrid secure computation framework for machine learning applications. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security ASIACCS '18. Association for Computing Machinery, New York, NY, USA. pp 707–721. https://doi.org/10.1145/3196494.3196522
Ristenpart T, Tromer E, Shacham H, Savage S (2009) Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM conference on Computer and Communications Security. ACM, New York. pp 199–212
Rouhani BD, Hussain SU, Lauter K, Koushanfar F (2018) Redcrypt: Real-time privacypreserving deep learning inference in clouds using fpgas. ACM Trans Reconfigurable Technol Syst (TRETS) 11(3):1–21
Rouhani BD, Riazi MS, Koushanfar F (2018) Deepsecure: Scalable provably-secure deep learning. In: Proceedings of the 55th Annual Design Automation Conference DAC '18. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3195970.3196023
Sachan A, Roy D, Arun PV (2013) An analysis of privacy preservation techniques in data mining. In: Meghanathan N, Nagamalai D, Chaki N (eds). Advances in Computing and Information Technology. Springer, Berlin, Heidelberg. pp 119–128
Sarwate AD, Chaudhuri K (2013) Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. IEEE Signal Proc Mag 30(5):86–94. https://doi.org/10.1109/MSP.2013.2259911
Sasy S, Gorbunov S, Fletcher CW (2018) ZeroTrace: Oblivious Memory Primitives from Intel SGX. In: NDSS, San Diego
Schapire RE (1999) A brief introduction to boosting. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2 IJCAI'99. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp 1401–1406
Shan Z, Ren K, Blanton M, Wang C (2018) Practical secure computation outsourcing: A survey. ACM Comput Surv 51:2
Sharma S, Chen K (2018) Image disguising for privacy-preserving deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, New York. pp 2291–2293
Sharma S, Chen K (2019) Confidential boosting with random linear classifiers for outsourced user-generated data. In: European Symposium on Research in Computer Security. Springer, Cham. pp 41–65
Sharma S, Chen K, Sheth A (2018) Toward practical privacy-preserving analytics for iot and cloud-based healthcare systems. IEEE Internet Comput 22(2):42–51. https://doi.org/10.1109/MIC.2018.112102519
Sharma S, Powers J, Chen K (2019) Privategraph: Privacy-preserving spectral analysis of encrypted graphs in the cloud. IEEE Trans Knowl Data Eng 31(5):981–995. https://doi.org/10.1109/TKDE.2018.2847662
Shinde S, Chua ZL, Narayanan V, Saxena P (2016) Preventing page faults from telling your secrets. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security ASIACCS16. Association for Computing Machinery, New York, NY, USA. pp 317–328. https://doi.org/10.1145/2897845.2897885
Shinde S, Tien DL, Tople S, Saxena P (2017) Panoply: Low-TCB Linux Applications With SGX Enclaves. In: NDSS, San Diego
Shokri R, Shmatikov V (2015) Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, New York City. pp 1310–1321
Shokri R, Stronati M, Song C, Shmatikov V (2017) Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE, San Jose. pp 3–18
Song C, Shmatikov V (2019) Overlearning reveals sensitive attributes. arXiv preprint arXiv:1905.11742
Tramèr F, Zhang F, Juels A, Reiter MK, Ristenpart T (2016) Stealing machine learning models via prediction apis. In: Proceedings of the 25th USENIX Conference on Security Symposium SEC'16. USENIX Association, USA. pp 601–618
Tsai C, Porter DE, Vij M (2017) Graphene-sgx: A practical library OS for unmodified applications on SGX. In: Silva DD, Ford B (eds). 2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12-14, 2017. pp 645–658
Unger L (2015) Breaches to customer account data. Comput Internet Lawyer 32(2):14–20
Veugen T (2014) Encrypted integer division and secure comparison. Int J Appl Crypt 3(2):166
Wu X, Kumar V, Ross Quinlan J, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS, Zhou Z-H, Steinbach M, Hand DJ, Steinberg D (2007) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
Xu H, Guo S, Chen K (2012) Building confidential and efficient query services in the cloud with RASP data perturbation. IEEE Trans Knowl Data Eng 26(2):322–335
Yao AC (1986) How to generate and exhange secrets. In: IEEE Symposium on Foundations of Computer Science. pp 162–167
Zahur S, Rosulek M, Evans D (2015) Two Halves Make a Whole. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin

Publisher's Note